Introduction to Statistical Distributions for Engineers with Python

Prepared by:

Dr. Gokhan Bingol (gbingol@pebytes.com)

September 04, 2023

Document version: 1.0

Updates will be available at: https://www.pebytes.com/pubs

Follow on YouTube: https://www.youtube.com/@sciencesuit

1. INTRODUCTION

We encounter with random variables (the life of a pump, the time it takes to complete a certain task, the occurrence of earthquakes, …) in everyday life and therefore they play an important role in many disciplines, including engineering (Forbes *et al*., 2011). The probability distribution for a random variable describes the distribution of the probabilities over the values of the random variable[1]. Wolfram MathWorld[2] lists a wealth of statistical distributions (discrete and continuous) and according to Forbes *et al*. (2011) only a relatively small number have come to prominence. This is in line with Bury's (1999) textbook which contains a small subset of the available distributions which were presumed to be used by scientists and engineers. If not all, the current work contains a fairly good number of the distributions frequently used by engineers.

With the advent of digitalization in the process industry, engineers need to be equipped not only with fundamental engineering knowledge, but with digital skills as well (Proctor & Chiang, 2023)[3]. Furthermore, with the rapidly rising popularity of machine learning techniques (and its superset artificial intelligence), engineers need the necessary mathematical skills for descriptive & inferential statistics and supervised/unsupervised modeling methods (Pinheiro & Patetta, 2021). Statistical distributions are among the pillars of all of the above-mentioned methods and therefore an understanding of the commonly used distributions is not only essential but also becoming vital for engineers.

The current work is an introduction to statistical distributions and readers who wish to take their learnings to the next level are recommended to refer to textbooks, especially on mathematical statistics. The target audience of the current work is engineers and therefore this document assumes the reader already has some background in statistics and calculus. Furthermore, at least a basic level of understanding of Python programming language is required.

This work heavily uses the following Python packages: *matplotlib*, *numpy* and *scisuit*[4]. The design of *scisuit*'s statistical library is inspired by *R*[5] and therefore the knowledge gained from the current work can be conveniently adapted to *R*, which is a popular software in data science realm.

1 Britannnica, https://www.britannica.com/science/statistics/Random-variables-and-probability-distributions
2 https://mathworld.wolfram.com/topics/StatisticalDistributions.html
3 https://www.thechemicalengineer.com/features/data-science-and-digitalisation-for-chemical-engineers/
4 **scisuit** at least v1.1.0. Unless otherwise stated, alternate names **np** and **plt** are used for **numpy** and **matplotlib**.
5 https://www.r-project.org/

1. FUNDAMENTALS

1.1. Permutations / Combinations

Any ordered sequence of k objects taken *without replacement* from a set of n objects is called a permutation of size k of the objects (Devore *et al*., 2021). There are two cases:

A) *Objects are Distinct:* The urn contains only distinct objects, such as A, B, C… Then the number of permutations of length k, that can be formed from a set of n elements is:

$$n P_k = n \cdot (n-1) \cdot \ldots \cdot (n-k+1) = \frac{n!}{(n-k)!} \tag{1.1}$$

Example 1.1

How many permutations of length k=3 can be formed from the elements A, B, C and D (Adapted from Larsen & Marx, 2011)?

Solution:

Mathematically the solution is: $\frac{4!}{(4-3)!} = 24$

```
from itertools import permutations

for p in permutations(["A", "B", "C", "D"], 3):
    print(p)
```

This will printout 24 tuples, each representing a permutation. ■

B) *Objects are NOT Distinct:* The urn contains n objects, n_1 being one kind, n_2 of second kind … and n_r of r^{th} kind, then:

$$\frac{n!}{n_1! \cdot n_2! \cdot n_r!} \tag{1.2}$$

where $n_1 + n_2 + \ldots + n_r = n$

Example 1.2

A biscuit in a vending machine cost 85 cents. In how many ways can a customer put 2 quarters, 3 dimes and 1 nickel (Adapted from Larsen & Marx, 2011)?

<u>*Solution*</u>:

$n_1=2$, $n_2=3$ and $n_3=1 \rightarrow n = n_1+n_2+n_3 = 2+3+1=6$.

Then we use Eq. (1.2):

$$\frac{6!}{2!3!1!}=60$$ ■

Combinations

The number of different combinations of *n* different things taken, *k* at a time, *without repetitions*, is denoted by (Kreyszig et al., 2011):

$$\binom{n}{k}=\frac{n!}{k!(n-k)!} \quad \textbf{(1.3)}$$

and if *repetitions allowed*:

$$\binom{n+k-1}{k} \quad \textbf{(1.4)}$$

Example 1.3:

Given a set of elements A, B, C and D list the combinations of size 2.

<u>*Solution:*</u>

There are 6 ways to list combinations of size 2: $\binom{4}{2}=\frac{4!}{2!(4-2)!}=6$

```
from itertools import combinations

for c in combinations(["A", "B", "C", "D"], 2):
   print(c)
```

Output: ('A', 'B'), ('A', 'C'), ('A', 'D'), ('B', 'C'), ('B', 'D'), ('C', 'D').

Note that each tuple contains *k=2* different “things” and none of the tuples contains exactly the same *2* things, i.e. there is no (‘A’, ‘A’). Please also note that unlike permutations, there is no ('B', 'A'), ('C', 'A') since there is alread (‘A’, ‘B’) and (‘A’, ‘C’), respectively.

Note that if *repetitions were allowed* Eq. (1.4) would be used:

$\binom{4+2-1}{2}=\binom{5}{2}=\frac{5\cdot 4}{2\cdot 1}=10$, adding (AA, BB, CC, DD) to the above list. ■

1.2. Random Variables

A random variable is a variable which can assume different values and whose value depends on the outcome of a chance experiment (Peck *et al*. 2016; Devore *et al*. 2021). For example, when two dice are tossed a sample space of a set of 36 ordered pairs, $S(i,j)=[(1,1),(1,6),\ldots,(6,1),(6,6)]$ is obtained. For some of the games only the sum of the numbers is of interest to us, therefore, we are only interested in eleven possible sums (2, 3,…, 11, 12), i.e. if we were interested in sum being 7, it does not matter if the outcome was (4, 3) or (6, 1). Therefore, in this case we have defined the random variable as $X(i,j)=i+j$ (Larsen & Marx, 2011).

There are two types of random variables:

1. *Discrete:* Takes values from either a finite set or a countably infinite set.
2. *Continuous:* Takes values from uncountably infinite number of outcomes, i.e. all numbers in a single interval on the number line.

1.2.1. Moment-Generating Function

Let X be a random variable. Then the moment-generating function for X is denoted by $M_x(t)$ and expressed as:

$$M_W(t)=E(t^{tW})=\begin{cases}\sum_{all\ k} e^{tk} p_W(k) & \text{if W is discrete}\\ \int_{-\infty}^{\infty} e^{tw} f_W(w) & \text{if W is continuous}\end{cases} \tag{1.5}$$

Theorem: Let $W_1, W_2, \ldots, W_n$ be independent random variables with mgfs $M_{W1}(t), M_{W2}(t), \ldots, M_{Wn}(t)$, respectively. Let $W = W_1+W_2+, \ldots, W_n$. Then,

$$M_W(t)=M_{W_1}(t)\cdot M_{W_2}(t)\ldots M_{W_n}(t) \tag{1.6}$$

Example 1.4:

Find the moment-generating function of a binomial random variable is given by the following equation:

$$p_x(k)=P(X=k)=\binom{n}{k}p^k(1-p)^{n-k}$$

<u>*Solution:*</u>

Binomial random variable is a discrete random variable, therefore $M_X(t)$ is:

$$M_X(t)=E(e^{tX})=\sum_{k=0}^{n}e^{tk}\binom{n}{k}p^k(1-p)^{n-k}$$

Rewriting the equation yields:

$$M_X(t)=\sum_{k=0}^{n}\binom{n}{k}(pe^t)^k(1-p)^{n-k}$$

Noting that Newton's binomial expansion formula:

$$(x+y)^n=\sum_{k=0}^{n}\binom{n}{k}x^k\cdot y^{n-k}$$

Observing that mgf and binomial expansion are exactly the same if we replace x and y with $x=pe^t$ and $y=1-p$. Therefore the moment-generating function is:

$$M_X(t)=(1-p+pe^t)^n$$ ■

1.2.2. Expected Value

It is the most frequently used statistical measure to describe central tendency (Larsen & Marx, 2011). Let X and Y be discrete and continuous random variables, respectively. The expected values of X and Y are denoted by E(X) and E(Y), respectively, and given by the following equations:

$$E(X)=\mu=\sum_{\text{all k}}k\cdot p_X(k) \tag{1.7}$$

$$E(Y)=\mu=\int_{-\infty}^{\infty}y f_Y(y)dy \tag{1.8}$$

One notable property of expected value is that it is a linear operator and therefore,

$$E(aX+bY)=a\cdot E(X)+b\cdot E(Y) \qquad \textbf{(1.9)}$$

Example 1.5:

Below table shows the number of courses a student registered in a university with 15,000 students. Find the average number of courses per student (Adapted from Carlton & Devore, 2014).

x	1	2	3	4	5	6	7
# Students	150	450	1950	3750	5850	2550	300

<u>Solution #1:</u>

The following is probably the approach most will take:

$$\bar{x}=\frac{1\times 150+2\times 450+...+7\times 300}{15000}=4.57$$

<u>Solution #2:</u>

We define a random variable X as the number of courses a student has enrolled. The mean value (weighted average) of a random variable is its expected value. Furthermore, since the random variable is discrete, Eq. (1.7) will be applied.

However, we first need to compute the probabilities:

p(x)	0.01 = 150/15000	0.03	0.13	0.25	0.39	0.17	0.02 = 300/15000

Eq. (1.7) can now be applied: $\bar{x}=1\times 0.01+2\times 0.03+...+7\times 0.02=4.57$ ■

Although Eqs. (1.7 & 1.8) can be used to find the expected value of a random variable, it is not always very convenient to do so.

If $M_W(t)$ is the moment-generating function (mgf) of the random variable W, then the following relationship holds as long as the r^{th} derivative of mgf exists:

$$M_W^{(r)}(0)=E(W^r) \qquad \textbf{(1.10)}$$

Let's prove for r=1.

$$M_Y^{(1)}(0)=\frac{d}{dt}\int_{-\infty}^{\infty} e^{ty} f_Y(y)\,dy$$

Placing the derivative as the integrand, then equation can be rewritten as:

$$M_Y^{(1)}(0)=\int_{-\infty}^{\infty} \frac{d}{dt} e^{ty} f_Y(y)\,dy$$

Noting that only e^{ty} is a function of t and performing the derivation yields:

$$M_Y^{(1)}(0)=\int_{-\infty}^{\infty} y\,e^{ty} f_Y(y)\,dy$$

Replacing t=0 gives:

$$M_Y^{(1)}(0)=\int_{-\infty}^{\infty} y\,f_Y(y)\,dy$$

Note that Eq. (IV) is exactly the same as Eq. (1.8), which is the expected value, *E(X)*. Therefore, the first-derivative of mgf with respect to *t=0* gives *E(X)* and second-derivative $E(X^2)$ and so on…

Example 1.6

Find the expected value of the binomial random variable.

Solution:

The moment-generating function was already computed in example (1.4) as:

$$M_X(t)=\left(1-p+pe^t\right)^n$$

Taking the derivative with respect to *t*:

$$M_X^{(1)}(t)=n\left(1-p+pe^t\right)^{n-1}\cdot pe^t$$

Replacing *t=0* yields the final answer:

$$M_X^1(t=0)=E(X)=np$$ ■

1.2.3. Variance

Although the expected value is an effective statistical measure of central tendency, it gives no information about the spreadout of a probability density function. Although the spread can be calculated using X-μ, it is immediately noted that negative deviations will cancel positive ones (Larsen & Marx, 2011). The variance of a random variable is defined as the expected value of its squared deviations. In mathematical terms,

$$Var(X)=E[(X-\mu)^2] \tag{1.11}$$

Noting the following property of expected value for the random variable X and $g(X)$ any function,

$$E[g(X)]=\sum_{\text{all k}} g(k)\cdot p_x(k) \tag{1.12}$$

If $g(X)$ in Eq. (1.12) is replaced with $(X\text{-}\mu)^2$, then Eq. (1.11) can also be expressed as,

$$Var(X)=\sum_{\text{all k}} (k-\mu)^2\cdot p_X(k) \tag{1.13}$$

If Y is a continuous random variable with probability density function $f_Y(y)$, then

$$Var(Y)=E[(Y-\mu)^2]=\int_{-\infty}^{\infty} (y-\mu)^2\cdot f_Y(y)dy \tag{1.14}$$

Let W be any random variable, discrete or continuous, and a and b any two constants. Then,

$$Var(a\cdot W+b)=a^2\cdot Var(W) \tag{1.15}$$

Let W_1, W_2, …., W_n be a set of independent random variables. Then,

$$Var(W_1+W_2+...+W_n)=Var(W_1)+Var(W_2)+...+Var(W_n) \tag{1.16}$$

Example 1.7

Test whether Eq. (1.11) represents population or sample variance.

