PHYC30300- Advanced Laboratory I

Distribution of Cooking Times

Nathan Power

19311361

Abstract

This experiment investigates the qualitative nature of a poisson distribution and the relationship between poisson distribution and exponential distribution as well as the parameters which shape the distribution themselves. The main question tackled in this report is "does the qualitative nature of the distribution change if the sample size is changed?" . Here, we show that by repeating a random variable experiment for different sample sizes that the decay constant of the time interval curve varies with the sample size. The results found in this experiment personally have furthered my understanding of counting statistics experiments and both Poisson and exponential distributions. This knowledge will be useful in my academic studies in areas such as statistics and quantum mechanics.

Introduction

In this experiment, we investigate the qualitative nature of a Poisson distribution.
We investigate the relationship between Poisson distribution and exponential distribution as well as the parameters which shape the distribution themselves.

In everyday life, we observe a number of phenomena in which events occur more or less at random. Events which occur perfectly at random are phenomena which are not affected by the occurrence of other events. For example, consider the rate of yawning in a classroom of students.This appears to be random with people yawning whenever they need to. However, is it really random? Does some person yawn because he sees another student yawn? Will a boring topic cause a number of students to yawn? If such factors are present, then the events have some correlation and so do not occur perfectly at random. A way to test for randomness is to set up a model for events occurring at random and from it to develop a theoretical distribution for a series of time intervals.[1]

In this experiment, we investigate if the qualitative nature of the distribution changes if the sample size and/or number of trials is changed for the same random phenomenon? We will address this problem by using a random number generator to simulate a random event and by changing the sample size parameter of each simulation while keeping all other variables constant, we will compare the qualitative nature of the distributions.

Theory

In probability and statistics, a random variable is a variable whose value is subject to variations due to chance. Random variables can be classified as either discrete or as continuous.[2]

A discrete random variable is a variable that can take only a finite number of distinct values (e.g integers). Discrete random variables are usually, but not always, counts. Examples of discrete random variables include the number of children in a family, the Friday night attendance at a cinema, the number of patients in a doctor's surgery, the number of defective light bulbs in a box of ten.[3]

Continuous random variables, on the other hand, take on values that vary continuously within an interval .Therefore, continuous random variables take an infinite number of possible values. A continuous random variable is not defined at specific values, it has an uncountable infinite number of possible values, all of which have probability 0. It is defined over an interval of values.[1]A continuous random variable is characterised by its probability density function, a graph which has a total area of 1 beneath it: The probability of the random variable taking values in any interval is simply the area under the curve over that interval. Examples include height, weight, the amount of sugar in an orange, the time required to run a mile.[3]

The Poisson distribution is used to describe discrete quantitative data in which the population size is large, the probability of an individual event is small. Current examples can be the number of covid cases in a town per day, or the number of admissions to a particular hospital. Here a version of a Poisson distribution is used to quantify superspreading for COVID-19. [4]

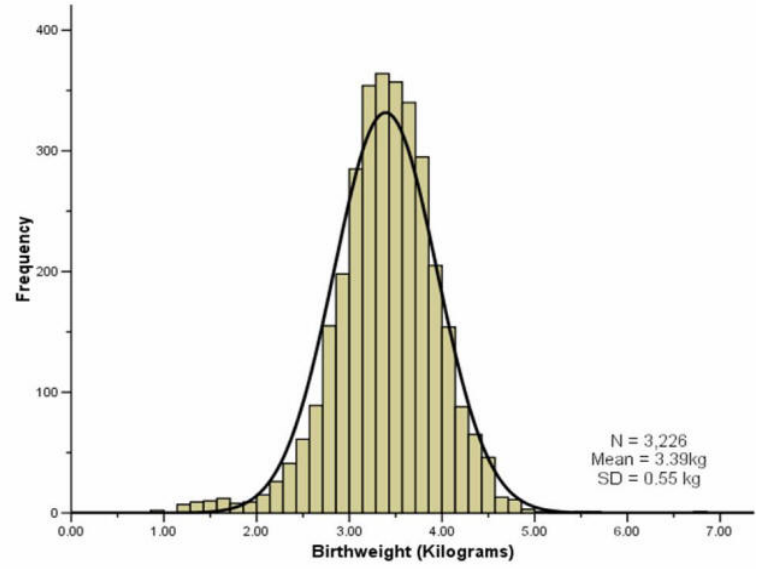


Fig (1) Poisson distribution showing the frequency of the birthweight of babies [5]

For a process to be a poisson process it must meet the following criteria :

1. Events are independent of each other. The occurrence of one event does not affect the probability another event will occur.
2. The average rate, events per time period, is constant.
3. Two events cannot occur simultaneously.[6]

The Poisson distribution deals with the number of occurrences in a fixed period of time, and the exponential distribution deals with the time between occurrences of successive events as time flows by continuously. [1]

Consider the time between successive orders at a pizzeria, or between successive calls to a radio show. These times are typically exponentially distributed.

If the mean interarrival time is $1/\lambda$, then the variance will be $1/\lambda^2$ and the standard deviation will be $1/\lambda$. [1]

The exponential distribution has a memoryless property, which means it “forgets” what has come before it. In other words, if you continue to wait, the length of time you wait neither increases nor decreases the probability of an event happening. Any time may be marked down as time zero.[1] After waiting a minute without a call, the probability of a call arriving in the next two minutes is the same as was the probability of getting a call in the following two minutes. As you continue to wait, the chance of something happening neither increases or decreases. This memorylessness is also known as the Markov property.

Assume we count the number of times, x, a given random event occurs in a certain time interval, t, with a mean number occurrences per interval, λ. The number of counts per interval will be distributed around this average value. Numbers close to the average will be recorded frequently, numbers very different from the average will be recorded infrequently. [2] For a large number of frequent events we find that the probability of recording exactly x occurrences in t is given by:[6]

$$P(x) = \frac{\lambda t^x}{x!} . \ e^{-\lambda t} \qquad \text{Eq.1}$$

The question posed to us is;

An industrious physics student finds a job at a local fast food restaurant to help her pay her way through college. Her task is to cook 20 hamburgers on a grill at any one time. When a hamburger is cooked, she is supposed to replace it with an uncooked hamburger. However, the physics student does not pay attention to whether the hamburger is cooked or not.

Her method is to choose a hamburger at random and replace it by an uncooked one and not bother to check whether the hamburger that she removes from the grill is cooked or not.

<u>What is the distribution of cooking times of the hamburgers that she removes?</u>

To simplify the problem, assume that she replaces a random hamburger at regular intervals of one minute and that there is an indefinite supply of uncooked hamburgers.

Does the qualitative nature of the distribution change if she cooks 40 or 200 hamburgers at any one time?

The first step we took to solving this problem was to find a way to use a random number generator to choose one hamburger out of 20 with equal probability at regular intervals of one minute and replace it with a raw hamburger.

Firstly, we wanted our function to output the time the burger had been on the grill. With this in mind, our first function, “burgers” from Appendix A Section 1.2, took the inputs n and t, which correspond to the number of burgers on the grill and amount of trials. This started with an initial condition of all raw burgers(Time=0), after a minute (Time = 1) and a burger was selected at random and replaced with a raw burger(Time=0). After this, one minute was added to the cooling time of every burger and another loop was engaged until the given amount of loops had been fulfilled. However, the main problem with this function was that it gave us the cooking times of the burgers still on the grill. We are looking for the removed burgers cooking times.

Next, we aimed to improve our function so that it would give us the cooking times of the replaced burgers. Our second function, “burgers2” from Appendix A Section 2.1 managed to achieve this. Before a new burger is put onto the grill, “burgers2” took the replaced burger and placed its cooking time into a separate array called “removed”. Outputting this array gave us the cooking times for the replaced burgers.

In an attempt to further this function, “burgers3” of Appendix A Section 3.1, we tried to start with all raw hamburgers. i.e. after the first lot has been fully replaced and initial conditions have been forgotten. To achieve this, we only started recording cooking times of the replacement burgers after an n amount of burgers had been removed.

Plotting our output array of cooking times for various numbers of burgers on the grill at any one time and amount of trials recorded we were able to compare the distribution of cooking times and thus determine if the qualitative nature of the distribution changed with these variables.

Results and Discussion

Increasing the number of trials

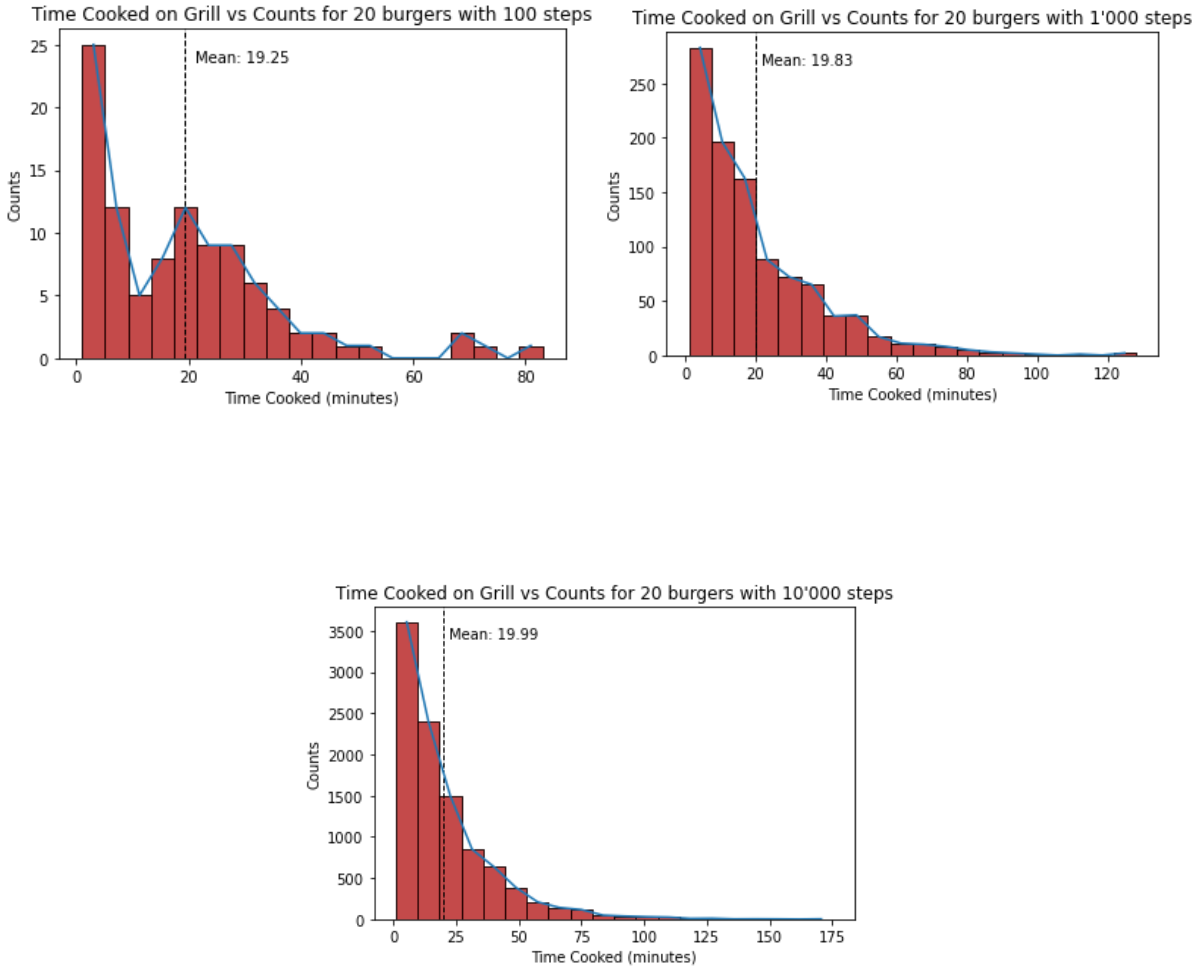


Fig.2a, Fig.2b and Fig2.c histograms showing the cooking time distribution for n=20 and various timesteps