<u>*Solution:*</u>

Let's work on an arbitrarily chosen dataset: [4, 7, 6, 2, 7, 6].

Spreadsheet's have two equations for computing sample and population variance, namely *Var.S* and *Var.P*, respectively. Computation with *Var.S* and *Var.P* yielded 3.86667 and 3.22222, respectively. Now let's inspect using Python tools:

```
import statistics as stat

x=np.array([4, 7, 6, 2, 7, 6]) #arbitrary numbers

#returns sample variance
varS1 = stat.variance(x.tolist())
varS2 = np.var(x, ddof=1) #notice ddof=1

print(f"Sample variance: statistics pkg= {varS1} and Numpy={varS2}")

#Using Equation
EX = np.mean(x)
EX2 = np.mean(x**2)

varEq = EX2 - EX**2
varP = np.var(x, ddof=0) #notice ddof=0

print(f"Population variance: Equation= {varEq} and Numpy={varP}")
```

***Sample** variance: statistics pkg= 3.8666 and Numpy=3.8666*

***Population** variance: Equation= 3.2222 and Numpy=3.2222*

Notice that the number of samples (x) was intentionally kept low to see the difference between sample and population variance since for large samples the difference becomes negligible. It is now clearly evident that Eq. (1.11) gives the population variance.

Although Eqs. (1.13 & 1.14) can be used to find variances of discrete and continuous random variables, respectively, using moment-generating function (if known/available) to find the variance can be more convenient as demonstrated in the following example.

Example 1.8

Find the variance of the binomial random variable.

Solution:

The moment-generating function was already computed in example (1.4) as: $M_X(t) = (1 - p + pe^t)^n$ and the expected-value was found in example (1.6) as: $E(X) = np$

From Eq. (1.10) we know that the second-derivative of mgf with respect to *t* gives *E(X*^{2}*)*, therefore:

$M_X^2(t) = pe^t \cdot n \cdot (n-1) \cdot (1 - p + pe^t)^{n-2}\, pe^t + n(1 - p + pe^t)^{n-1}\, pe^t$

Replacing *t=0* gives *E(X*^{2}*)*:

$E(X^2) = n(n-1)\, p^2 + np$

Putting expected value and last equation together and noting $E[(X - \mu)^2] = E(X^2) - E(X)^2$:

$Var(X) = n(n-1)\, p^2 + np - (np)^2 = np(1 - p)$ ■

2. DISCRETE PROBABILITY DISTRIBUTIONS

All discrete probability distributions has the following properties:

1. For every possible x value, $0 \leq x \leq 1$.
2. $\sum_{\text{all x values}} p(x) = 1$

As shown in the following figure, a discrete probability distribution can be visualized using a probability histogram.

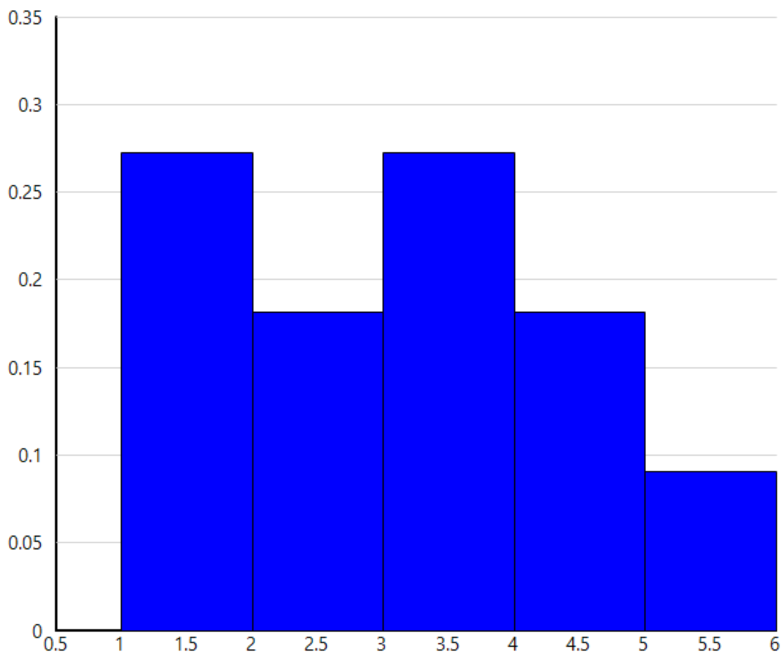


Fig 2.1: *Density histogram for a random data*

```
from scisuit.plot import histogram
from scisuit import App

app = App()

histogram([1, 2, 3, 4, 5, 3, 4, 2, 5, 4, 6])

app.mainloop()
```

Note that after the histogram has been plotted, the density option was selected and the number of bins were adjusted to 5.

2.1. Bernoulli Distribution

A Bernoulli trial can have one of the two outcomes, success or failure. The probability of success is p and therefore the probability of failure is *1-p* (Forbes *et al.*, 2011). It is the simplest discrete distribution; however, it serves as the building block for other complicated discrete distributions (Weisstein[6] 2023).

The probability density function is:

$$X_i = \begin{cases} 1 & p \\ 0 & 1-p \end{cases}, \quad 0 < p < 1 \tag{2.1}$$

MGF, Mean and Variance

$$M_W(t) = E(t^{tW}) = \sum_{all\,k} e^{tk} p_W(k)$$

$$M_W(t) = e^{t\cdot 0} \cdot p(X=0) + e^{t\cdot 1} p(X=1)$$

$$M_W(t) = (1-p) + p \cdot e^t \tag{2.2}$$

$$E(X) = p \tag{2.3}$$

$$Var(X) = E(X^2) - E(X)^2 = p - p^2 = p(1-p) \tag{2.4}$$

6 Weisstein, Eric W. "Bernoulli Distribution." From https://mathworld.wolfram.com/BernoulliDistribution.html

2.2. Binomial Distribution

The outcome of the experiment is either a *success* or a *failure*. The term *success* is determined by the random variable of interest (X). For example, if X counts the number of female births among the next n births, then a female birth can be considered as a *success* (Peck *et al.,* 2016).

We run n independent trials and define probability as $p = P(success\, occurs)$ and assume p remains ***constant*** from trial to trial (Larsen & Marx, 2011). We are only interested in the total number of *successes,* therefore we define X as the total number of successes in n trials. This definition then leads to binomial distribution and expressed as:

$$p_x(k) = P(X = k) = \binom{n}{k} p^k (1-p)^{n-k} \tag{2.5}$$

Imagine 3 coins being tossed, each having a probability of p of coming up heads. Then the probability of all heads (HHH) coming up is p^3 and all tails (no heads, TTT) is $(1\text{-}p)^3$ and HHT is $3p^2(1\text{-}p)$.

Observe that in Eq. (2.5) the combination part shows the number of ways to arrange k heads and n-k tails (section 1.1), therefore:

$$\binom{n}{k} = \frac{n!}{k!(n-k)!} \tag{2.6}$$

the remaining part of Eq. (2.5), $p^k \cdot (1-p)^{n-k}$, is the probability of any sequence having k heads and n-k tails.

Example 2.1:

An IT center uses 9 drives for storage. The probability that any of them is out of service is 0.06. For the center at least 7 of the drives must function properly. What is the probability that the computing center can get its work done (Adapted from Larsen & Marx, 2011)?

Solution #1:

$$\binom{9}{7} 0.94^7 0.06^2 + \binom{9}{8} 0.94^8 0.06^1 + \binom{9}{7} 0.94^9 0.06^0 = 0.986$$

```
>> sum( dbinom(x=[7, 8, 9], size=9, prob=0.94) )
0.986
```

Solution #2:

$$\binom{9}{7}0.94^7 0.06^2+\binom{9}{8}0.94^8 0.06^1+\binom{9}{7}0.94^9 0.06^0=1-\sum_{i=0}^{6}\binom{9}{i}0.94^i 0.06^{(9-i)}$$

```
>> 1 - pbinom(q=6, size=9, prob=0.94)
0.986
```

■

Example 2.2:

Find the 10% quantile of a binomial distribution with 10 trials and probability of success on each trial is 0.4?

```
>> qbinom(p=0.10, size=10, prob=0.4)
2.0
```

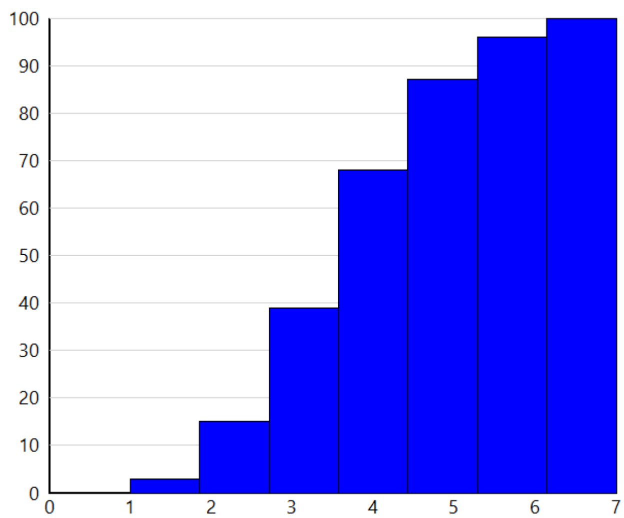


Figure shows the results of a simulation run by generating 100 random data points from the binomial distribution.

```
from scisuit.plot import histogram
from scisuit.stats import rbinom
>> data = rbinom(n=100, size=10, prob=0.4)
>> histogram(data)
```

Fig 2.2: *Cumulative histogram of 100 random data points*

It is seen that 10% quantile is somewhere around 1.8, less than 2; however, when reporting it is rounded up[7]. The following two commands shines more light on this policy:

7 https://www.boost.org/doc/libs/1_40_0/libs/math/doc/sf_and_dist/html/math_toolkit/policy/pol_tutorial/understand_dis_quant.html

```
>> pbinom(q=1, size=5, prob=0.3)
0.52822

>> pbinom(q=2, size=5, prob=0.3)
0.83692

>> qbinom(p=[0.53, 0.80], size=5, prob=0.3)
[2, 2]
```

It can be seen that although p=0.53 is closer to q=1 (0.52822) whereas p=0.80 is closer to q=2 (p=0.83692), any number in between p=0.52822 and p=0.83692 will be reported as q=2 by the *qbinom* function. ■

MGF, Mean and Variance

The derivations of MGF, *E(X)* and *Var(X)* were already presented in Examples (1.4), (1.6) and (1.8), respectively. Although approaches presented in the examples work very well, one can also keep in mind that each binomial trial is actually a Bernoulli trial, therefore the random variable *W* for binomial distribution is a function of Bernoulli random variables: $X_1, X_2, \ldots, X_n,$ *yielding* $W=X_1+X_2+\ldots+X_n$. Thus Eq. (2.2) and Eq. (1.6) can be combined to derive Eq. (2.7). Remembering the linearity of expected value, similar approach can be used for *E(W)* and *Var(W)* to obtain Eqs. (2.7 & 2.8).

$$M_X(t)=(1-p+pe^t)^n \qquad \textbf{(2.7)}$$

$$E(X)=np \qquad \textbf{(2.8)}$$

$$Var(X)=np(1-p) \qquad \textbf{(2.9)}$$

```
>> rbinom(n=10, size=5, prob=0.5)
[1, 2, 2, 1, 2, 3, 2, 2, 3, 2]
```

What do the numbers returned by the function mean?

In an analogy, we flip 5 coins (*size=5*) and count the number of heads (*prob=0.5*) which we consider as success. We run this experiment for 10 times (*n=10*). In the first experiment we have 1 heads, in the second 2 heads and so on.

Let's run a simple simulation to test Eq. (2.8):

```
from scisuit.stats import rbinom

N=1000
p = 0.3

for size in [5, 10]:
    x = rbinom(n=N, size=size, prob=p)
    print(f"size={size}, mean= {np.mean(x)}")
```

size=5, mean= 1.489

size=10, mean= 2.983

We have intentionally run large number of experiments (*N=1000*) for the simulation. Note that Eq. (2.8) and rbinom function match when *n=size* and *p=prob*. Therefore for the first case $E=5\times0.3=1.5$, which is close to 1.49.

To test Eq. (2.9) following simulation can be run:

```
N = 5000
p = 0.3
n=10

x = rbinom(n=N, size=n, prob=0.3)
print(f"variance = {np.var(x, ddof=0)}")
print(f"equation = {n*p*(1-p)}")
```

variance = 2.108

equation = 2.099

2.3. Hypergeometric Distribution

Suppose that an urn contains r good chips and w defective chips. Then the total number of chips (N) is $N=r+w$.

If n chips are drawn out at random *without replacement*, and X denotes the total number of good chips selected, then X has a hypergeometric distribution and,

$$P(X=k)=\frac{\binom{r}{k}\cdot\binom{w}{n-k}}{\binom{N}{n}} \qquad \textbf{(2.10)}$$

Notes:

1. If the selected chip was returned back to the population, that is the chips were drawn *with replacement*, then X would have a binomial distribution (see Example 2.3).
2. Since we are interested in total number of good chips, it does not matter if it is $r_1r_2r_3$…. or $r_2r_1r_3$…. Therefore instead of $\frac{r!}{(r-k)!}$ we used $\binom{r}{k}=\frac{r!}{k!\cdot(r-k)!}$.

Example 2.3:

An urn has 100 items, 70 good and 30 defective. A sample of 7 items is drawn. What is the probability that it has 3 good and 4 defective items? (adapted from Tesler 2017[8])

Solution #1: Sampling with replacement

$$p=\frac{70}{100}=0.7$$

$$p(X=3)=\binom{7}{3}\cdot 0.7^3\cdot(1-0.7)^4=0.0972$$

8 https://mathweb.ucsd.edu/~gptesler/186/slides/186_hypergeom_17-handout.pdf

Solution #2: Sampling without replacement

Number of samples with 3 good items: $\binom{70}{3}$

Number of samples with 4 bad items: $\binom{30}{4}$

Number of samples of size 7: $\binom{100}{7}$

$$P(3 \text{ good and } 4 \text{ bad}) = \frac{\binom{70}{3} \cdot \binom{30}{4}}{\binom{100}{7}} = 0.0937$$

```
#Solution 1
>> dbinom(x=3, size=7, prob=0.7)
0.09724


#Solution 2
>> dhyper(x=3, m=70, n=30, k=7)
0.093715
```

■

MGF, Mean and Variance

The mgf, mean and variance of hypergeometric distribution are presented by Walck (2007) and derivation of expected-value is given by Hogg *et al*. (2019).

$$M_X(t) = \frac{\binom{W}{n}}{\binom{N}{n}} \cdot {}_2F_1(-n, -r; w-n+1; t) \qquad \textbf{(2.11)}$$

where F is hypergeometric function.

Let $p = \frac{r}{N}$ and $q = 1 - p$ then,

$$E(X) = np \tag{2.12}$$

$$Var(X) = npq\frac{N-n}{N-1} \tag{2.13}$$

Let’s test Eq. (2.12) using Python code:

```
>> N=1000 #number of experiments
>> x=st.rhyper(nn=1000, m=70, n=30, k=7)


>> np.mean(x)
4.937
```

Explicitly expressing Eq. (2.12):

$$E(X) = np = n \cdot \frac{r}{N}$$

Transforming above equation to the notation used by the rhyper function:

$$E(X) = k \cdot \frac{m}{m+n} = 7 \cdot \frac{70}{30+70} = 4.9$$

2.4. Geometric Distribution

It is similar to binomial distribution such that trials have two possible outcomes: success or failure. However, unlike binomial distribution where we were interested in the total number of successes, now we are only interested in the trial where *first success occurs*. Therefore, if *k* trials were carried out, *k-1* trials end up in failures and the k^{th} one occurs with success. Thus we define the random variable *X* as the trial at which the first success occurs (Larsen & Marx, 2011).

In more explicit terms, we have thus far said that: "first *k-1* trials end up in failure" ***and*** "k^{th} trial ends in success". Mathematically expressing,

$$P(X=k) = P(\text{first success on } k^{th} \text{ trial}) = P(\text{first } k\text{-}1 \text{ ends in failure}) \cdot P(k^{th} \text{ trial ends in success}) \quad \mathbf{(2.14)}$$

which then leads to the following equation:

$$P(X=k)=(1-p)^{k-1}\cdot p \quad \mathbf{(2.15)}$$

MGF, Mean and Variance

$$M_x(t)=\frac{pe^t}{1-(1-p)e^t} \quad \mathbf{(2.16)}$$

$$E(X)=\frac{1}{p} \quad \mathbf{(2.17)}$$

$$var(X)=E(X^2)-E(X)^2=\frac{1-p}{p^2} \quad \mathbf{(2.18)}$$

Example 2.4

A political pollster randomly selects persons on the street until he encounters someone who voted for the political party, Fun-Party. What is the probability he encounters 3 people who did not vote for the Fun-Party before he encounters one who voted. It is known that 20% of the population voted for the Fun-Party (adapted from Foley[9] 2019)?

---

9 https://rpubs.com/mpfoley73/458721

Solution:

The probability of success (voted for Fun-Party) is: $p=\frac{20}{100}=0.2$

Since 3 have not voted for the Fun-Party (failure) and the next one voted, 4 trials carried out.

$P(X=4)=(1-0.2)^3 \cdot 0.2^1=0.1024$

Using Python code:

```
>> dgeom(x=3, prob=0.2)
0.1024
```

Note that, in the definition of the function dgeom *x* is the number of failures, therefore, instead of x=4, x=3 was used. ■

2.5. Negative Binomial Distribution

In section (2.4) the geometric distribution was introduced where we defined the random variable X as the trial at which the *first* success occurs. Therefore the trials were discontinued as soon as a success occurred. Now instead of first success, we are interested in r^{th} success. Similar to geometric distribution each trial has a probability p of ending in success.

Therefore, we might have a sequence of {S, F, F, S, S, S} if we were interested in the r=4$^{\text{th}}$ success out of $k=6$ trials. Putting it in more mathematical terms,

3 successes before the 4$^{\text{th}}$ success: r-1

2 failures before the 4$^{\text{th}}$ success out of k-1=5 trials: (k-1) - (r-1)

Now if we define the random variable X as the trial at which the r^{th} success occurs, then all the background work to obtain the probability density function (pdf) has been done.

Before proceeding with the final pdf, also note that before the r^{th} success occurs, k-1 trials might have various different sequences having r-1 successes, such as {SFFSS} or {FSFSS} or so on… Note that this is indeed very similar to the idea presented in section (2.2) by Eq. (2.6). Therefore,

I) Before the r^{th} success occurs different sequences with r-1 successes: $\binom{k-1}{r-1}$

II) (r-1 success in the first k-1 trials) and (success on kth trial): $p^{r-1}(1-p)^{k-1-(r-1)}$

Putting the equations in (I) and (II) together gives the pdf for negative binomial distribution:

$$p_X(k)=\binom{k-1}{r-1}p^r\cdot(1-p)^{k-r} \qquad \textbf{(2.19)}$$

2.5.1. Relationship to Geometric Distribution

Let G and B be random variables for geometric and negative-binomial distributions. The definitions of random variables are then as follows:

G: Trial at which the *first* success occurs

B: Trial at which the r^{th} success occurs

It is clearly seen that if $r=1$ then $B=G$, it can therefore be said that the negative-binomial distribution generalizes the geometric distribution. Larsen & Marx (2011) expresses the relationship between negative-binomial and geometric distributions in the following way which is easier derive a mathematical relationship between the random variables:

X = total number of trials to achieve *rth* success

= number of trials to achieve 1st success +

number of additional trials to achieve 2nd + … +

number of additional trials to achieve *rth* success.

$$X = X_1 + X_2 + \ldots + X_r \qquad \textbf{(2.20)}$$

where X_1, X_2,…, X_r are random variables for geometric distributions.

It should be observed that until the 1st success occurs the trials overlaps with the definition of geometric random variable. However, after the 1st success we are interested in the additional trials (the word additional is important) to observe the 2nd success and therefore the trials *between the 1st and 2nd success* fits again with the definition of geometric random variable. Continuing in this fashion the rationale for Eq. (2.20) is justified.

Example 2.5

A process engineer wishes to recruit 4 interns to aid in carrying out lab tests for the development of a new technology. Let p= P(randomly chosen CV is a fit). If p is 0.2, what is the probability that exactly 15 CVs must be examined before 4 interns can be recruited (Adapted from Carlton & Devore, 2014)?

<u>*Solution:*</u>

The pdf for negative-binomial distribution is:

$p_X(k) = \binom{k-1}{r-1} p^r \cdot (1-p)^{k-r}$ where $k=15$, $r=4$ and $p=0.2$.

Substituting *k* and *r* in the equation:

$$p(X=15)=\binom{15-1}{4-1}0.2^4\cdot(1-0.2)^{15-4}=0.050$$

Using Python:

```
>> dnbinom(x=15-4, size=4, prob=0.2)
0.050
```

Note that in dnbinom function the argument *x* represents the number of failures (*k*-*r*). ■

2.5.2. MGF, Mean and Variance

$$M_X(t)=\left[\frac{pe^t}{1-(1-p)e^t}\right]^r \quad \textbf{(2.21)}$$

$$E(X)=\frac{r}{p} \quad \textbf{(2.22)}$$

$$var(X)=\frac{r(1-p)}{p^2} \quad \textbf{(2.23)}$$

Although MGF, *E(X)* and *Var(X)* can be derived using Eq. (2.19) it should be noted that Eq. (2.20) paves the way to combine Eqs. (1.6 & 2.16) to derive MGF in a very straightforward way. Also using Eqs. (2.17 & 2.18) expected-value and variance can be derived conveniently as shown below:

1) $M_X(t)=M_{X_1}(t)\cdot M_{X_2}(t)\cdot \ldots . M_{X_r}(t)=\left[\frac{pe^t}{1-(1-p)e^t}\right]^r$

2) $E(X)=E(X_1)+E(X_2)+\ldots+E(X_r)=1/p+1/p+\ldots+1/p=r/p$

3) $var(X)=var(X_1)+var(X_2)+\ldots+var(X_r)=\frac{1-p}{p^2}+\frac{1-p}{p^2}+\ldots+\frac{1-p}{p^2}=\frac{r(1-p)}{p^2}$

2.6. Poisson Distribution

Poisson distribution is a consequence of Poisson limit, which is an approximation to binomial distribution when $n \to \infty$ and $p \to 0$.

2.6.1. Poisson Limit

The Poisson limit states that, if $n \to \infty$ and $p \to 0$ such that $\lambda = np$ remains constant, then for $k \geq 0$, the following relationship holds (Larsen & Marx, 2011):

$$\lim_{n \to \infty} \binom{n}{k} p^k (1-p)^{n-k} = \frac{e^{-np}(np)^k}{k!} \qquad \textbf{(2.24)}$$

A proof of Eq. (2.24) is presented in various textbooks (Devore *et al*., 2021; Larsen & Marx, 2011).

Let's inspect the accuracy of Eq. (2.24) using Python code. There are two tests where each has different probabilities (p); however for both tests $\lambda = np$ remains constant as 1.

Test #1

```
n, kmax = 5, 5
```

Test #2

```
n, kmax = 100, 10
```

Rest of the code for both tests:

```
p = 1/n #probability
binom = dbinom(x=x, size=n, prob=p)
pois = dpois(x=x, mu=n*p) #lambda=1

D = np.abs(np.array(binom)-np.array(pois)) #difference

print(f"min:{min(D)} at k={np.argmin(D)}")
print(f"max:{max(D)} at k={np.argmax(D)}")
```

Test #1: min:0.0027 at k=5 & max:0.0417 at k=1,

Test #2: min:3.13e-08 at k=10, max:0.0018 at k=1

It is clearly seen that in both tests the Poisson limit approximates binomial probabilities fairly well. However, as evidenced from Test #2 where n was larger and p was smaller, the agreement between Poisson limit and binomial probabilities became remarkably good for all k.