Above, cooking histograms displaying time distributions for 20 burgers on the grill for timesteps of 100, 1’000, 10’000 have been plotted. We can see these plots all resemble decaying exponential functions.These are examples of a right-skewed exponential distribution. This can be also referred to as positively skewed. We can see that as the number of trials increased, the smoother the decay of the plot was. We can see as the number of trials increased, the closer the mean came to the expected theoretical mean value of 20. Increasing the number of trials carried out gives a more accurate representation of the probabilities of each outcome occurring. This matches what is expected theoretically.

Table1. cooking time distribution for n=20			
Number of trials	10’000	1’000	100
Mean	19.99	19.83	19.25

Increasing the number of burgers on the grill

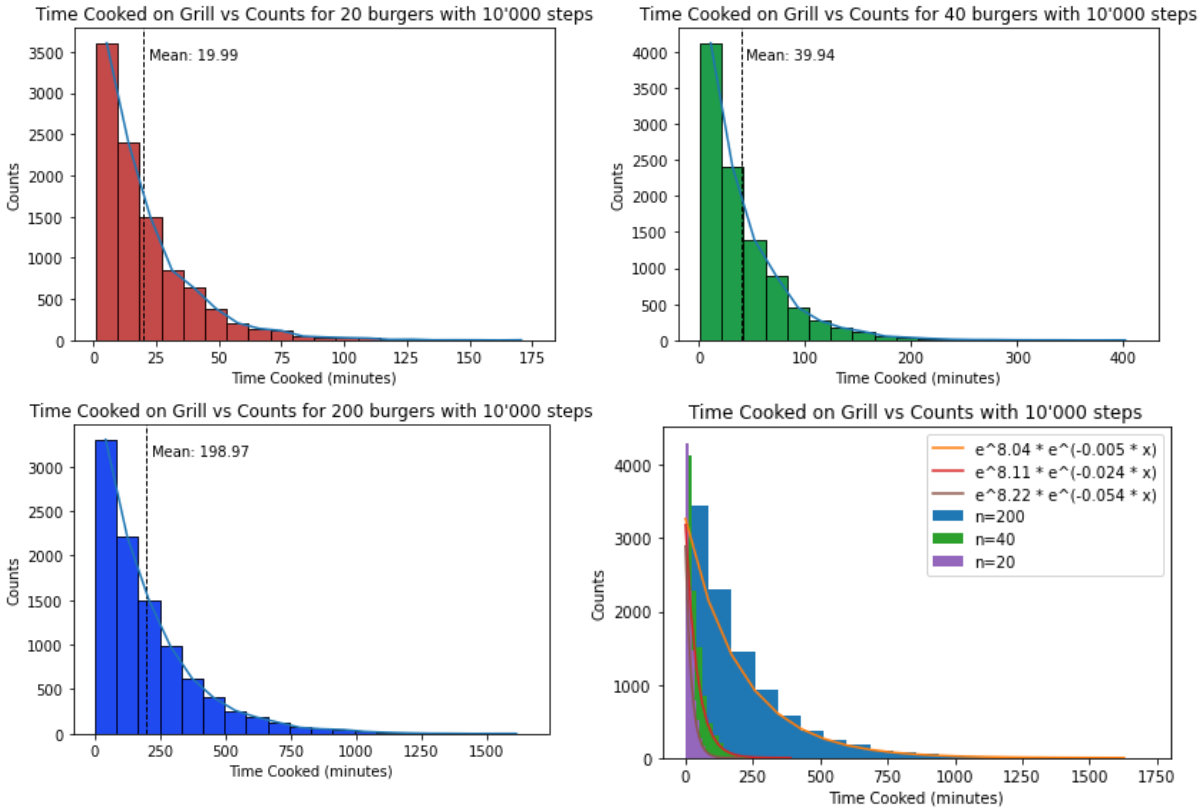


Fig.3a, Fig.3b and Fig3.c histograms showing the cooking time distribution for various n values for 10’000 trials
Fig.3d shows all distributions together on the same plot

Above, cooking histograms displaying time distributions for 10’000 trials and n values of 20, 40 and 200. We can see,yet again, these plots all resemble right-skewed decaying exponential functions. We can see that as the number of burgers on the grill at one time increased, the higher the mean was. This also meant that the decay constant of each graph becomes less steep and decays slower with increasing size of n. We can also see for a higher n value that the outcome for cooking time has a larger spread. This means that the cooking times have a larger amount of possible outcomes. This matches what is expected theoretically from increasing the value of n.