Example 2.6

When data is transmitted over a data link, there is a possibility of errors being introduced. Bit error rate is defined as the rate (errors/total number of bits) at which errors occur in a transmission system[10]. Assume you have a 4 MBit modem with bit error probability 10^{-8}. What is the probability of exactly 3 bit errors in the next minute (adapted from Devore *et al.* 2021)?

<u>*Solution:*</u>

In a minute $4\cdot 10^6 \dfrac{bits}{second} \times 60\, second = 240\cdot 10^6$ bits will be transferred and probability of error is 10^{-8}.

The errors can be at any sequence and we are interested in total number of errors, which is by definition is the binomial probability: $P(3) = \binom{240\cdot 10^6}{3} (10^{-8})^3 (1-10^{-8})^{240\cdot 10^6 - 3}$

Since n is very large (240,000,000) and p is very small (10^{-8}) the above computation is an excellent candidate for Poisson's limit: $\lambda = np = 2.4\cdot 10^8 \times 10^{-8} = 2.4$

```
#Binomial probability
>> dbinom(x=3, size=240000000, prob=1E-8)
0.2090142


#Poisson limit
>> dpois(x=3, mu=2.4)
0.2090142
```

If we pose the question, "what is the probability at most 3 bit errors in the next minute?", then the solution is:

$$P(X \leq 3) = \sum_{k=0}^{3} \binom{240\cdot 10^6}{k} (10^{-8})^k (1-10^{-8})^{240\cdot 10^6 - k}$$

10 https://www.electronics-notes.com/articles/radio/bit-error-rate-ber/what-is-ber-definition-tutorial.php

```
#binomial probability
pbinom(q=3, size=240000000, prob=1E-8)
0.7787229


#Poisson limit
>> ppois(q=3, mu=2.4)
0.7787229
```

■

2.6.2. Poisson Distribution

The random variable X is said to have a Poisson distribution if,

$$P_X(k)=\frac{e^{-\lambda}\lambda^k}{k!} \tag{2.25}$$

where λ>0.

Example 2.7

7 cards drawn (with replacement) from a deck containing numbers from 1 to 10. Success is considered when 5 is drawn. Can the produced data be described by the Poisson distribution?

Solution: Simulation will be run using the following script:

```
N=7
k = np.arange(0, N+1)
X = np.array(rbinom(n=10000, size=N, prob=0.1), dtype=np.int32)

unique, Frequencies = np.unique(X, return_counts=True)

total=np.sum(Frequencies)
aver = sum(Frequencies*unique)/total

probabilities = Frequencies/total
poisson = [dpois(x=i, mu=aver) for i in unique]

print(probabilities)
print(poisson)
```

[0.4781 0.3733 0.1253 0.0209 0.0021 0.0003]

[0.4983, 0.34708, 0.12087, 0.02806, 0.0048, 0.0007]

It is seen from the output that the probabilities can be well described by Poisson distribution. It should be noted that when the probability value in the simulation was increased to 0.5, the difference between actual and predicted probabilities increased. ■

2.6.3. MGF, Mean and Variance

$$M_X(t)=e^{\lambda\cdot(e^t-1)} \quad \textbf{(2.26)}$$

$$E(X)=\lambda \quad \textbf{(2.27)}$$

$$Var(X)=\lambda \quad \textbf{(2.28)}$$

Derivation of Eq. (2.26) can be found in mathematical statistic textbooks (Devore *et al*. 2021; Wackerly *et al*. 2008).

2.6.4. Poisson Process

It is widely used counting processes (the number of accidents in an area; the outbreaks of diseases, …) and mostly used in situations where we *only* know the rate of occurrence of an event but the events occur completely at random, for example using historic data knowing that earthquakes occurring in a certain area with a rate of 3 per year. Note that we only know the rate of earthquakes and do not have any information on timings of the earthquakes as they occur completely at random (Anon[11]. 2023). If an event satisfies the above-mentioned conditions we can assume that Poisson process might be a good candidate to model such event.

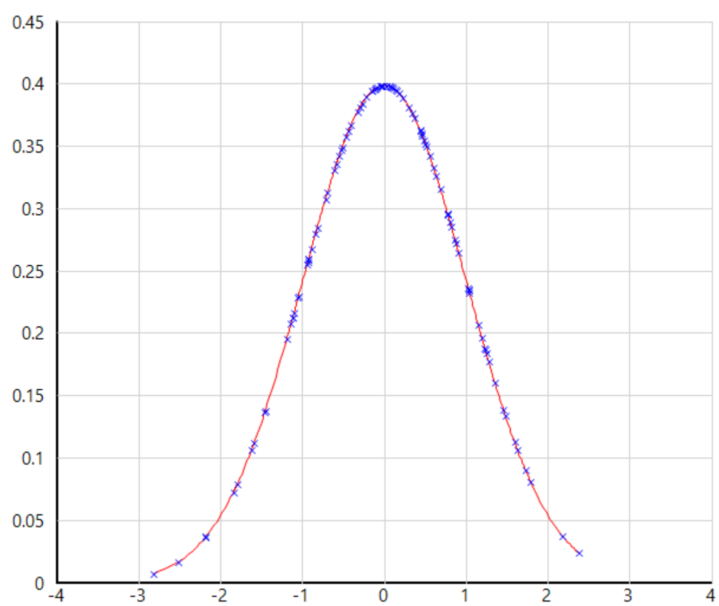
11 https://www.probabilitycourse.com/chapter11/11_1_2_basic_concepts_of_the_poisson_process.php

3. CONTINUOUS PROBABILITY DISTRIBUTIONS

Continuous probability distributions have the following properties:

1. $f(x) \geq 0,$
2. $\int_{-\infty}^{\infty} f(x) = 1$

Continuous probability distributions can be visualized by a curve called a density curve. The function that defines this curve is called the density function.




```
from scisuit.plot import scatter
from scisuit.stats import dnorm

>> x = np.random.standard_normal(100)
>> x.sort()
>> y = dnorm(x)

>> scatter(x=x, y=y)
```

Fig 3.1: *Density curve of standard normal distribution*

<u>*Notes:*</u>

1. Using the following rationale in the above-given script, probability density curve for other distributions can be obtained.
2. Although we sampled *x* variable from standard distribution and then computed the corresponding densities (*y*) using the dnorm function, one could also have used Numpy's linspace function as `x=np.linspace(start=-3, stop=3, num=100)`. In fact, the latter approach is more efficient and used throughout the document.

3.1. Uniform Distribution

If you generate random numbers between 0 and 1 using a computer, you will get observations from a uniform distribution since there will be *almost* same amount of numbers in each equally spaced sub-interval, i.e. 0-0.2 or 0.2-0.4. Let's run a simulation:

```
import random

x = np.array([random.random() for i in range(1000)])

start, dx=0.0, 0.2

while start<1.0:
    L = len( np.where( np.logical_and(x>=start, x<(start+dx)) )[0] )
    print(f"({start}, {start+dx}): {L}")

    start += dx
```

The output[12] is: *(0.0, 0.2): 218*, *(0.2, 0.4): 202*, *(0.4, 0.6): 192*, *(0.6, 0.8): 209*, *(0.8, 1.0): 179*

Although number of samples drawn was relatively small, it is seen that each sub-interval in the range of [0, 1] has similar amount of numbers. Instead of 1000 samples, if the simulation was run with 10,000,000 samples the difference between amount of numbers in each sub-interval would have been negligible.

A random variable *Y* has a continuous uniform probability distribution on the interval (a, b) if the probability density function is defined as follows:

$$f(y)=\begin{cases} \dfrac{1}{b-a} & a \le y \le b \\ 0 & elsewhere \end{cases} \tag{3.1}$$

The uniform distribution is very important for theoretical studies (Wackerly *et al*. 2008). For example if *F(y)* is a distribution function, it is often possible to transform uniform distribution to *F(y)*. For example, it is possible to transform uniform distribution to standard normal distribution using Box-Muller transform[13].

12 It should be reminded that in random sampling each run will produce different results.
13 https://en.wikipedia.org/wiki/Box%E2%80%93Muller_transform

MGF, Mean and Variance

For $t \neq 0$:

$$M_Y(t) = \int_{-\infty}^{a} 0 \cdot e^{ty}\, dy + \int_{a}^{b} \frac{e^{ty}}{b-a} + \int_{b}^{\infty} 0 \cdot e^{ty}\, dy$$

$$M_Y(t) = \frac{e^{tb} - e^{ta}}{t(b-a)}$$

and for $t=0$:

$$M_Y(t) = \int_{-\infty}^{\infty} e^{ty} \cdot f_Y(y)\, dy$$

$$M_Y(t) = \int_{-\infty}^{\infty} 1 \cdot \frac{1}{b-a}\, dy = 1$$

Therefore moment-generating function is:

$$f(y) = \begin{cases} \dfrac{e^{tb} - e^{ta}}{t(b-a)} & t \neq 0 \\ 1 & t = 0 \end{cases} \quad \textbf{(3.2)}$$

$$E(Y) = \frac{a+b}{2} \quad \textbf{(3.3)}$$

$$Var(Y) = \frac{(b-a)^2}{12} \quad \textbf{(3.4)}$$

It should be noted that the derivation (presented by Wolfram[14]) of Eq. (3.3) from Eq. (3.2) might pose challenges for many. Instead, it is recommended to use Eq. (1.8) as then the derivation becomes considerably more convenient.

14 https://mathworld.wolfram.com/UniformDistribution.html

Example 3.1

As evidenced from above a random number generator will spread its output uniformly across the entire interval from 0 to 1. What is the probability that the numbers will be in between 0.3 and 0.7?

<u>*Solution*</u>

This is a rather straightforward question and the answer is $P(0.3 \leq X \leq 0.7)=0.4$.

Let's demonstrate it with a short script:

```
import random

N = np.array( [10**i for i in range(1, 6)] )

x = list  ( map( lambda x: [random.random() for i in range(x)], N ) ) #2D list

#i is a 1D Python list
L = np.array( [len( np.where( np.logical_and(np.array(i)>=0.3, np.array(i)<0.7) )[0] ) for i in x] )

print(L/N)
```

The output is: *[0.3   0.41   0.396   0.4022   0.39924]*

As evidenced from the above output, as the number of samples (N) increased from 10 to 10^5, the simulated probability approached to the computed probability. ■

3.2. Normal Distribution

Normal distributions are bell-shaped and symmetric curves. They are widely used and are the single most important probability model in all of statistics since:

1. They provide a reasonable approximation to the distribution of many different variables,
2. They play a central role in many of the inferential procedures (Larsen & Marx, 2011; Peck *et al.*, 2016).

In section (2.6) it was shown that the Poisson limit approximated binomial probabilities when $n \rightarrow \infty$ and $p \rightarrow 0$. Historically, this was not the only approximation [interested reader can find a historical evolution of the normal distribution in the paper from Stahl (2006)]. Abraham DeMoivre showed that when X is a binomial random variable and n is large the probability for $P\left(a \leq \frac{X-np}{\sqrt{np(1-p)}} \leq b\right)$ can be estimated using the following equation:

$$f_Z(z)=\frac{1}{\sqrt{2\pi}} e^{\frac{-z^2}{2}}, -\infty < z < \infty \tag{3.5}$$

The formal statement of the approximation is known as DeMoivre-Laplace limit theorem (Larsen & Marx, 2011):

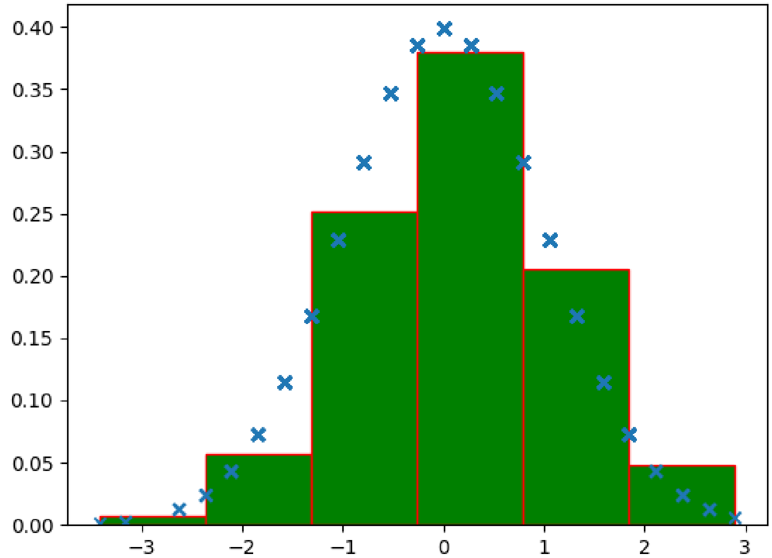
$$\lim_{n \rightarrow \infty} P\left(a \leq \frac{X-np}{\sqrt{np(1-p)}} \leq b\right)=\frac{1}{\sqrt{2\pi}} \int_a^b e^{-z^2/2} dz \tag{3.6}$$

Eq. (3.5) is referred as the *standard normal curve* where μ=0 and σ=1. If μ≠0 and σ≠1 then the equation is expressed as follows:

$$f_Z(z)=\frac{1}{\sqrt{2\pi\sigma}} e^{\frac{-1}{2} \cdot \left(\frac{x-\mu}{\sigma}\right)^2}, -\infty < x < \infty \tag{3.7}$$

In order to show DeMoivre's idea, let's write a fairly short Python script. rbinom function was used to sample 1000 experiments where each experiment consists of 60 trials with a probability of success of 0.4 (adapted from Larsen & Marx, 2011).

```
import math

n=60
p=0.4

#Generate random numbers from a binomial distribution
x = np.array(rbinom(n=1000, size=n, prob=p), dtype=np.float32)

#z-ratio
z = (x - n*p)/math.sqrt(n*p*(1-p))

#DeMoivre's equation
f = 1.0/math.sqrt(2*math.pi)*np.exp(-z**2/2.0)

#Density scaled histogram
plt.hist(z, bins=6, density=True, color = "green", ec="red")

#Overlay scatter plot
plt.scatter(x=z, y=f, marker="x")

plt.show()
```



It is seen that the curve generated by DeMoivre's approximation equation is fairly well describing the variation of the histogram generated by the binomial data.

Fig 3.2: *Density scaled histogram and scatter plot (x-axis: z-ratio, y-axis: density)*

3.2.1. MGF, Mean and Variance

$$M_Y(t) = e^{\mu t + \sigma^2 t^2/2} \tag{3.8}$$

$$E(Y) = \mu \tag{3.9}$$

$$Var(Y) = \sigma^2 \tag{3.10}$$

3.2.2. Sampling Variability

When we would like to estimate the mean value of a population, we would take samples of size n from the population and try to make inferences based on the sample. It is natural that the average value of samples will change from sample to sample. This is known as *sampling variability*.

In order to simulate this we will generate a sample space of size 250 from an exponential distribution. Then we will draw samples of size 5, 10, 20 and 30 (250 times) from the sample space and compute the average of each sample. It will reveal us how the choice of sample size affects sampling distribution.

We will run the following script:

```
#sample space size
N = 250

#Sample space generated by random numbers from an exponential distribution
SS = np.array(rexp(n=N))

figure, axis = plt.subplots(nrows=3, ncols=2)

#Density scaled histogram of exponential distribution
axis[0,0].hist(SS, color = "green", ec="red")
axis[0, 0].set_title(f"aver={round(np.mean(SS), 2)}")

#sample sizes
n = [5, 10, 20, 30]
colors=["blue", "orange", "green", "brown"]

row, col = 1, 0
```

```
for i in range(len(n)):

    #Sample 5, 10, ... samples from sample space and find average (repeat N times)
    x = np.array([np.mean(np.random.choice(SS, size = n[i], replace=False)) for _ in range(N)])

    axis[row, col].hist(x, color = colors[i], ec="red")
    axis[row, col].set_title(f"{chr(65+i)}) n={n[i]} - aver={round(np.mean(x), 2)}")

    col += 1
    if col%2 == 0:
        row += 1
        col = 0

plt.tight_layout()
plt.show()
```

This will produce an output similar to the following:

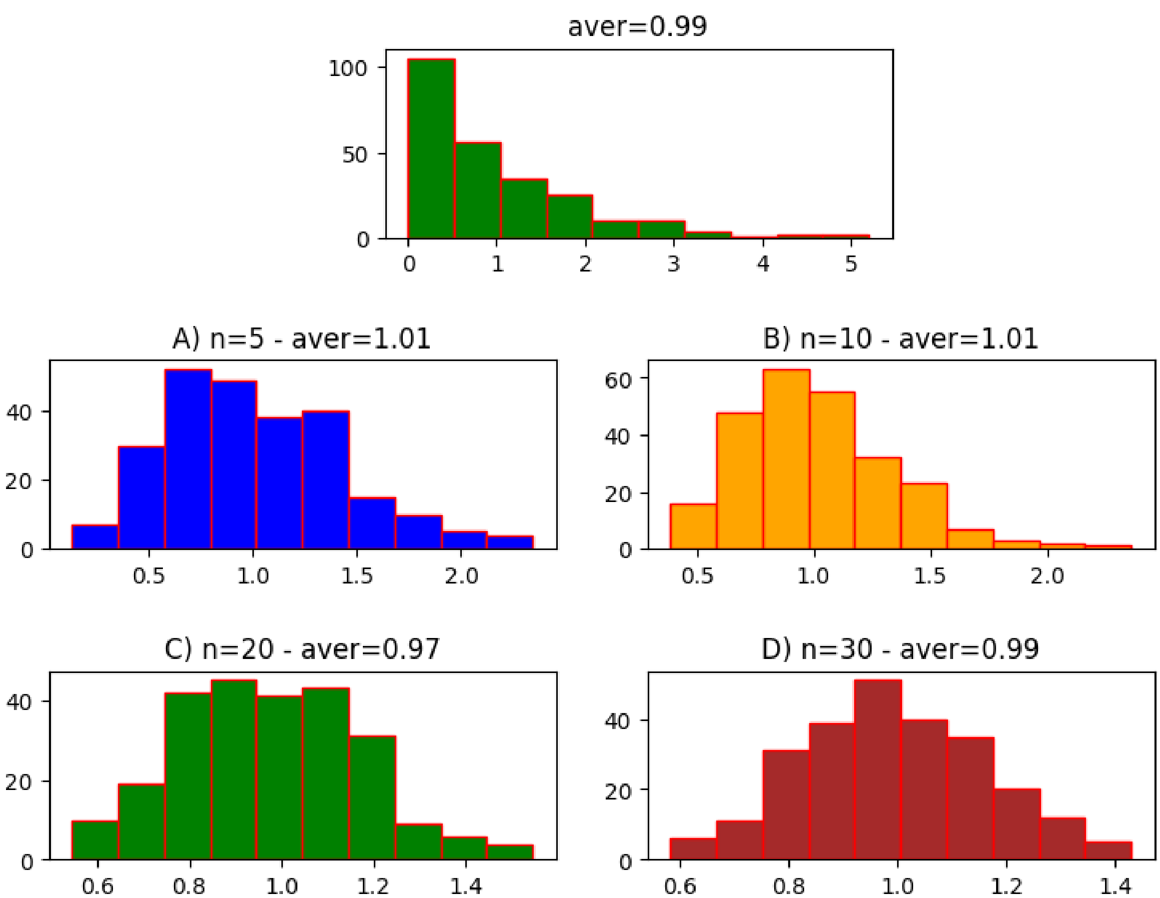


Fig 3.3: *Frequency histogram of sample space and different sample sizes*

The following inferences can be made from Fig. (3.3):

1. Although the histogram of sample space (variable *SS*) does not look like normal in shape, each of the four histograms is resembles to normal in shape,
2. Each of the histogram (*A*-*D*) has an average value close to the sample space's average value. Generally, $\bar{x}$ based on a larger sample size is closer to the mean value of the population.
3. The smaller the value of sample size, the greater the sampling distribution spreads out (compare the limits of x-axis for *A* and *D* where sample sizes were 5 and 30, respectively).

3.2.3. Central Limit Theorem

When n is sufficiently large (n≥30), the sampling distribution of $\bar{x}$ is well approximated by a normal curve (Peck *et al*., 2016). Formally expressing, let W_1, W_2, … be an infinite sequence of independent random variables each with the same distribution. Then,

$$\lim_{n\to\infty} P\left(a \le \frac{W_1+\ldots+W_n-n\mu}{\sqrt{n}\,\sigma} \le b\right) = \frac{1}{\sqrt{2\pi}}\int_a^b e^{-z^2/2}\,dz \qquad \textbf{(3.11)}$$

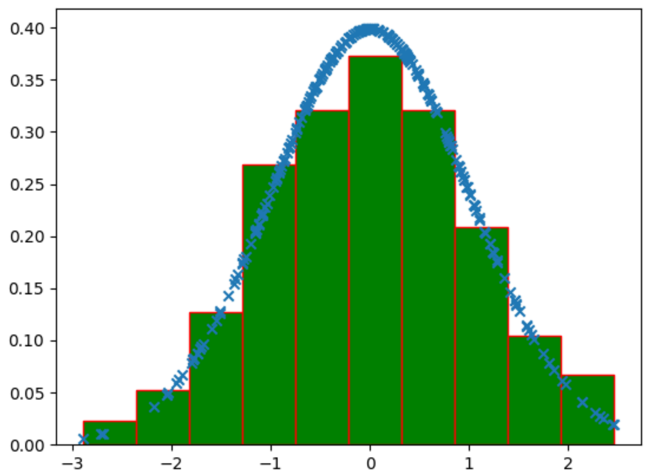
$$E\left[\frac{1}{n}(W_1+\ldots+W_n)\right] = E(\bar{W}) = \mu \qquad \textbf{(3.12)}$$

$$Var\left[\frac{1}{n}(W_1+\ldots+W_n)\right] = \frac{\sigma^2}{n} \qquad \textbf{(3.13)}$$

The implication of Eq. (3.13) could be observed from Fig. (3.3) where increasing the sample size decreased the variability of the distribution. In order to show how Eq. (3.11) works we will be generating an array with 5 columns and 250 rows from a standard uniform distribution. Then, sum of 5 columns will be computed to generate another array (250 rows). Since for a standard uniform distribution $\mu=0.5$ and $\sigma^2=1/12$, z-ratio will be computed using $\frac{y-5/2}{\sqrt{5/12}}$.

```
import math

n = 5


#Generate 250 random numbers from uniform dist
W = np.array(list( map( lambda _: runif(n=250) , [None]*n) )).transpose() #2D array (250*5)

#For uniform distribution
mu = 0.5
sigma = math.sqrt(1/12)

#W1+W2+...
x = np.sum(W, axis=1) #len=250

#z-ratio
z = (x - n*mu)/(math.sqrt(n)*sigma)

#DeMoivre's equation
f = 1.0 / math.sqrt(2*math.pi)*np.exp(-z**2/2.0)

#Density scaled histogram
plt.hist(z, density=True, color = "green", ec="red")

#Overlay scatter plot
plt.scatter(x=z, y=f, marker="x")

plt.show()
```

The output will be similar to the following figure:



It is seen that even the number of samples were small (n=5), the sums yielded a distribution closely resembling to normal distribution.

Larsen & Marx (2011) states that samples from symmetric distributions will produce sums that will quickly converge to the theoretical limit (normal dist). However, if samples come from a skewed distribution then larger *n* is needed (see section on *sampling variability*)

Fig 3.4: *Density scaled histogram and scatter plot (x-axis: z-ratio, y-axis: density)*

3.2.4. The 68-95-99.7 Rule

A normal distribution with mean μ and standard deviation σ:

1. Approximately 68% of the observations fall within σ of the mean μ.
2. Approximately 95% of the observations fall within 2σ of μ.
3. Approximately 99.7% of the observations fall within 3σ of μ.

The following short script demonstrates this:

```
N = 10000 #number of samples

#sample from standard normal distribution
x = np.array(rnorm(n=N))

for i in [1, 2, 3]:
    L = len(np.where(np.logical_and(x>=-i, x<=i))[0])

    print(f"{i} sigma= {L/N*100}%")
```

1 sigma= 68.61%, 2 sigma= 95.42%, 3 sigma= 99.75%

Note that *rnorm(n=,)* function samples from standard normal distribution where μ=0 and σ=1.

Example 3.2

A producer claims that bottles contain μ=12 deciliters of soda with σ=0.16 deciliters. To verify this claim as a quality control engineer you have randomly selected 16 bottles and measured the volume in each bottle. What is the probability that the average value of 16 bottles is in between 11.96 and 12.08 deciliters (adapted from Peck *et al*., 2016)?

<u>*Solution:*</u>

It is reasonable to assume that the samples come from a normal distribution.