Table2 . cooking time distribution for 10’000 trials			
Number of burgers on grill	20	40	200
Mean	19.99	39.94	198.97
Decay Constant	≈0.05	≈0.025	≈0.005

Conclusion

In this experiment, we aimed to investigate the qualitative nature of a Poisson distribution We investigated the relationship between Poisson distribution and exponential distribution as well as the parameters which shape the distribution themselves.
From our results, it can be seen that by increasing the number of trials carried out and by increasing the number of burgers on the grill the distribution of cooking times did in fact change. By increasing the amount of burgers on the grill we found that the decay constant for our positively-skewed exponential decreased and was not as sharp. We expected this from theory. We also discovered that by increasing the amount of trials carried out that we found a more accurate representation of our sample and a smoother decaying curve.
In this experiment, we did succeed in answering the question, “*Does the qualitative nature of the distribution change if she cooks 40 or 200 hamburgers at any one time?”.* The results found in this experiment personally have furthered my understanding of counting statistics experiments and both Poisson and exponential distributions. This knowledge will be useful in my academic studies in areas such as statistics and quantum mechanics.

References

[1]M. S. Lafleur, P. F. Hinrichsen, P. C. Landry, et al.,The Physics Teacher 10, 314 (1972); https://doi.org/10.1119/1.2352241 Published Online: 01 September 2006
[2]Discrete Random Variables | Boundless Statistics,Courses.lumenlearning.com,https://courses.lumenlearning.com/boundless-statistics/chapter/discrete-random-variables/
[3]Random Variables,Stat.yale.edu, http://www.stat.yale.edu/Courses/1997-98/101/ranvar.htm#:~:text=A%20continuous%20random%20variable%20is,required
[4] Kremer, C., Torneri, A., Boesmans, S., Meuwissen, H., Verdonschot, S., Driessche, K.V., Althaus, C.L., Faes, C. and Hens, N., 2021. Quantifying superspreading for COVID-19 using Poisson mixture distributions. *Scientific reports*, *11*(1), pp.1-11.
[5] MJ Campbell 2016, S Shantikumar 2016, *HealthKnowledge, Standard Statistical Distributions (e.g. Normal, Poisson, Binomial) and their uses*, Date Last Accessed; 30/11/2021, https://www.healthknowledge.org.uk/public-health-textbook/research-methods/1b-statistical-methods/statistical-distributions
[6]Will Koehrsen,The Poisson Distribution and Poisson Process Explained, https://towardsdatascience.com/the-poisson-distribution-and-poisson-process-explained-4e2cb17d459

Appendix A

Section 1

In [ ]:

```python
#Section 1.1 # importing
import numpy as np
import math
import random
import matplotlib.pyplot as plt
import scipy.stats
from scipy.optimize import curve_fit
from scipy.stats import expon
import matplotlib.pyplot as plt
```

In [3]:

```python
#Section 1.2 # first attempt at a random number function
def burgers(n,t): #array for cooking times of burgers left on grill after t loops
    myarray =  np.zeros(n)+1
    i = 0
    while i < t-1:
        indices = np.random.choice(np.arange(myarray.size), replace=False,size=int(myarray.size * 1/n))
        myarray[indices] = 0
        i += 1
        myarray = myarray + 1
    return myarray
```

In [4]:

```python
burgers(20,1000)
```

Out[4]:

```
array([15., 16., 19., 21., 39., 29.,  3.,  4.,  8.,  1., 28.,  6., 36.,
       11., 56.,  2., 12., 40.,  9.,  7.])
```

Section 2

In [9]:

```python
#section2.1

def burgers2(n,t): #array of cooking times for removed burgers
    myarray =  np.zeros(n)+1 #burgers on grill
    removed = np.array([]) #burgers removed
    i = 0
    while i < t-1:
        indices = np.random.choice(np.arange(myarray.size), replace=False,size=int(myarray.size * 1/n))
        removed = np.append(removed,myarray[indices])
        myarray[indices] = 0
        i += 1
        myarray = myarray + 1
    return removed
```

In [10]:

```python
burgers2(20,20)
```

Out[10]:

```
array([ 1.,  2.,  1.,  4.,  5.,  6.,  3.,  8.,  9.,  2.,  6., 11., 13.,
        8., 15.,  9., 17., 18.,  8.])
```

Section 3

In [14]:

```python
#section 3.1
def burgers3(n,t): #array of cooking times for removed burgers in steady state
    myarray =  np.zeros(n)+1 #burgers on grill
    removed = np.array([]) #burgers removed
    i = 0
    while i < t+(n-1):
        indices = np.random.choice(np.arange(myarray.size), replace=False,size=int(myarray.size * 1/n))
        removed = np.append(removed,myarray[indices])
        myarray[indices] = 0
        i += 1
        myarray = myarray + 1
    return removed[n-1:t+(19)] #only returns after first n burgers are removed
```