Standard deviation of sample: $\sigma_{\bar{x}}=\frac{\sigma}{n}=\frac{0.16}{\sqrt{16}}=0.04$

Approach #1

Standardizing the given limits:

$$z_1 = \frac{11.96 - 12}{0.04} = -1.0 \quad z_2 = \frac{12.08 - 12}{0.04} = 2.0$$

Probability that sample average will be between 11.96 and 12.08 is:

$$P(z_1 \leq \bar{x} \leq z_2) = P(-1.0 \leq \bar{x} \leq 2.0) = 0.8185$$

Since the limits have been standardized we can use standard normal distribution to compute probabilities:

```
>> pnorm(q=2) - pnorm(q=-1)
0.8186
```

Approach #2

If not using the standard normal distribution then mean and standard deviation must be specified.

```
>> pnorm(q=12.08, mean=12, sd=0.04) - pnorm(q=11.96, mean=12, sd=0.04)
0.8186
```

■

3.3. Exponential Distribution

In section (2.6.4) it was mentioned that in situations where we only know the rate of occurrence (λ) of an event where the events occur completely at random might be a good candidate to be modeled by a Poisson model. However, situations might arise where the time interval between consecutively occurring event is an important random variable.

The exponential distribution has many applications:

- The time to decay of a radioactive atom,
- The time to failure of components with constant failure rates,
- In the theory of waiting lines or queues (for example, time taken for an ambulance to arrive at the scene of an accident) (Forbes *et al*., 2011).

Suppose a series of events satisfying the Poisson process are occurring at a rate of λ per unit time. Let random variable Y denote the interval between consecutive events. Then,

$$f_Y(y)=\lambda e^{-\lambda y}, y>0 \tag{3.14}$$

MGF, Mean and Variance

$$M_Y(t)=\frac{\lambda}{\lambda-t} \tag{3.15}$$

$$E(Y)=\frac{1}{\lambda} \tag{3.16}$$

$$Var(Y)=\frac{1}{\lambda^2} \tag{3.17}$$

Example 3.3

During the period of 1832 to 1950, the following data was collected for the eruptions of a volcano:

126	73	3	6	37	23
73	23	2	65	94	51
26	21	6	68	16	20
6	18	6	41	40	18
41	11	12	38	77	61
26	3	38	50	91	12

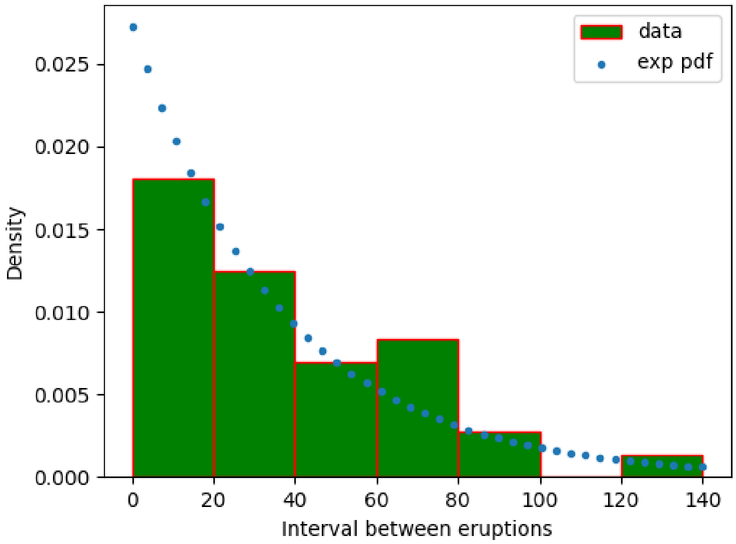
Can the data be described by an exponential distribution model? (Adapted from Larsen & Marx, 2011)

Solution:

In order to test whether exponential distribution is an adequate choice, first a density histogram of the data needs to be plotted. Then a scatter plot in the domain of the data using Eq. (3.14) will be overlaid. The following script handles both tasks:

```
Data = np.array([ [126, 73, 3, 6, 37, 23],
     [73, 23, 2, 65, 94, 51],
     [26, 21, 6, 68, 16, 20],
     [6, 18, 6, 41, 40, 18],
     [41, 11, 12, 38, 77, 61],
     [26, 3, 38, 50, 91, 12]])

X = Data.flatten() #1D

n, bins, patches = plt.hist(X, bins=range(0,160, 20), density=True, color = "green", ec="red", label="data")

x_axis = np.linspace(start=bins[0], stop=bins[-1], num=len(bins)*5)
plt.scatter(x=x_axis, y=dexp(x=x_axis, rate=1/np.average(X)), marker=".", label="exp pdf")

plt.xlabel("Interval between eruptions")
plt.ylabel("Density")
plt.legend()

plt.show()
```

The script will produce the following figure:



It is seen that the shape of the histogram is consistent with the theoretical model (exponential distribution).

Fig 3.5: *Histogram of raw data and exponential distribution describing data*

■

3.4. Gamma Distribution

In section (3.3), it was mentioned that if a series of events satisfying the Poisson process are occurring at a rate of λ per unit time and the random variable Y denote the interval between consecutive events it could be modeled with exponential distribution. Here the random variable Y can also be interpreted as the *waiting time* for the *first* occurrence.

This is similar to geometric distribution (section 2.4) where we were only interested in the trial where *first success occurs*. In section (2.5), in negative-binomial distribution instead of first success, we were interested in r^{th} success. Therefore, it was mentioned that the negative-binomial distribution generalizes the geometric distribution.

In a similar fashion, gamma distribution generalizes the exponential distribution such that we are now interested in the occurrence of (waiting time of) r^{th} event. However, before we proceed with the probability density function of gamma distribution we need to define the gamma function.

3.4.1. Gamma Function

It is a commonly used extension of factorial function and defined as:

$$\Gamma(z)=\int_0^\infty t^{z-1}\cdot e^{-t}\,dt \qquad \textbf{(3.18)}$$

With minor calculus, one can quickly see that $\Gamma(1)=1$. Using integration by parts[15], it is seen that: $\Gamma(z+1)=z\cdot\Gamma(z)$. Using induction one can further see that $\Gamma(n)=(n-1)!$. Experimenting using Python:

```python
import math

for i in [1, 2, 3, 4]:
    print(f"T({i})={math.gamma(i)}, ({i}-1)!={math.factorial(i-1)}")
```

T(1)=1.0, (1-1)!=1

T(2)=1.0, (2-1)!=1

T(3)=2.0, (3-1)!=2

T(4)=6.0, (4-1)!=6

15 https://en.wikipedia.org/wiki/Gamma_function

3.4.2. Probability Density Function

Suppose that Poisson events are occuring at constant rate of λ. Let random variable Y denote the waiting time for r^{th} event. Then,

$$f_Y(y)=\frac{\lambda^r}{(r-1)!}y^{r-1}e^{-\lambda y}, y>0 \quad \textbf{(3.19)}$$

A proof of Eq. (3.19) can be found in mathematical statistics textbooks (Larsen & Marx, 2011). Eq. (3.19) is often expressed in the following form (Devore *et al.*, 2021; Miller & Miller, 2014; R-Documentation[16]) :

$$f(x;\alpha,\beta)=\frac{1}{\beta^{\alpha}\Gamma(\alpha)}x^{\alpha-1}e^{-x/\beta}, x>0 \quad \textbf{(3.20)}$$

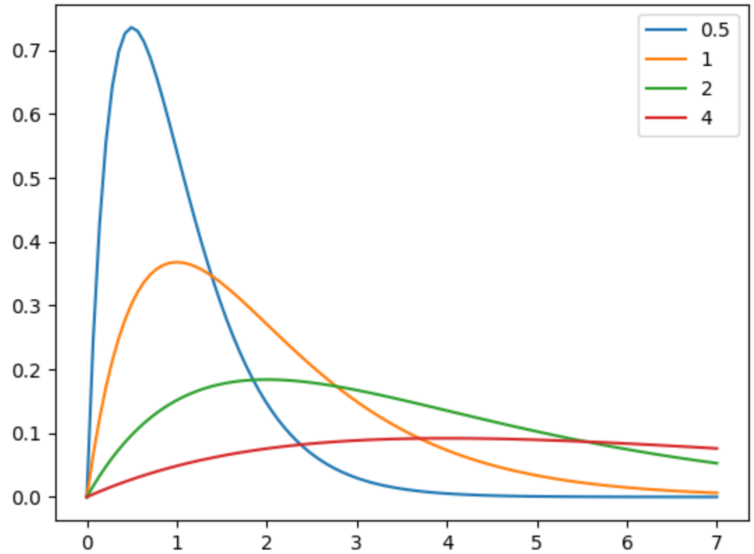
Softwares such as *R* and *scisuit* Python package calls α as the shape and β as the scale parameter. Note that in Eq. (3.19) $r=\alpha$ and $\lambda=1/\beta$. When $\beta=1$ the distribution is called the standard gamma distribution.

Devore et al. (2021) states that the parameter β is called a scale parameter because values other than 1 either stretch or compress the pdf in the x-direction. Let's visualize this using a constant shape factor, *shape=2*:

```
x=np.linspace(0, 7, num=100)

for beta in [0.5, 1, 2, 4]:
    plt.plot(x, dgamma(x=x, shape=2, scale=beta), label=str(beta))

plt.legend()
plt.show()
```

The following figure will be generated:

16 https://search.r-project.org/CRAN/refmans/ExtDist/html/Gamma.html



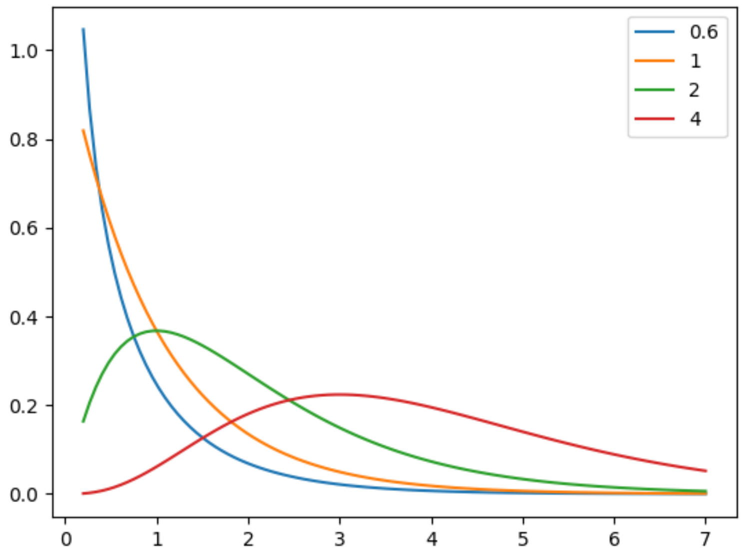
It is seen that for β=1, max value is around 0.35.

For a smaller β value, the curve is “compressed” and therefore became narrower and the max value increased to ~0.7.

For a larger β value the curve is “stretched” and therefore became wider and max value decreased to ~0.2 for β=2.

Fig 3.6: *Gamma density curves for different scale (β) values (α=2)*

With minor editing if the same script is run for different values of α=[0.6, 1, 2, 4], where β=1, then the following figure will be obtained:



It is seen that:

1) when α≤1, the curve is strictly decreasing as x increases.

2) when α> 1, f(x; α) rises to a maximum and then decreases as x increases.

Fig 3.7: *Standard gamma (β=1) density curves for different shapes (α)*

3.4.3. MGF, Mean and Variance

$$M_Y(t)=\frac{1}{(1-\beta t)^{\alpha}} \quad \textbf{(3.21)}$$

$$E(Y)=\alpha\cdot\beta \quad \textbf{(3.22)}$$

$$Var(Y)=\alpha\cdot\beta^2 \quad \textbf{(3.23)}$$

Example 3.4

As a process engineer you are given the task of designing a system to pump fluid from a reservoir to the processing plant. As this is important for the manufacturing to continue smoothly you have included two pumps, one active and one as a backup to be brought on line.