Section 4

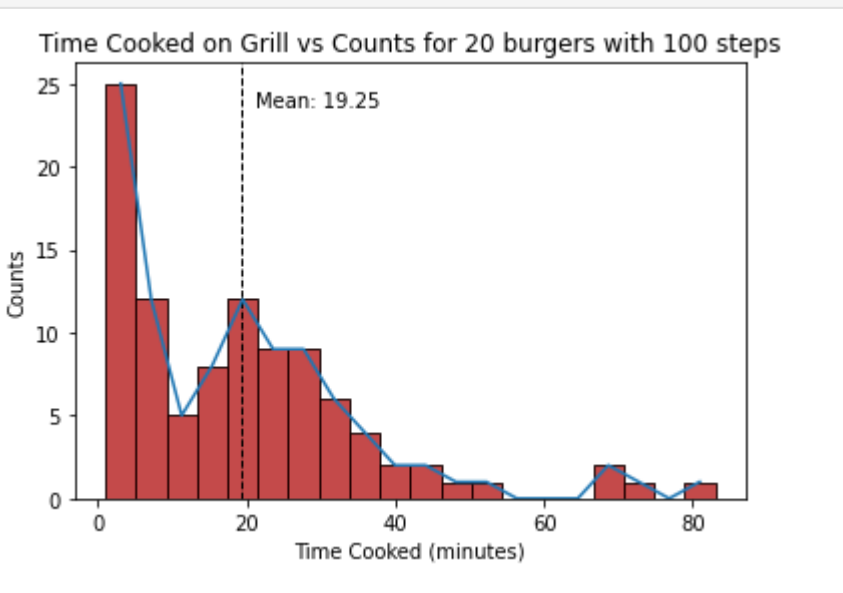
Histogram for 20 burgers

In [266…

```python
#section 4.1
x1 = burgers3(20,100)
result = plt.hist(x1, bins=20, color='c', edgecolor='k', alpha=0.65)
plt.axvline(x1.mean(), color='k', linestyle='dashed', linewidth=1)

min_ylim, max_ylim = plt.ylim()
plt.text(x1.mean()*1.1, max_ylim*0.9, 'Mean: {:.2f}'.format(x1.mean()))
n1, x1, _ = plt.hist(x1,20, color='r', edgecolor='k', alpha=0.65)
bin_centers1 = 0.5*(x1[1:]+x1[:-1])
plt.plot(bin_centers1,n1)


plt.xlabel("Time Cooked (minutes)")
plt.ylabel("Counts")
plt.title("Time Cooked on Grill vs Counts for 20 burgers with 100 steps");
```

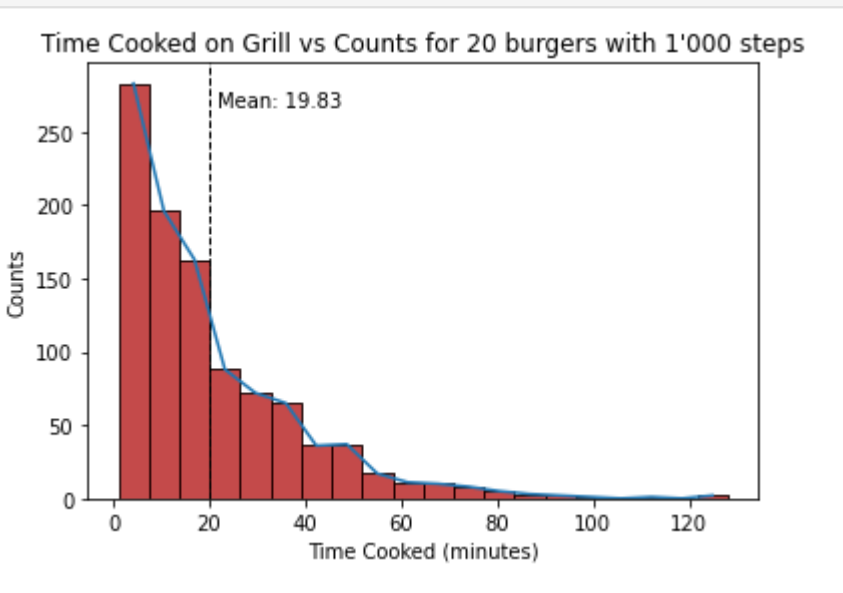


In [247…

```python
#section 4.2
x1 = burgers3(20,1000)
result = plt.hist(x1, bins=20, color='c', edgecolor='k', alpha=0.65)
plt.axvline(x1.mean(), color='k', linestyle='dashed', linewidth=1)

min_ylim, max_ylim = plt.ylim()
plt.text(x1.mean()*1.1, max_ylim*0.9, 'Mean: {:.2f}'.format(x1.mean()))
n1, x1, _ = plt.hist(x1,20, color='r', edgecolor='k', alpha=0.65)
bin_centers1 = 0.5*(x1[1:]+x1[:-1])
plt.plot(bin_centers1,n1)


plt.xlabel("Time Cooked (minutes)")
plt.ylabel("Counts")
plt.title("Time Cooked on Grill vs Counts for 20 burgers with 1'000 steps");
```

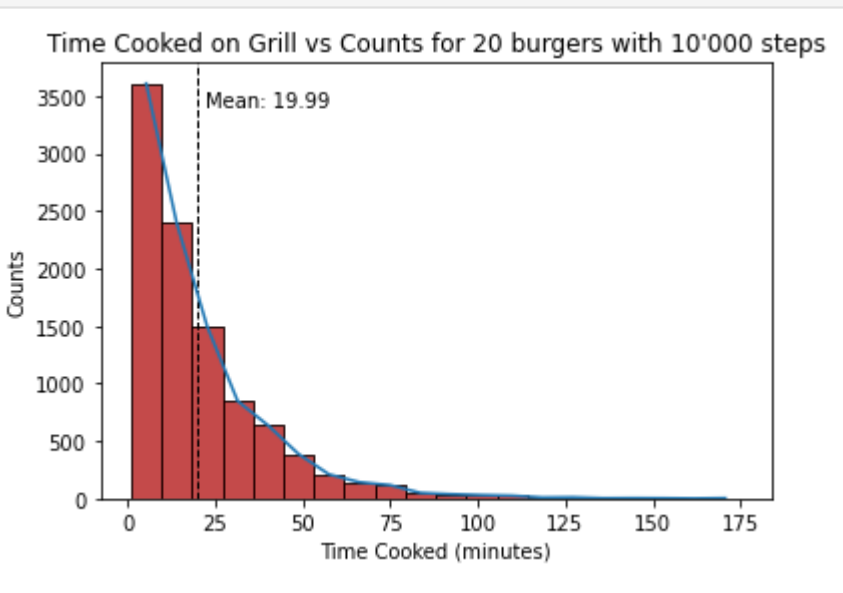


In [242…

```python
#section 4.3
x1 = burgers3(20,10000)
result = plt.hist(x1, bins=20, color='c', edgecolor='k', alpha=0.65)
plt.axvline(x1.mean(), color='k', linestyle='dashed', linewidth=1)

min_ylim, max_ylim = plt.ylim()
plt.text(x1.mean()*1.1, max_ylim*0.9, 'Mean: {:.2f}'.format(x1.mean()))
n1, x1, _ = plt.hist(x1,20, color='r', edgecolor='k', alpha=0.65)
bin_centers1 = 0.5*(x1[1:]+x1[:-1])
plt.plot(bin_centers1,n1)


plt.xlabel("Time Cooked (minutes)")
plt.ylabel("Counts")
plt.title("Time Cooked on Grill vs Counts for 20 burgers with 10'000 steps");
```



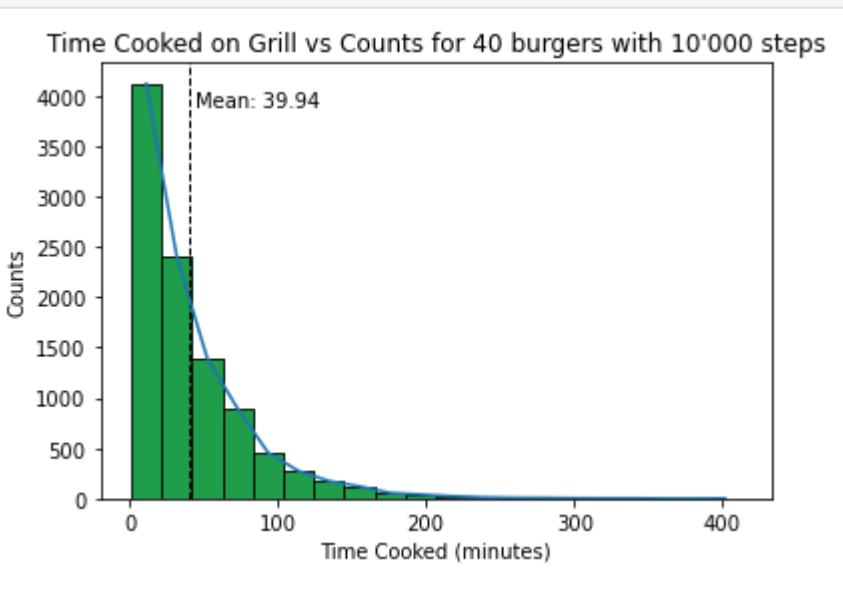
Histogram for 40 burgers

In [241…

```python
#section 4.4
x2 = burgers3(40,10000)
result2 = plt.hist(x2, bins=20, color='c', edgecolor='k', alpha=0.65)
plt.axvline(x2.mean(), color='k', linestyle='dashed', linewidth=1)

min_ylim, max_ylim = plt.ylim()
plt.text(x2.mean()*1.1, max_ylim*0.9, 'Mean: {:.2f}'.format(x2.mean()))
n2, x2, _ = plt.hist(x2,20, color='g', edgecolor='k', alpha=0.65)
bin_centers2 = 0.5*(x2[1:]+x2[:-1])
plt.plot(bin_centers2,n2)

plt.xlabel("Time Cooked (minutes)")
plt.ylabel("Counts")
plt.title("Time Cooked on Grill vs Counts for 40 burgers with 10'000 steps");
```



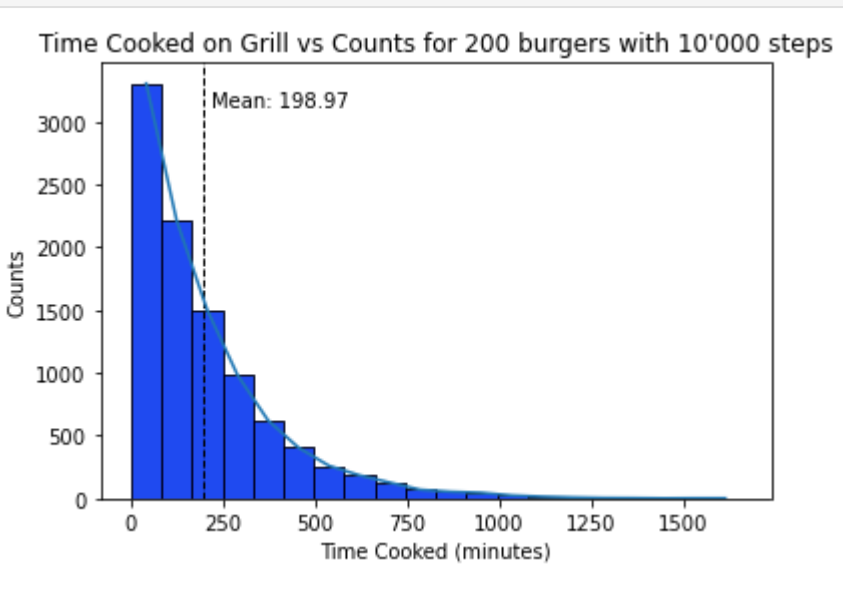
Distribution for 200 burgers

In [243…

```python
#section 4.
x3 = burgers3(200,10000)
result = plt.hist(x3, bins=20, color='c', edgecolor='k', alpha=0.65)
plt.axvline(x3.mean(), color='k', linestyle='dashed', linewidth=1)

min_ylim, max_ylim = plt.ylim()
plt.text(x3.mean()*1.1, max_ylim*0.9, 'Mean: {:.2f}'.format(x3.mean()))
n3, x3, _ = plt.hist(x3,20, color='b', edgecolor='k', alpha=0.65)
bin_centers3 = 0.5*(x3[1:]+x3[:-1])
plt.plot(bin_centers3,n3)

plt.xlabel("Time Cooked (minutes)")
plt.ylabel("Counts")
plt.title("Time Cooked on Grill vs Counts for 200 burgers with 10'000 steps");
```