The manufacturer of the pump specifies that the pump is expected to fail once every 100 hours. What are the chances that the whole manufacturing will not remain functioning for 50 hours? (Adapted from Larsen & Marx, 2011)

Solution:

For the whole manufacturing to be interrupted, 2 pumps should fail, for example first after 10 hours and second after 40 hours…

Failure rate: λ= 0.01 failure/hour

Approach 1: We are going to use Eq. (3.19) where λ= 0.01 and r=2.

$$f_Y(y)=\frac{0.01^2}{(2-1)!}y^{2-1}e^{-0.01y}$$

$$P(\text{manufacturing fails to last for 50 hours})=\int_0^{50}0.01^2\,y\,e^{-0.01y}=0.09$$

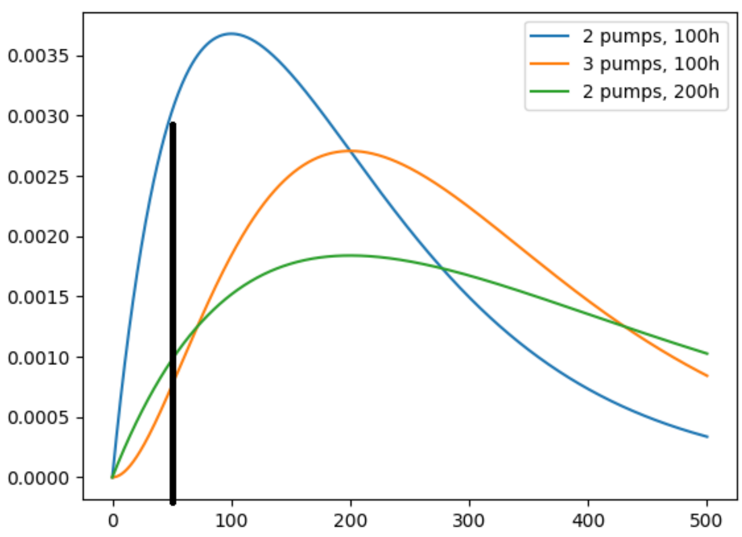
Approach 2: We are going to use Eq. (3.20) where β= 100 and α=2.

```
>> pgamma(q=50, shape=2, scale=100)
0.09
```

Assume that 9% probability is too high for you. Another manufacturer claims that the pump they are offering is expected to fail once every 200 hours, but the price is double, therefore your costs will double. Would you use 3 pumps where each is expected to fail once every 100 hours or 2 pumps where each is expected to fail once every 200 hours to minimize the probability of 9%?

We will use a short script to generate the probability density curves and inspect the pdf's.

```
x=np.linspace(0, 500, num=1000)

plt.plot(x, dgamma(x=x, shape=2, scale=100), label="2 pumps, 100h")
plt.plot(x, dgamma(x=x, shape=3, scale=100), label="3 pumps, 100h")
plt.plot(x, dgamma(x=x, shape=2, scale=200), label="2 pumps, 200h")

plt.legend()
plt.show()
```



It is seen that using 3 pumps where each pump is expected to fail once every 100 hours will have a lower probability than using 2 pumps where each pump is expected fail once every 200 hours. Therefore, there is no need to double the cost.

However, also note that if instead of 50 hours we would like 80 hours or greater than using 2 pumps where each pump is expected fail once every 200 hours is a more reasonable approach in terms of lowering the probability.

Fig 3.8: *Gamma curves for different number of pumps and failure rates.*

3.5. Chi-Square Distribution

Chi-squared distribution is the sum of the squares of a number of normal distribution and this fact gives to important applications of it, i.e. analysis of contingency tables (Forbes *et al*., 2011).

3.5.1. One-way Frequency Table

Categorical univariate data consists of non-numerical observations which maybe placed in categories(Wikipedia[17]) and are most conveniently summarized in a one-way frequency table (Peck *et al*., 2016). Suppose 100 people being surveyed whether they will go to a certain movie and choices (categories) are: *Definitely will, Probably will, Probably will not, Definitely will not*. Now a table can be formed from counting the observations:

Table 3.1: Results of the hypothetical survey

	Definitely	Probably	Probably not	Definitely not
Frequency	20	40	25	15

Let k be the number of categories of a categorical variable and p_k population proportion for category $k>0$. Then,

H_0: p_k is the hypothesized proportion for category k

H_a: H_0 is not true (at least one of the population category proportions differs from the corresponding hypothesized value).

$$X^2 = \frac{\sum_{\text{all cells}} (\text{Observed cell count} - \text{Expected cell count})^2}{\text{Expected cell count}} \qquad \textbf{(3.24)}$$

where X^2 has approximately a chi-square distribution with $df = k-1$.

17 https://en.wikipedia.org/wiki/Univariate_(statistics)

Example 3.5

Lunar Phase	Number of Days	Number of Births
Phase 1	24	7680
Phase 2	152	48442
Phase 3	24	7579
Phase 4	149	47814
Phase 5	24	7711
Phase 6	150	47595
Phase 7	24	7733
Phase 8	152	48230

An urban legend claims that more babies are born during certain phases of the lunar cycle, especially near the full moon. Data for a sample of randomly selected births occurring during 24 lunar cycles are given in the table. Test whether the data support the urban legend claim (Adapted from Peck *et al*., 2016).

Solution:

There are 699 total days and a total of 222,784 births. The probability of a birth to happen at *Phase 1* is *24/699=0.0343* and at *Phase 8* is *152/699=0.2175*.

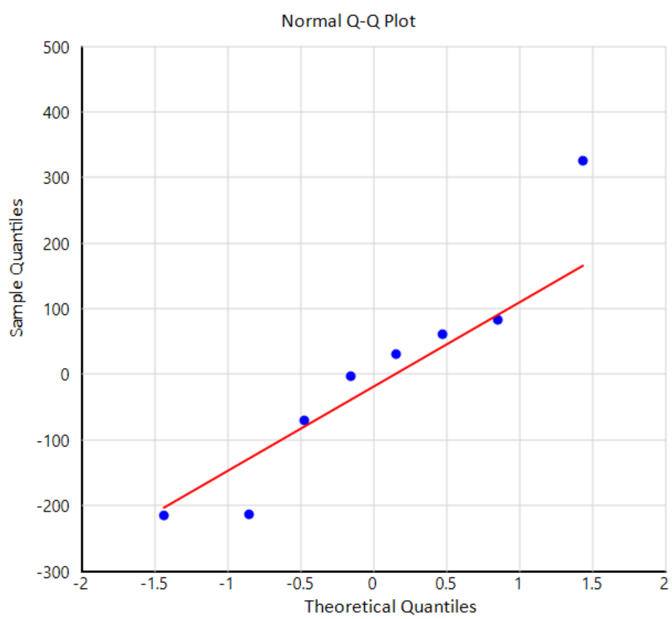
So if lunar phase did not have any effect, then we expect that at *Phase 1* there would be $0.0343 \times 222784 = 7649.23$ births. We continue our computations in this fashion and then use Eq. (3.24) to compute X^2 value.

```
from scisuit.stats import pchisq

#Lunar periods
days = np.array([24, 152, 24, 149, 24, 150, 24, 152])

#Observed births at each lunar cycle
observed = np.array([7680, 48442, 7579, 47814, 7711, 47595, 7733, 48230])

#probabilities (ratios)
probs = days / np.sum(days)

#expected birth numbers
expected = np.sum(observed)*probs

chisq = (expected-observed)**2 / expected

#pchisq gives left tails probability
pval = 1 - pchisq(q=np.sum(chisq), df=len(chisq) - 1)

print(f"p-value: {round(pval, 3)}")
```

The output is: *p-value: 0.504*. Therefore, we cannot accept H_a (population category proportions differs from the corresponding hypothesized value). Thus, the claim is not supported by statistical evidence.

Forbes et al. (2011) states that to be able to use Eq. (3.24), the data produced from the differences between observed and expected values should be normally distributed. We will use QQ plot to check whether the data is normally distributed.

```
from scisuit.plot import qqnorm
from scisuit.stats import test_norm_ad
from scisuit import App

app=App()

diff = observed-expected

qqnorm(data=diff)
print(f"Anderson-Darling test: {test_norm_ad(x=diff)}")

app.mainloop()
```



Although there is an outlier point, the rest of the data follows the QQ-Line fairly well.

Moreover, the p-value reported by Anderson-Darling test is 0.448, therefore we cannot reject H_0 that “the data follows normal distribution”.

Fig 3.9: *QQ plot of the differences between observed and expected birth rates.*

■

3.5.2. Probability Density Function

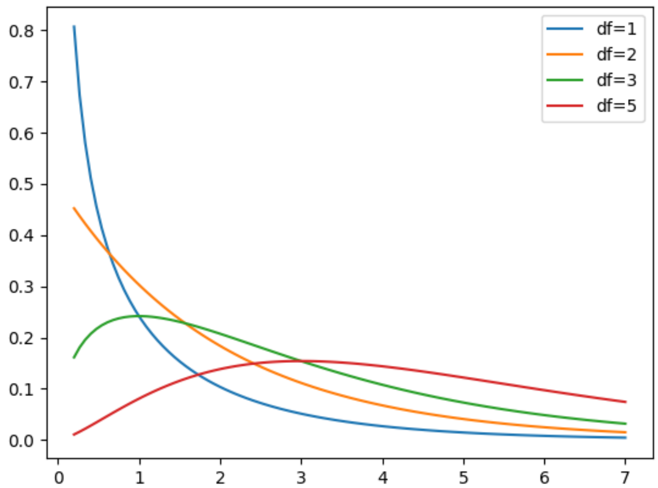
A random variable Y is said to have a chi-square distribution with n degrees of freedom ($n>0$), if

$$f_Y(y)=\frac{1}{2^{n/2}\Gamma(n/2)}y^{(n/2)-1}e^{-y/2}, y>0 \qquad \textbf{(3.25)}$$

Please note that Eq. (3.25) is a special case of Eq. (3.19) where $r=n/2$ and $\lambda=1/2$. Substituting *n/2* and *1/2* for r and λ in Eq. (3.19) and tidying up slightly yields:

$$f_Y(y)=\frac{1}{2^{(n/2)}(n/2-1)!}y^{n/2-1}e^{-1/2\,y}$$

Noticing that $(n/2-1)!=\Gamma(n/2)$, then one can see that above equation is equal to Eq. (3.25).



The shape of chi-square distribution depends on the value of degrees of freedom(df):
df<3: decreases strictly as x increases,
df≥3: increases to a maximum and then decreases.

It should also be noted that regardless of the degrees of freedom, all chi-square distributions are skewed to right.

Fig 3.10: *Chi-square distribution with different degrees of freedoms*

Theorem: Let $Z_1, Z_2, \ldots, Z_n$ be n independent standard normal random variables. Then $\sum_{i=1}^{n} Z_i^2$ has a chi-square distribution with n degrees of freedom.

A proof of the theorem can be found in mathematical statistics textbooks (Larsen & Marx, 2011).

3.5.3. MGF, Mean and Variance

$$M_Y(t)=(1-2t)^{-n/2}, t<1/2 \quad \textbf{(3.26)}$$

$$E(Y)=n \quad \textbf{(3.27)}$$

$$Var(Y)=2n \quad \textbf{(3.28)}$$

```
from scisuit.stats import rchisq

#number of samples
N = 1000

#arbitrary values for degrees of freedom
df = [1, 3, 5, 10]

#2D array of random values for each degrees of freedom
X=np.array([rchisq(N, x) for x in df])

#mean and variance
print(f"mean = {np.average(X, axis=1)}")
print(f"variance = {np.var(X, axis=1, ddof=0)}")
```

mean = [1.019 2.968 4.914 9.817]
variance = [2.131 5.696 9.669 19.553]

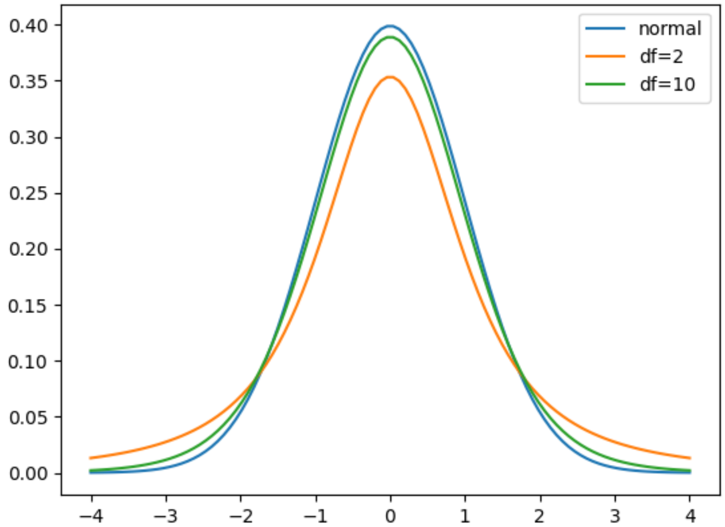
Notice how close the values are to the values that would be computed by Eqs. (3.27 & 3.28). For example for *df=1*, *E(Y)=1* and *Var(Y)=2*.