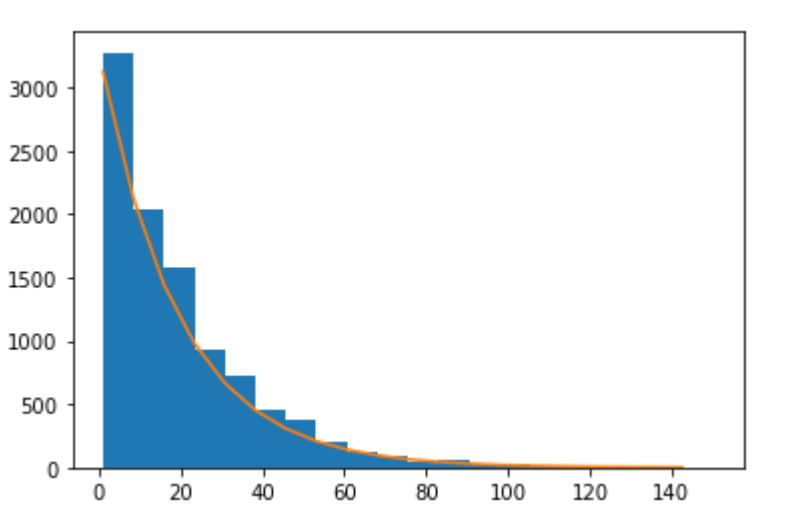
In [213…

```python
x1 = burgers3(20,10000)
n1, x1, _ = plt.hist(x1,20)
x= np.sort(x1)
y= -np.sort(-n1)
print(y)
x =x[0:20]
w,d =np.polyfit(x, np.log(y ,where=y>0),1)
y = np.exp(d) * np.exp(w * x)
plt.plot(x,y)
```

```
[3.279e+03 2.040e+03 1.580e+03 9.350e+02 7.260e+02 4.500e+02 3.840e+02
 2.000e+02 1.290e+02 9.000e+01 5.300e+01 4.600e+01 2.800e+01 2.300e+01
 1.200e+01 8.000e+00 6.000e+00 5.000e+00 4.000e+00 2.000e+00]
```

Out[213…

```
[<matplotlib.lines.Line2D at 0x1bcec5beee0>]
```



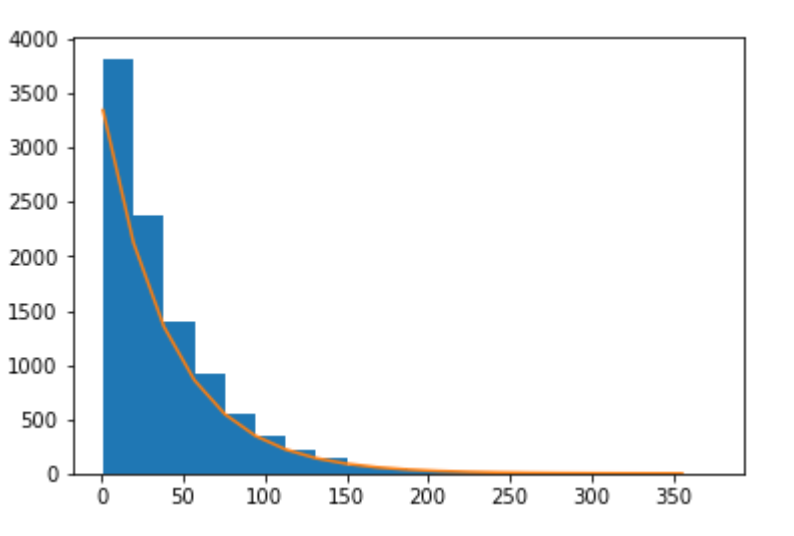
In [222…

```python
x1 = burgers3(40,10000)
n1, x1, _ = plt.hist(x1,20)
x= np.sort(x1)
y= -np.sort(-n1)
print(y)
x =x[0:20]
w2,d2 =np.polyfit(x, np.log(y ,where=y>0),1)
y = np.exp(d2) * np.exp(w2 * x)
plt.plot(x,y)
```

```
[3.822e+03 2.369e+03 1.393e+03 9.110e+02 5.590e+02 3.490e+02 2.240e+02
 1.400e+02 7.600e+01 4.700e+01 3.100e+01 2.000e+01 1.400e+01 1.200e+01
 4.000e+00 4.000e+00 3.000e+00 1.000e+00 1.000e+00 0.000e+00]
```

Out[222…

```
[<matplotlib.lines.Line2D at 0x1bced910e50>]
```



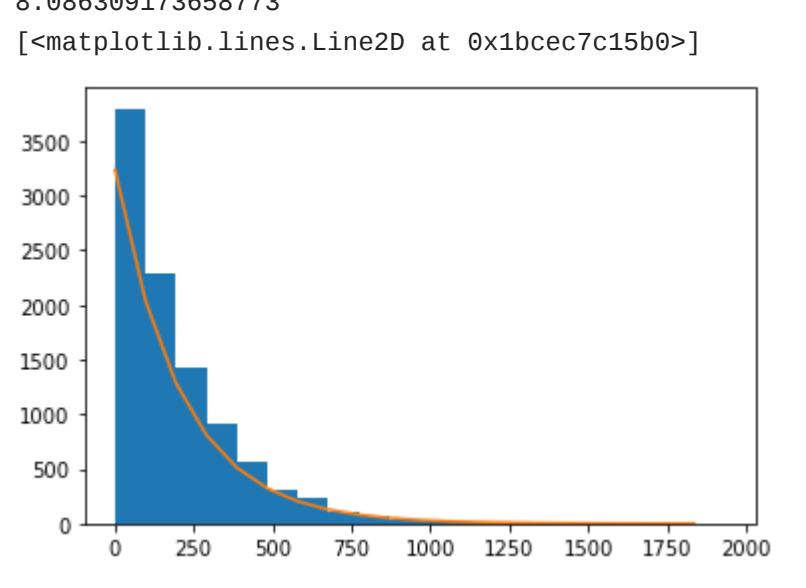
In [216…

```python
x1 = burgers3(200,10000)
n1, x1, _ = plt.hist(x1,20)
x= np.sort(x1)
y= -np.sort(-n1)
print(y)
x =x[0:20]
w3,d3 =np.polyfit(x, np.log(y ,where=y>0),1)
y = np.exp(d3) * np.exp(w3 * x)
plt.plot(x,y)
```

```
[3.795e+03 2.291e+03 1.434e+03 9.070e+02 5.680e+02 3.030e+02 2.260e+02
 1.100e+02 6.600e+01 3.500e+01 3.400e+01 2.100e+01 1.300e+01 5.000e+00
 4.000e+00 3.000e+00 2.000e+00 1.000e+00 1.000e+00 1.000e+00]
-0.0047822940006111425
8.086309173658773
```