3.6. The Student's *t* distribution

The t distribution is used to test whether the difference between the means of two samples of observations is statistically significant assuming they were drawn from the same population (Forbes *et al.*, 2011).

In sections (3.2.2 & 3.2.3) it was shown that if $y_1, y_2, \ldots, y_n$ is a random sample from a normal distribution with mean μ and standard deviation ρ then $\frac{\bar{Y}-\mu}{\rho/n}$ has a standard normal distribution. However Gosset (Student, 1908) realized that $\frac{\bar{Y}-\mu}{s/n}$ does not have a standard normal distribution and derived the probability density function of it. Let's see the differences between standard normal distribution and t-distribution using the short Python code:

```
from scisuit.stats import dnorm, dt

x=np.linspace(-4, 4, num=100)

plt.plot(x, dnorm(x=x), label="normal")

for n in [2, 10]:
    plt.plot(x, dt(x=x, df=n), label="df="+str(n))

plt.legend()
plt.show()
```



1) Both dists are symmetric.

2) Both dists have a mean of 0.

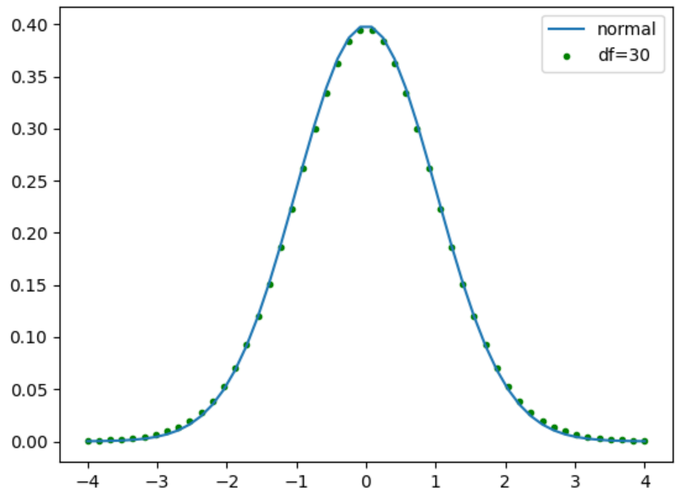
3) t-dist is characterized by the degrees of freedom (df). As df increases, t-dist becomes more similar to a normal dist.

4) The curves of t-dist with larger df are taller and have thinner tails.

5) t-dist is most useful for small sample sizes.

Fig 3.11: *Standard normal distribution and t-distribution with different degrees of freedoms*

In comparison of t-distribution with standard normal distribution it was mentioned that t-distribution is most useful for *small* sample sizes, but have not explained what is meant by *small*. Larsen & Marx (2011) states that many tables providing probability values for t-distribution will have it for degrees of freedom in the range of [1, 30]. Furthermore, elsewhere[18] it is mentioned that for a sample size of at least 30, standard normal distribution can be used instead of t-distribution.



One can see that t-dist with 30 degrees of freedom well overlaps with a standard normal dist.

Fig 3.12: *Standard normal distribution and t-distribution with df=30*

Let Z be a standard normal random variable and *V* an independent chi-square random variable with *n* degrees of freedom. The Student t ratio with *n* degrees of freedom is,

$$T_n=\frac{Z}{\sqrt{V/n}} \qquad \textbf{(3.29)}$$

In line with observations from Fig. (3.11), Eq. (3.29) is symmetric: $f_{T_n}(t)=f_{T_n}(-t)$.

The probability density function for a Student t random variable with *n* degrees of freedom is,

$$f_{T_n}(n)=\frac{\Gamma\left(\frac{n+1}{2}\right)}{\sqrt{n\pi}\,\Gamma\left(\frac{n}{2}\right)\left(1+\frac{t^2}{n}\right)^{\frac{(n+1)}{2}}},-\infty<t<\infty \qquad \textbf{(3.30)}$$

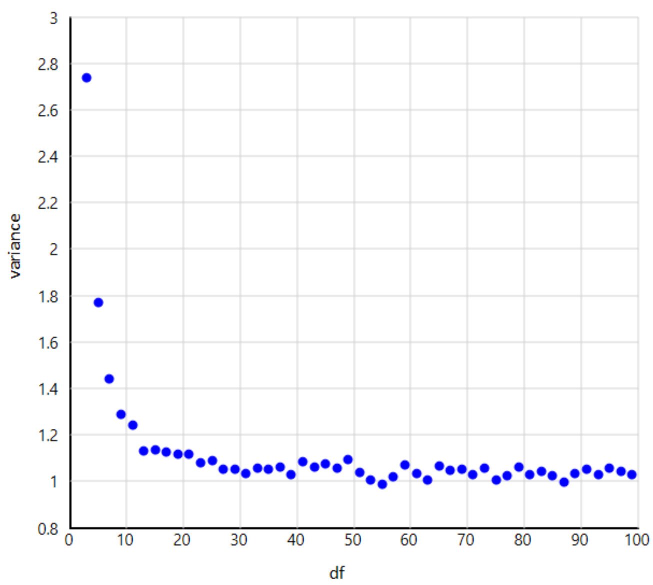
18 https://www.jmp.com/en_no/statistics-knowledge-portal/t-test/t-distribution.html

MGF, Mean and Variance

The moment-generating function of t-distribution is undefined[19] and its mean is 0 as can be observed from Figs. (3.11 & 3.12) for different degrees of freedom.

$$Var(Y)=\frac{n}{n-2}, n>2 \tag{3.31}$$

```
from scisuit.plot import scatter
from scisuit import App

dfs = range(3, 100, 2) #degrees of freedom
Variance = []

for df in dfs:
    x = np.array(rt(n=5000, df=df))
    Variance.append(np.var(x, ddof=0))

app = App()

scatter(x=list(dfs), y=Variance)

app.mainloop()
```



It is seen that for df>2 the variance is always larger than 1 and for large df the variance is close to 1 (this can also be observed from the equation).

Devore *et al*. (2021) states that for small dfs the t-dist curve spreads out more than the standard normal dist curve; however, for large dfs the t-dist curve approaches to standard normal dist curve (μ=0, ρ=1) (see above figure).

Fig 3.13: *Variance of t-distribution with different df values*

19 https://en.wikipedia.org/wiki/Student's_t-distribution

3.7. F (Fisher–Snedecor) Distribution

It is the ratio of independent chi-square random variables. Many experimental scientists use the technique called analysis of variance (ANOVA) (Forbes *et al.*, 2011). ANOVA analyzes the variability in the data to see how much can be attributed to differences in the means and how much is due to variability in the individual populations (Peck *et al.*, 2016). In one-way ANOVA, F is the ratio of variation among the samples to variation within the samples.

Suppose that U and V are independent chi-square random variables with m and n degrees of freedom, respectively. Then,

$$F=\frac{U/m}{V/n} \tag{3.32}$$

The probability density function for F distribution is:

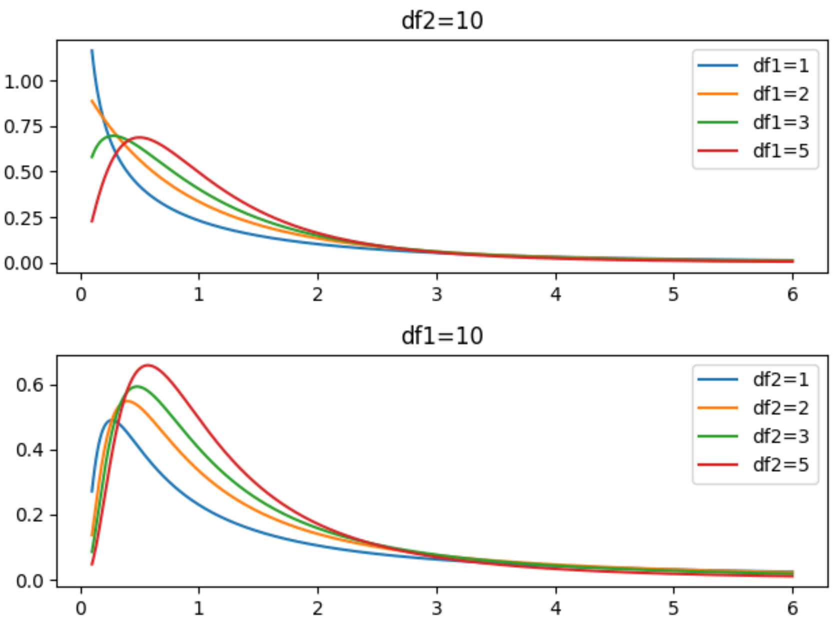
$$f_{F_{m,n}}(r)=\frac{\Gamma\left(\frac{m+n}{2}\right)m^{m/2}n^{n/2}r^{(m/2)-1}}{\Gamma\left(\frac{m}{2}\right)\Gamma\left(\frac{n}{2}\right)(n+mr)^{(m+n)/2}},r>0 \tag{3.33}$$

The derivation of Eq. (3.33) is detailed in the textbook from Larsen & Marx (2011).

Let's use a fairly short script to generate F-distribution curves for constant m (df_1) and varying n (df_2) and for constant df_2 and varying df_1.

```
from scisuit.stats import df

x_axis=np.linspace(0.1, 6, num=500)
dfree = 10

figure, axis = plt.subplots(2)
for x in [1, 2, 3, 5]:
    axis[0].plot(x_axis, df(x_axis, df1=x, df2=dfree), label="df1="+str(x))
    axis[1].plot(x_axis, df(x_axis, df1=dfree, df2=x), label="df2="+str(x))

axis[0].set_title("df2=10")
axis[1].set_title("df1=10")
```

```
axis[0].legend()
axis[1].legend()

plt.tight_layout()
plt.show()
```



It is seen that when **df2** is constant the F dist curves looks very much like a typical chi-square dist curves.

When **df1** is constant, all F dist curves rapidly rises to a maximum and then decreases in value as x increases.

In all cases, F values are never negative and sharply skewed to the right.

Fig 3.14: *F-distribution curves for constant A) df2, B) df1*

MGF, Mean and Variance

The moment-generating function of F distribution does not exist[20].

$$E(Y)=\frac{n}{n-2}, n>2 \quad \textbf{(3.34)}$$

$$Var(Y)=\frac{2n^2(m+n-2)}{m(n-2)^2(n-4)} \quad \textbf{(3.35)}$$

20 https://en.wikipedia.org/wiki/F-distribution

4. REFERENCES

Bury K (1999). Statistical Distributions in Engineering, Cambridge University Press.

Carlton MA, Devore JL (2014). Probability with Applications in Engineering, Science and Technology. Springer USA.

Devore JL, Berk KN, Carlton MA (2021). Modern Mathematical Statistics with Applications. 3rd Ed., Springer.

Forbes C, Evans M, Hastings N, Peacock B (2011). Statistical Distributions, 4th Ed., Wiley.

Hogg RV, McKean JW, Craig AT (2019). Introduction to mathematical statistics, 8th Ed., Pearson.

Kreyszig E, Kreyszig H, Norminton EJ (2011). Advanced Engineering Mathematics, 10th Ed., John Wiley & Sons Inc.

Larsen RJ, Marx ML (2011). An Introduction to Mathematical Statistics and Its Applications. 5th Ed., Prentice Hall.

Miller I, Miller M (2014). John E. Freund's Mathematical Statistics with Applications. 8th Ed., Person New International Edition.

Peck R, Olsen C, Devore JL (2016). Introduction to Statistics and Data Analysis. 5th Ed., Cengage Learning.

Pinheiro, CAR, Patetta M (2021).Introduction to Statistical and Machine Learning Methods for Data Science. Cary, NC: SAS Institute Inc.

Stahl S (2006). The Evolution of the Normal Distribution. *Mathematics Magazine*, 76(2), pp. 96-113. Available at: https://www.maa.org/sites/default/files/pdf/upload_library/22/Allendoerfer/stahl96.pdf

Student (1908). The probable error of a mean. *Biometrika*, 6(1), 1-25.

Wackerly DD, Mendenhall W, Scheaffer RL (2008). Mathematical Statistics with Applications, 7th Ed., Thomson/Brooks Cole.

Walck C (2007). Handbook on Statistical Distributions for Experimentalists. Available at: https://s3.cern.ch/inspire-prod-files-1/1ab434101d8a444500856db124098f9c