Out[216…

```
[<matplotlib.lines.Line2D at 0x1bcec7c15b0>]
```



In [267…

```python
print(w),print(d) # fit parameters for the exponential curves
print(w2),print(d2)
print(w3),print(d3)
```

```
-0.05158807842658731
8.218700332015874
-0.023780143644115372
8.114327245169587
-0.004627059082988711
8.040437142588889
```

Out[267…

```
(None, None)
```

In [269…

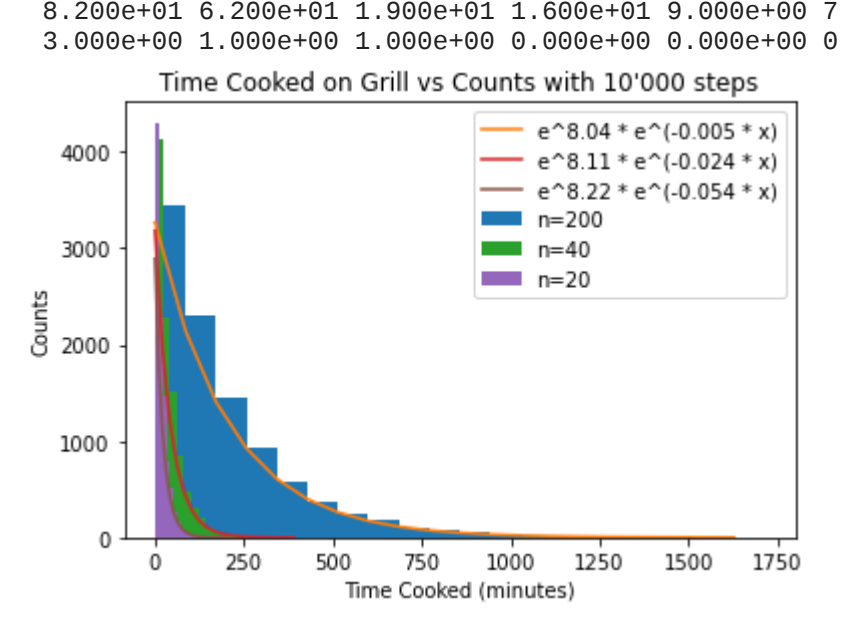
```python
x1 = burgers3(200,10000)
n1, x1, _ = plt.hist(x1,20)
x= np.sort(x1)
y= -np.sort(-n1)
print(y)
x =x[0:20]
w3,d3 =np.polyfit(x, np.log(y ,where=y>0),1)
y = np.exp(d3) * np.exp(w3 * x)
plt.plot(x,y)

x1 = burgers3(40,10000)
n1, x1, _ = plt.hist(x1,20)
x= np.sort(x1)
y= -np.sort(-n1)
print(y)
x =x[0:20]
w2,d2 =np.polyfit(x, np.log(y ,where=y>0),1)
y = np.exp(d2) * np.exp(w2 * x)
plt.plot(x,y)


x1 = burgers3(20,10000)
n1, x1, _ = plt.hist(x1,20)
x= np.sort(x1)
y= -np.sort(-n1)
print(y)
x =x[0:20]
w,d =np.polyfit(x, np.log(y ,where=y>0),1)
y = np.exp(d) * np.exp(w * x)
plt.plot(x,y)
plt.legend(["e^8.04 * e^(-0.005 * x)","e^8.11 * e^(-0.024 * x)", "e^8.22 * e^(-0.054 * x)", "n=200", "n=40","n=20"])

plt.xlabel("Time Cooked (minutes)")
plt.ylabel("Counts")
plt.title("Time Cooked on Grill vs Counts with 10'000 steps");
```

```
[3.440e+03 2.300e+03 1.454e+03 9.270e+02 5.790e+02 3.800e+02 2.350e+02
 1.780e+02 1.080e+02 8.200e+01 4.800e+01 3.000e+01 2.500e+01 1.100e+01
 7.000e+00 6.000e+00 4.000e+00 3.000e+00 2.000e+00 1.000e+00]
[4.134e+03 2.272e+03 1.504e+03 8.410e+02 4.750e+02 2.980e+02 2.010e+02
 1.030e+02 6.400e+01 3.900e+01 2.100e+01 1.100e+01 6.000e+00 5.000e+00
 4.000e+00 1.000e+00 1.000e+00 0.000e+00 0.000e+00 0.000e+00]
[4.302e+03 2.273e+03 1.468e+03 7.920e+02 5.250e+02 2.690e+02 1.670e+02
 8.200e+01 6.200e+01 1.900e+01 1.600e+01 9.000e+00 7.000e+00 4.000e+00
 3.000e+00 1.000e+00 1.000e+00 0.000e+00 0.000e+00 0.000e+00]
```



In [ ]: