

Chapter 7 Probability Distributions Assessment

“A reasonable probability is the only certainty.”

E.W. Howe

7.1	Introduction.....	2
7.2	Assessing Probabilities from Experts	3
7.2.1	Eliciting Discrete Probability Distribution from an Expert	3
7.2.2	Eliciting Continuous Probability Distribution from an Expert	4
7.2.3	Fitting Expert’s CDF to a Theoretical Distribution	5
7.3	Biases in Probability Assessment	9
7.3.1	Types of Biases	9
7.3.2	Sources of Bias.....	10
7.3.3	Probability Assessment Protocols.....	12
7.4	Learning Probability Distributions from Data.....	13
7.4.1	Relative Frequencies Method.....	13
7.4.2	Some Commonly Used Probability Distributions.....	14
7.4.3	Fitting Distribution to Data.....	20
7.4.4	Fitting Normal Distribution with Two Known Percentile Values	35
7.5	Discrete Approximation of Continuous Probability Distribution.....	38
7.5.1	Introduction.....	38
7.5.2	Desired Properties of the Discrete Approximation.....	38
7.5.3	Limit on the number of moments that can be preserved.....	39
7.5.4	Discrete Approximation for the Uniform Distributions.....	40
7.5.5	Discrete Approximation for Normal Distributions	41
7.5.6	3-Branch Discretization of the Asymmetric Triangular Distribution	43
7.5.7	Discrete Approximation for Continuous Distributions in DPL	44
7.5.8	Three-Point Quick Approximation Methods	44
	References.....	45
	Exercises.....	46

7.1 Introduction

- To perform Decision Analysis, we require the probability distributions of all the uncertain variables in the decision model.
- These probability distributions may be obtained by the following methods:
 1. Assessed directly via **Experts' Judgments** using **Probability Assessment Protocols**.
 2. **Learned from Data** using *Statistical or Machine Learning*.
 3. **Generated** from *Stochastic Models* such as queuing theory, Monte Carlo simulation, discrete event simulation, Markov chain, etc. But these models also require probability distribution assessments.
 4. Combinations of any of the above.
- We will cover methods 1 and 2 briefly in this chapter.
- Method 3 is covered in other IE courses.
- Note that although we have adopted the subjective view of the meaning of probabilities, we may still combine probabilities obtained objectively from data/models with those elicited directly from experts using the rules of Probability Theory.

7.2 Assessing Probabilities from Experts

7.2.1 Eliciting Discrete Probability Distribution from an Expert

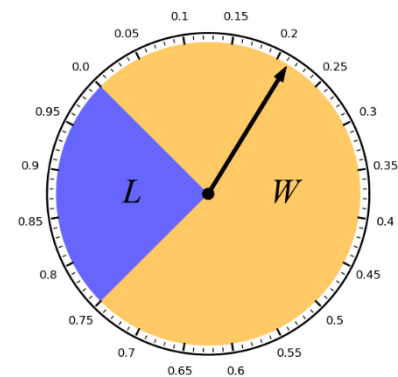
- For a discrete variable, the method will produce a probability mass function (PMF) for the variable.
- Let X be a discrete uncertain variable with two outcomes: x_1 and x_2 .
- The steps to assess the probability distribution of X are as follows:
 - Let W and L be two rewards such that $W \succ L$.
For example, W could be a small cash amount and L is nothing.
 - Set the probability wheel with the orange sector at p .
 - Repeat**
 - Ask the expert to choose between the two options:

A: Spin the probability wheel.
If the outcome is orange, he receives W .
Otherwise, he receives L .

B: Do not spin the wheel.
If the outcome of $X = x_1$, he receives W .
Otherwise, he receives L .
 - If the expert chose *A*, reduce the value of p set on the wheel.
If the expert chose *B*, increase the value of p set on the wheel.
 - Until** the expert is indifferent between options *A* and *B*.
 - Result:

$$\Pr \{X = x_1\} = p$$

$$\Pr \{X = x_2\} = 1 - p.$$



Discrete variables with more than 2 outcomes:

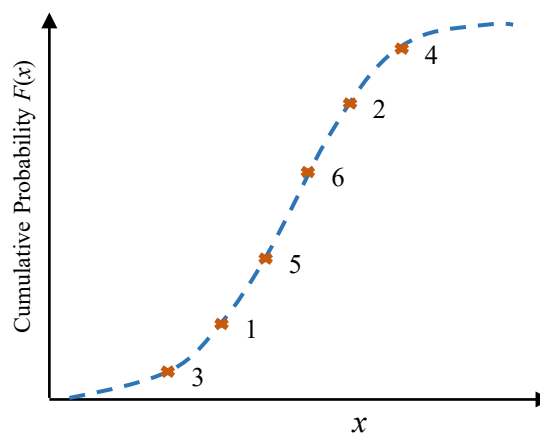
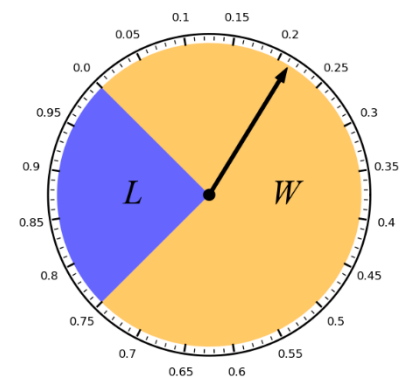
- If X has more than two discrete outcomes, the procedure above is repeated for outcomes x_2 to x_{n-1} to obtain p_2, \dots, p_{n-1} , and let $p_n = 1 - \sum_{i=1}^{n-1} p_i$.
- The output is a probability mass function for X .

7.2.2 Eliciting Continuous Probability Distribution from an Expert

- For a continuous uncertain variable, the method will produce a cumulative distribution function (CDF) for the variable.
- Let X be a continuous uncertain variable.
- The steps to assess its CDF are as follows:
 - Let W and L be two rewards such that $W \succ L$.
For example, W could be a small cash amount and L is nothing.
 - Select a value of x from the range of possible values of X based on a selected assessment protocol.
 - Set the probability wheel with the orange sector at p .
 - Repeat**
 - Ask the expert to choose between the two options:

A: Spin the probability wheel.
If the outcome is orange, he receives W .
Otherwise, he receives L .

B: Do not spin the wheel.
If the actual outcome of $X \leq x$, he receives W .
Otherwise, he receives L .
 - If the expert chose *A*, reduce the value of p set on the wheel.
If the expert chose *B*: increase the value of p set on the wheel.
 - Until** the expert is indifferent between options *A* and *B*.
 - Encode $\Pr(X \leq x) = p$ on the CDF for X .
 - Repeat Steps 1 to 8 for a number of other selected values of x based on the selected assessment protocols. Each of these values results in a point on CDF for X .
 - Plot the CDF for X .



- Note: The choice and sequence of x values are important to avoid biases in the results. See Stanford/SRI protocol for guidelines on how to choose the sequence of values.

7.2.3 Fitting Expert's CDF to a Theoretical Distribution

- The probability distribution for a continuous variable assessed by an expert are points on a CDF.
- One way to use this information in a decision tree or an influence diagram is to fit the CDF of a theoretical probability distribution to the data.

Example: Using Python `scipy.optimize.curve_fit` function.

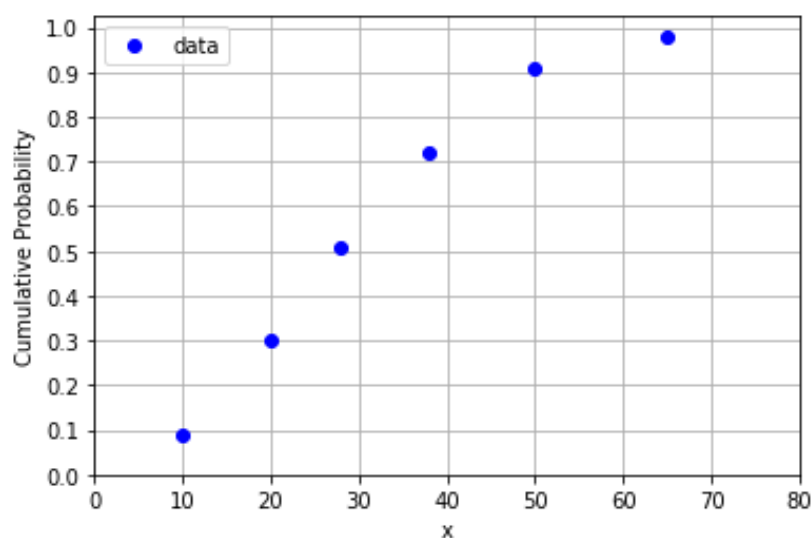
- An expert assessed the following 6-point CDF for an uncertain variable X .

x	$F(x)$
10	0.09
20	0.30
28	0.51
38	0.72
50	0.91
65	0.98

```
In [1]: """ Fit a distribution to Expert's CDF data """
import numpy as np
import matplotlib.pyplot as plt
from scipy.optimize import curve_fit

# Expert's CDF data points
xdata = [ 10,  20,  28,  38,  50,  65 ]
ydata = [0.09, 0.30, 0.51, 0.72, 0.91, 0.98 ]
```

```
In [2]: # Visualize the CDF data points
fig0, ax0 = plt.subplots()
ax0.plot(xdata, ydata, 'bo', ms=6, label='data')
ax0.set_xlim(0, 80)
ax0.set_yticks(np.linspace(0, 1, 11))
ax0.set_ylabel("Cumulative Probability")
ax0.set_xlabel("x")
ax0.grid()
ax0.legend()
plt.show()
```

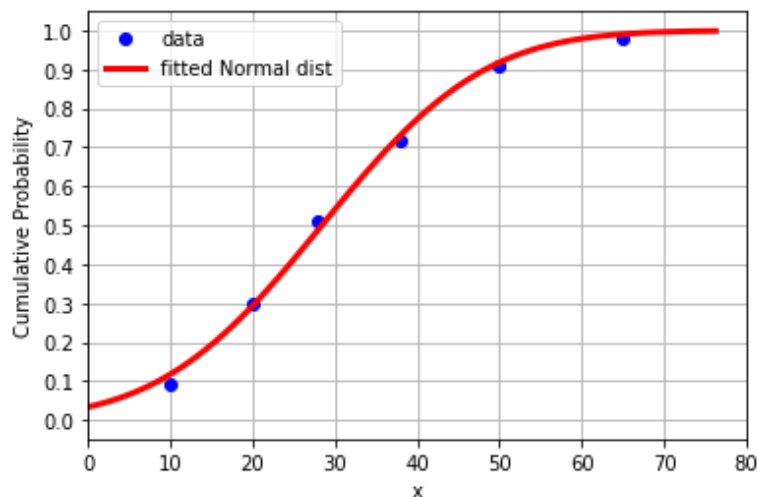


```
In [3]: # First, try fitting a Normal Distribution
from scipy.stats import norm
# The CDF to fit
cdf_norm = lambda x, mu, sigma: norm(mu, sigma).cdf(x)

# Fit the data and show results
par_norm, pcov_norm = curve_fit(cdf_norm, xdata, ydata, p0=(50,10))
print("\nFitted Normal Distribution:")
print(f"  mean = {norm(*par_norm).mean():.6f}")
print(f"  sd = {norm(*par_norm).std():.6f}")
```

```
Fitted Normal Distribution:
mean = 28.425794
sd = 15.480955
```

```
In [4]: # Visualize the fitted Normal distribution against the data
fig1, ax1 = plt.subplots()
ax1.plot(xdata, ydata, 'bo', ms=6, label='data')
xnorm = np.linspace(norm.ppf(0.001, *par_norm),
                    norm.ppf(0.999, *par_norm), 100)
ax1.plot(xnorm, norm(*par_norm).cdf(xnorm),
        'r-', lw=3, label='fitted Normal dist')
ax1.set_xlim(0, 80)
ax1.set_yticks(np.linspace(0,1,11))
ax1.set_ylabel("Cumulative Probability")
ax1.set_xlabel("Value")
ax1.grid()
ax1.legend()
plt.show()
```



- The fitted Normal distribution looks reasonable, but the left tail does not fit well and goes theoretically to minus infinity.
- Also, the expert's CDF is skewed a bit to the left.
- Let's try to fit an asymmetrical Beta distribution to the data instead.

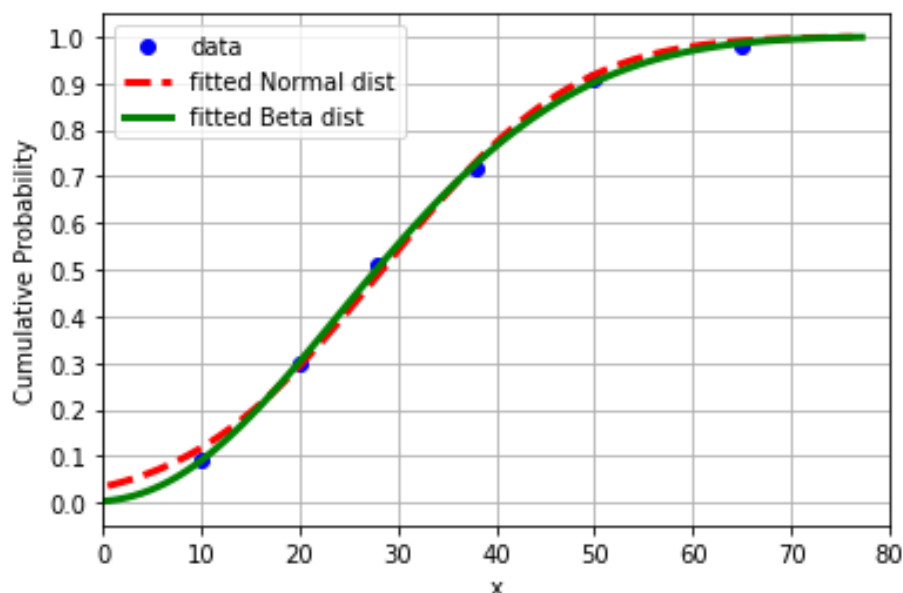
```
In [5]: # Try fitting a 4-parameter Beta distribution
from scipy.stats import beta
# The CDF to fit
cdf_beta = lambda x, a, b, loc, scale: beta(a,b,loc,scale).cdf(x)

# Fit the data and show results
par_beta, pcov_beta = curve_fit(cdf_beta, xdata, ydata, p0=(5,5,0,50))
print("\nFitted 4-parameter Beta Distribution:")
print(f"  a = {par_beta[0]:.6f}")
print(f"  b = {par_beta[1]:.6f}")
print(f"  loc = {par_beta[2]:.6f}")
print(f"  scale = {par_beta[3]:.6f}")
print(f"  mean = {beta(*par_beta).mean():.4f}")
print(f"  sd = {beta(*par_beta).std():.4f}")
```

Fitted 4-parameter Beta Distribution:

```
a = 2.842947
b = 5.933263
loc = -3.115719
scale = 99.784961
mean = 29.2084
sd = 14.9350
```

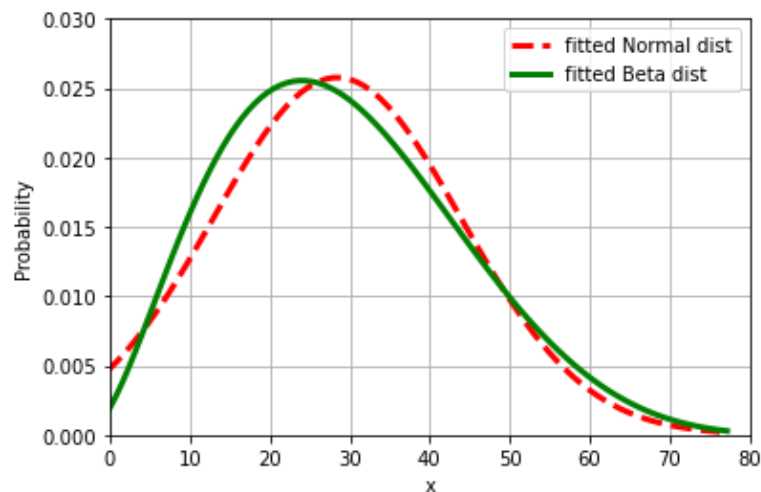
```
In [6]: # Visualize the 2 fitted distributions against the data
fig2, ax2 = plt.subplots()
ax2.plot(xdata, ydata, 'bo', ms=6, label='data')
ax2.plot(xnorm, norm(*par_norm).cdf(xnorm),
        'r--', lw=3, label='fitted Normal dist')
xbeta = np.linspace(beta.ppf(0.001, *par_beta),
                    beta.ppf(0.999, *par_beta), 100)
ax2.plot(xbeta, beta(*par_beta).cdf(xbeta),
        'g-', lw=3, label='fitted Beta dist')
ax2.set_xlim(0, 80)
ax2.set_yticks(np.linspace(0,1,11))
ax2.set_ylabel("Cumulative Probability")
ax2.set_xlabel("x")
ax2.grid()
ax2.legend()
plt.show()
```



- The Beta distribution is a better fit than the Normal distribution.

In [7]: *# Compare the PDF of the 2 fitted distributions.*

```
fig3, ax3 = plt.subplots()
ax3.plot(xnorm, norm(*par_norm).pdf(xnorm),
        'r--', lw=3, label='fitted Normal dist')
ax3.plot(xbeta, beta(*par_beta).pdf(xbeta),
        'g', lw=3, label='fitted Beta dist')
ax3.set_xlim(0, 80)
ax3.set_ylim(0, 0.03)
ax3.set_ylabel("Probability")
ax3.set_xlabel("x")
ax3.grid()
ax3.legend()
plt.show()
```



Note about the parameters for the standard Beta distribution in DPL

- In DPL the parameters a' , b' for the standard (2-parameter) Beta distribution $\text{Beta}(a', b')$ are defined differently from the parameters a , b used in the distribution `scipy.stats.beta(a, b, loc=0, scale=1)` where $a' = a$ and $b' = a + b$.

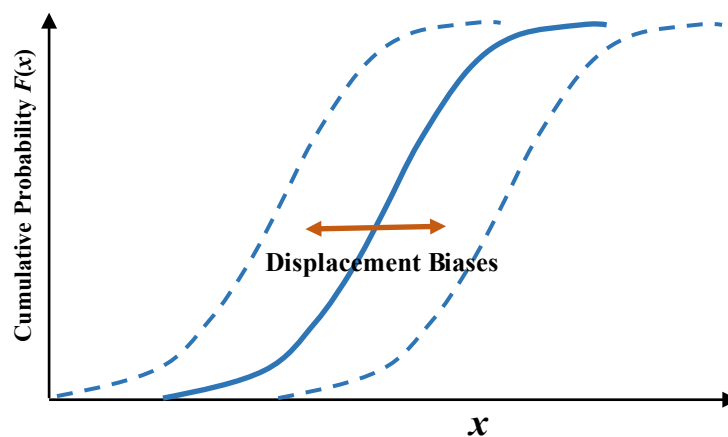
7.3 Biases in Probability Assessment

7.3.1 Types of Biases

- The task of the analyst is to elicit from the expert a probability distribution that best describes the underlying knowledge of the expert.
- Conscious or subconscious discrepancies between the expert's responses and an accurate description of his underlying knowledge are called **biases**.
- There are mainly two types of biases:
 1. Displacement bias
 2. Variability bias

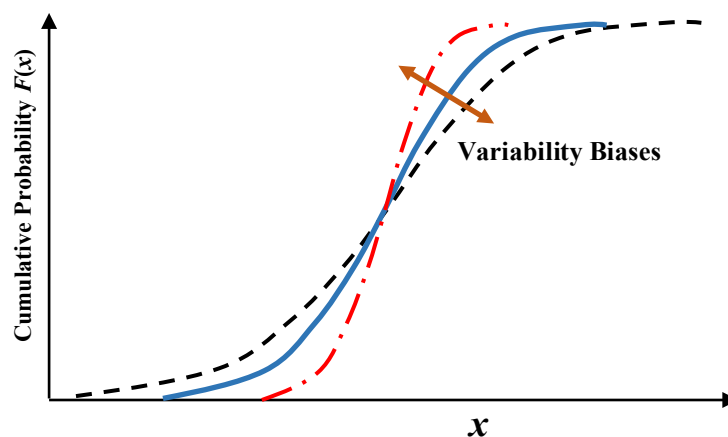
Displacement Bias

- Here there is a shift of the estimated probability distribution from the correct distribution in one direction.



Variability Bias

- Here there is an error in the shape of the distribution compared with the correct distribution.



- When the assessed distribution becomes tighter (has less spread) than the actual distribution (i.e., the red distribution), it is known as **Central Bias**.
- It shows overconfidence in the estimation of the variability of the variable by the expert.

7.3.2 Sources of Bias

- Sources of bias in probability assessment may be classified into two categories:
 1. Motivational biases
 2. Cognitive biases

1. Motivational Biases

- These are either conscious or subconscious adjustments in the expert's responses motivated by his personal interests. He may want to influence the decision in his favor by giving a particular set of responses.

Example

- A sale manager may consciously give a low prediction of potential sales so that he will look better if the actual sales, later on, exceed his forecast.

2. Cognitive Biases

- These are either conscious or subconscious adjustments in the expert's responses that are systematically introduced by the way he intellectually processes his perceptions depending on his mode of judgment.

- **Major Sources of Cognitive Bias**

1. Availability
2. Adjustment and Anchoring
3. Representativeness
4. Unstated assumptions
5. Coherence

1. Availability

- People often assign probabilities based on information that they can recall or visualize.
- For example, the probability that a machine may break down may be assigned by recalling past occurrences of breakdown.
- *Availability* refers to the ease with which relevant information is recalled or visualized.
- Information that are more recent or had made a great impression on the expert is easy to recall and may be given too much weight thus creating a bias.

Example

- A person who takes the MRT (subway) every day will assign a higher probability that the train service will be disrupted than another person who seldom takes the train.
- The MRT rider can easily recall all the incidents that had happened, whereas the other person may rely on reported news media.

2. Adjustment and Anchoring

- People often assign the probability of an event initially using the most readily available piece of information, and then subsequently make adjustments from there. However, the adjustments may be insufficient to encompass the range of values that could actually occur.
- The assessment appears to be “anchored” to the initially estimated value resulting in a central bias.

Example

- When predicting this year’s sales, one may use last year’s sales as a starting point. Recently years’ sales with the biggest and smallest sales are then used as extreme values for this year’s sales.

3. Representativeness

- People often assign the probability of an event according to the degree to which it is considered representative of, or similar to, some specific major characteristics of the process or population from which it originated.
- Probability judgment then becomes a matter of similarity judgment.

Example

- Which of the following sequence of coin tosses is more likely to occur?

1. HTTHTHTHHT
2. HHHHTTTTTT

- Did you think that sequence (1) is more likely to occur than sequence (2)?
- Most people would judge the first sequence to be more likely than the second one because they judged that the first one is a random sequence whereas the second one is not random and therefore has a smaller chance of occurrence.
- But the probability of occurrence for both sequences is 0.5^{10}

4. Unstated Assumptions

- A person may estimate probabilities that are conditioned on various unstated assumptions. Hence, the results do not properly reflect the total uncertainty.

Example

- In estimating the future revenues of a company, one may not have considered such possibilities as future price controls, major labor unrest, currency devaluation, terrorism, etc. The person may think that he is not responsible for considering these factors.

5. Coherence

- A person may assign a probability to an event based on the ease with which he can fabricate a plausible scenario that would lead to the occurrence of the event.
- An event is considered quite unlikely if no reasonable scenario can be found. On the other hand, it is considered more likely if many scenarios can be constructed that could make this event occur.

Example

- The probability that the sales would exceed a certain high volume may depend on how well the market researcher has put together scenarios that would lead to that volume.

7.3.3 Probability Assessment Protocols

- A *Probability Assessment Protocol* is a set of steps and procedures that an analyst should follow in interacting with and eliciting probabilities from an expert so that the results are as accurate as possible and biases are minimized.
- **The Stanford / SRI Assessment Protocol** comprises five steps:
 1. Motivating
 2. Structuring
 3. Conditioning
 4. Encoding
 5. Verifying
- Notice that many things need to be done before the actual encoding in Step 4.
- See Spetzler and von Holstein (1975) for details.
- Other protocols which are variations/extensions of the Stanford/SRI include:
 1. Wallsten /EPA Protocol (Wallsten and Whitefield, 1986).
 2. Morgan and Henrion Protocol (Morgan *et al.*, 1984).

7.4 Learning Probability Distributions from Data

- We may make use of data to access probability distribution.
- Some basic methods of using data for assessing probability distributions are discussed in this section.

7.4.1 Relative Frequencies Method

- The **relative frequency** of past events is often a good starting point to obtain an estimate of the probabilities of a discrete variable with few outcomes.

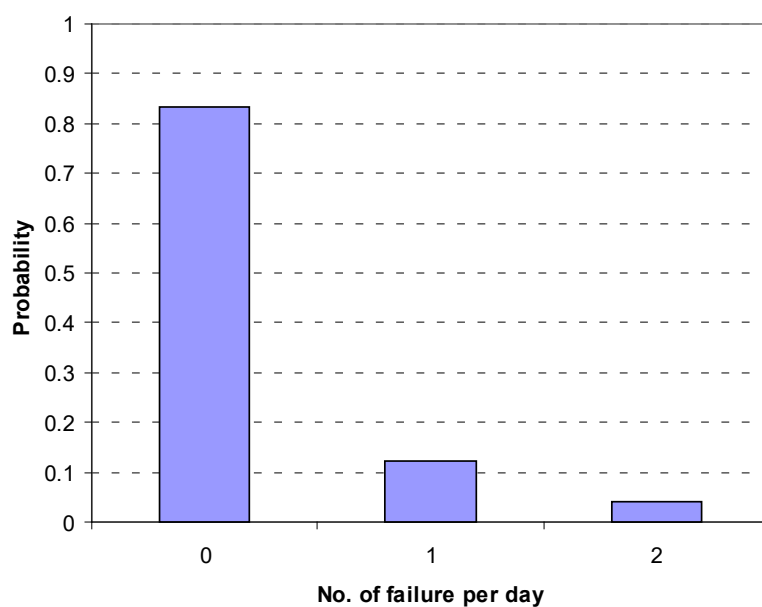
Example

- Let X = number of machine failures per day.
- The following data were collected over a period of 260 days:

No. of failures	No. of occurrences
0	217
1	32
2	11

- The probability mass function for X may be approximated by:

No. of Failures	Probability
0	$217/260 = 0.835$
1	$32/260 = 0.123$
2	$11/260 = 0.042$



7.4.2 Some Commonly Used Probability Distributions

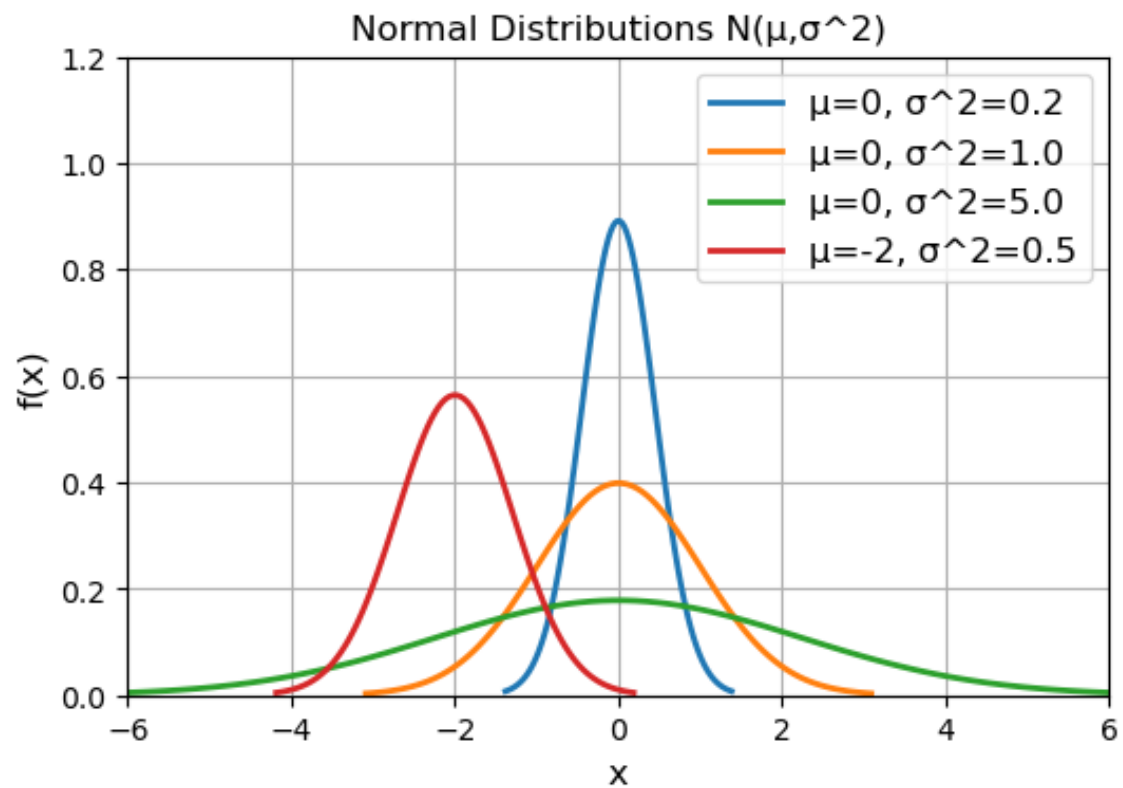
Normal Distribution: $N(\mu, \sigma^2)$

- The most commonly used distribution.

$$\text{PDF: } f(x; \mu, \sigma) = \frac{1}{\sigma\sqrt{2\pi}} e^{-1/2((x-\mu)^2/\sigma^2)} \text{ for } -\infty < x < \infty$$

μ = mean, σ = standard deviation,

location = μ , *scale* = σ

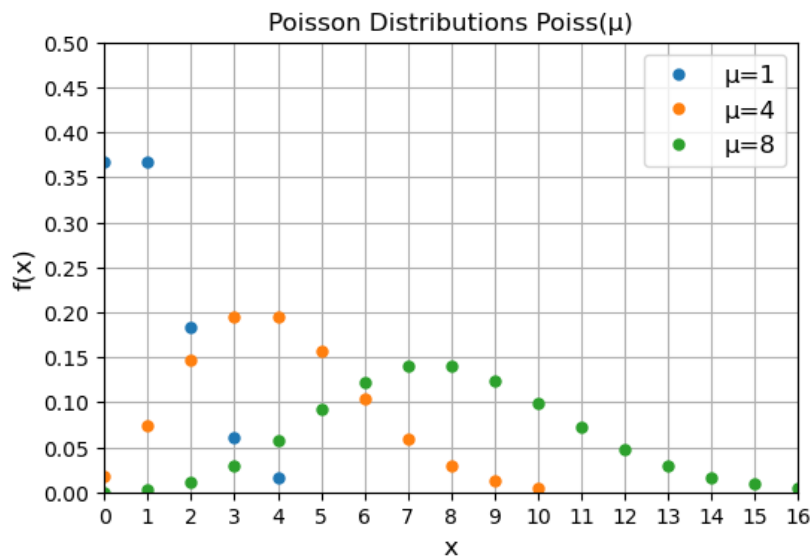


Poisson Distribution: $\text{Pois}(\mu)$

- The Poisson distribution is a **discrete distribution** for modeling the number of occurrences in a given time interval.

$$\text{PMF: } f(x; \mu) = \begin{cases} \frac{\mu^x e^{-\mu}}{x!} & \text{if } x = 1, 2, 3, \dots \\ 0 & \text{Otherwise} \end{cases}$$

μ = average per unit time, variance = μ , location = 0, scale = μ .

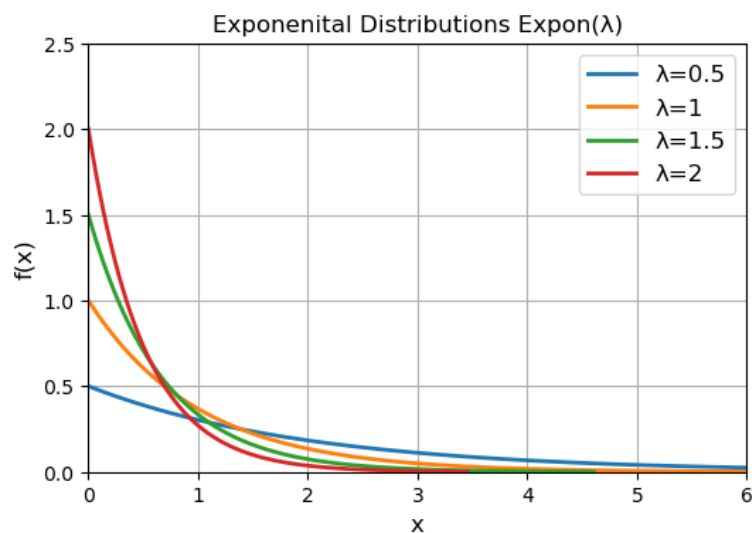


Exponential Distribution: $\text{Expon}(\lambda)$

- The exponential distribution is popular for modeling the inter-arrival time of a Poisson process.

$$\text{PDF: } f(x; \lambda) = \begin{cases} \lambda e^{-\lambda x} & x \geq 0 \\ 0 & x < 0 \end{cases}$$

where $\lambda > 0$ is the rate parameter, mean = $1/\lambda$, variance = $1/\lambda^2$, location = 0, scale = $1/\lambda$

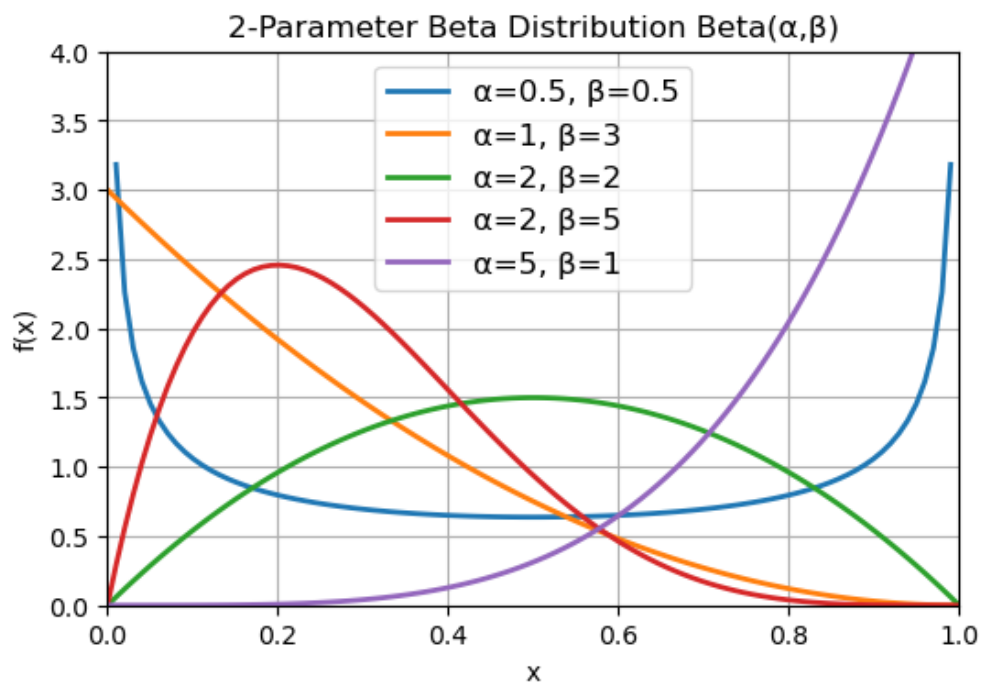


Standard (2-parameter) Beta Distribution: $\text{Beta}(\alpha, \beta)$.

- The standard 2-parameter Beta distribution is often used for modeling the uncertainty over the probability p of an event where $0 \leq p \leq 1$.

$$\text{PDF: } f(x; \alpha, \beta) = \begin{cases} \frac{x^{\alpha-1}(1-x)^{\beta-1}}{\int_0^1 t^{\alpha-1}(1-t)^{\beta-1} dt} & 0 \leq x \leq 1 \\ 0 & \text{Otherwise} \end{cases}$$

$\alpha > 0$ and $\beta > 0$ are shape parameters, mean = $\frac{\alpha}{\alpha + \beta}$, variance = $\frac{\alpha\beta}{(\alpha + \beta)^2(\alpha + \beta + 1)}$



General (4-parameter) Beta Distribution: Beta (α, β, L, H)

- The standard (2-parameter) Beta distribution is defined over a variable x where $0 \leq x \leq 1$.
- We can use the general (4-parameter) Beta distribution when $L \leq x \leq H$, by applying the linear transformation $y = x(H - L) + L$.
- Y follows Beta (α, β, L, H) if and only if X follows Beta (α, β) where $X = (Y - L)/(H - L)$.

$$\text{PDF: } f(y; \alpha, \beta, L, H) = \begin{cases} \frac{f(x; \alpha, \beta)}{(H - L)} & L \leq y \leq H \\ 0 & \text{Otherwise} \end{cases}$$

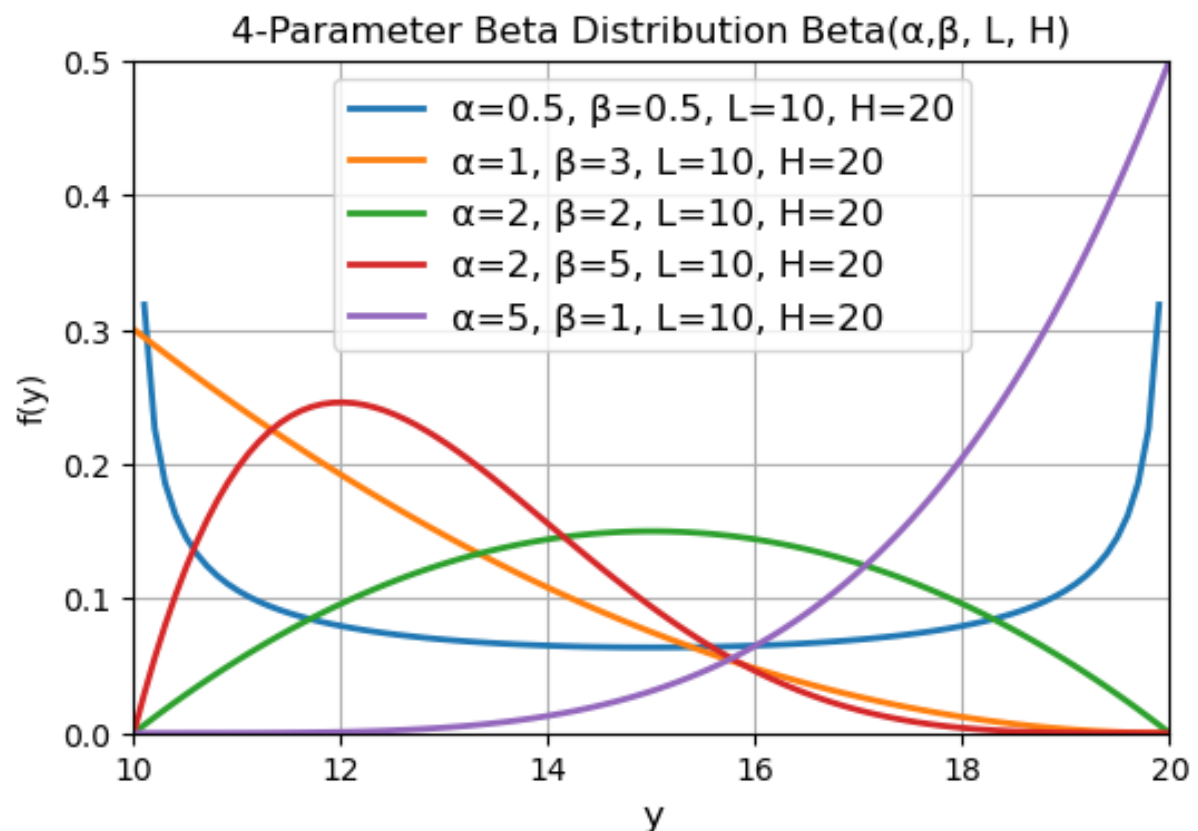
where $x = \frac{(y - L)}{(H - L)}$ and $\alpha > 0$ and $\beta > 0$ are shape parameters.

$$\text{mean} = \frac{\alpha H + \beta L}{\alpha + \beta}$$

$$\text{variance} = \frac{\alpha \beta (H - L)^2}{(\alpha + \beta)^2 (\alpha + \beta + 1)}$$

$$\text{location} = L$$

$$\text{scale} = (H - L)$$



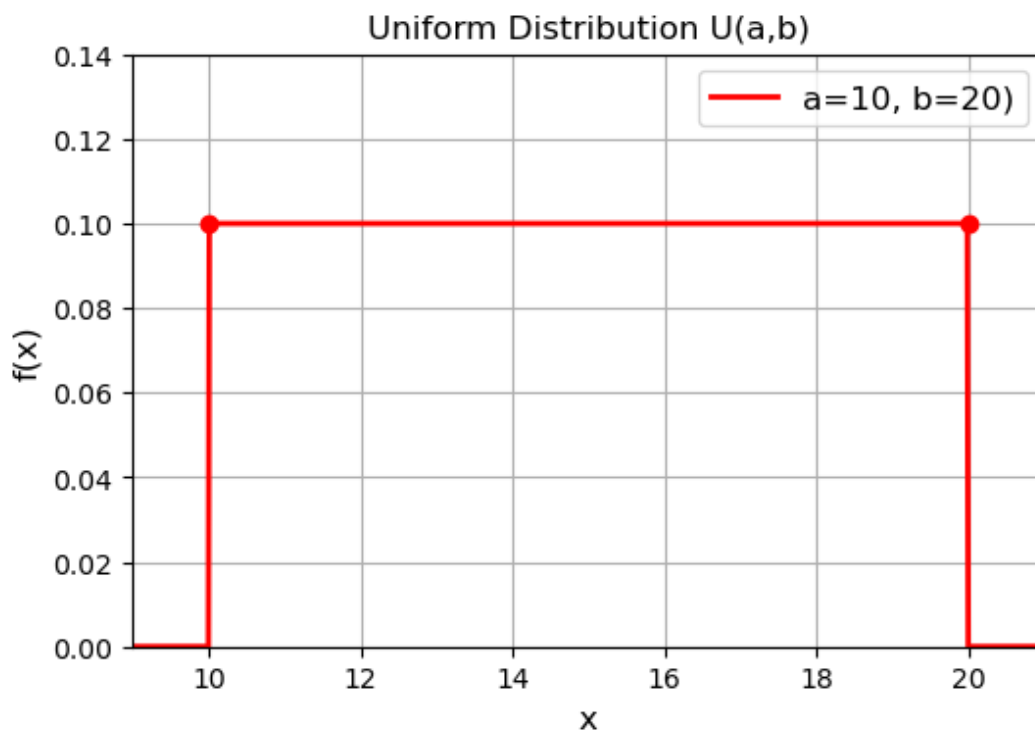
General Uniform Distribution: $U(a, b)$

- We can use the uniform distribution when we are totally uncertain about an event and have completely no other information about it.
- The uniform distribution represents the state of highest uncertainty over an event.

$$\text{PDF: } f(x) = \begin{cases} \frac{1}{b-a} & a \leq x \leq b \\ 0 & \text{otherwise} \end{cases}$$

$$\text{mean} = \frac{(a+b)}{2}, \text{variance} = \frac{(b-a)^2}{12}$$

$$\text{Location} = a, \text{Scale} = (b - a)$$



Triangular Distribution: Triangular (a, b, m)

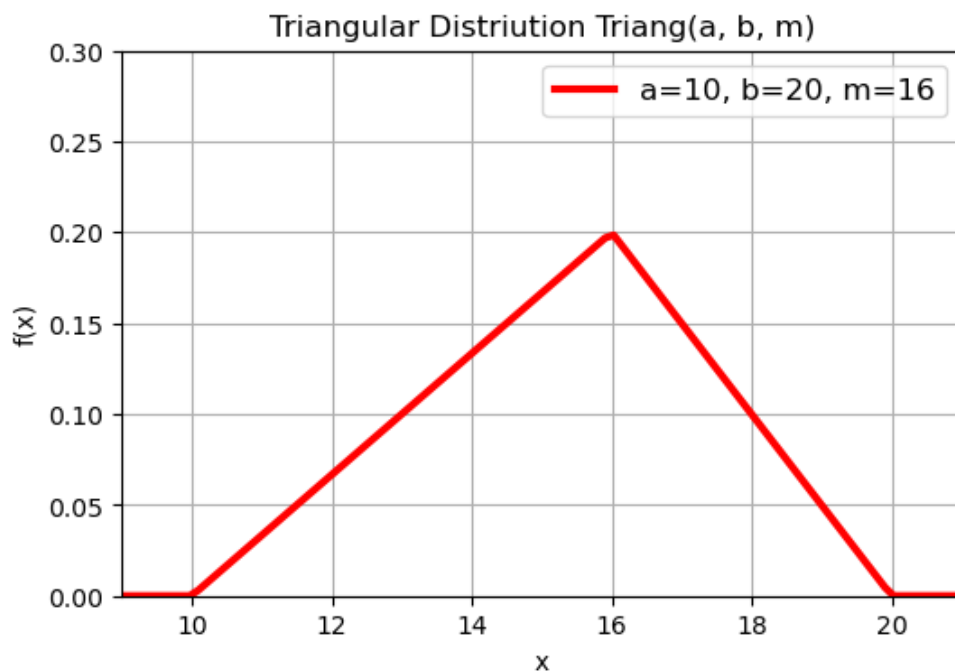
- The triangular distribution may be to approximate a variable that is known to be bounded between a and b , is asymmetrical, and has highest chance of occurrences around the mode m .

$$\text{PDF: } f(x) = \begin{cases} 0 & x < a \\ \frac{2(x-a)}{(b-a)(m-a)} & a \leq x < m \\ \frac{2}{b-a} & x = m \\ \frac{2(b-x)}{(b-a)(b-m)} & m < x \leq b \\ 0 & b < x \end{cases}$$

a = lower bound, b = upper bound, m = mode.

$$\text{mean} = \frac{(a+b+m)}{3}, \quad \text{variance} = \frac{a^2 + b^2 + m^2 - ab - am - bm}{18},$$

$$\text{Location} = a, \quad \text{Scale} = \frac{m-a}{b-a}$$



7.4.3 Fitting Distribution to Data

The Maximum Likelihood Estimation (MLE) Method

- This is a popular parameters estimation method and is easy to implement computationally.
- Let $\mathbf{D} = \{x_1, x_2, \dots, x_n\}$ be a set of observed data values. Suppose we want to fit data \mathbf{D} to a probability distribution with PDF or PMF $f(x|\boldsymbol{\theta})$ where $\boldsymbol{\theta}$ is a vector of unknown parameters.
- The **Likelihood Function** $L(\boldsymbol{\theta} | \mathbf{D})$ is the joint probability of observing all the x values in \mathbf{D} given that the distribution to be fitted has the parameter $\boldsymbol{\theta}$. That is,

$$L(\boldsymbol{\theta} | \mathbf{D}) = f(x_1, x_2, \dots, x_n | \boldsymbol{\theta})$$

- The MLE method finds the value of $\boldsymbol{\theta}$ that maximizes $L(\boldsymbol{\theta} | \mathbf{D})$, that is

$$\hat{\boldsymbol{\theta}} = \arg \text{Max}_{\boldsymbol{\theta}} f(x_1, x_2, \dots, x_n | \boldsymbol{\theta})$$

- If the data values are assumed to be mutually independent, then

$$\hat{\boldsymbol{\theta}} = \arg \text{Max}_{\boldsymbol{\theta}} \left(\prod_{i=1}^n f(x_i | \boldsymbol{\theta}) \right)$$

- It is usually more convenient to **Minimize the Negative Log-Likelihood Function** instead as $\log()$ is a monotonically increasing function. That is

$$\hat{\boldsymbol{\theta}} = \arg \text{Min}_{\boldsymbol{\theta}} \left(- \sum_{i=1}^n \log f(x_i | \boldsymbol{\theta}) \right)$$

- Some distributions (e.g. Normal, Poisson, Gamma) have known closed-form optimal solutions for $\hat{\boldsymbol{\theta}}$. Otherwise, numerical optimization algorithms are often used to find $\hat{\boldsymbol{\theta}}$.

Using Software Tools

1. Excel Add-in @Risk.

- This is a Monte-Carlo simulation software that can fit both continuous and discrete distributions.
- It can also fit data in CDF format. Hence you can use this to fit an Expert's CDF for a continuous variable.
- @Risk is part of the DecisionTools suite (<http://www.palisade.com>). Limited-time trial version available.
- It is also available in our ISEM Labs.

2. Python scipy.stats

- **Continuous distributions:** Use `scipy.stats.rv_continuous.fit()` to directly fit the data using MLE (default) or Method of Moments (MM). See Example 1 below.
- **Discrete distributions:** Not directly supported. You have to write your own negative log-likelihood function, and use `sci.optimize.minimize` or `.fmin` to find the optimal MLE parameters. See Example 2 below.

3. Other Software

- EasyFit
- XLSTAT
- ExpertFit from FlexSim
- etc

Example 1 (Continuous Distributions)

- Fit a continuous distribution to the following data set (100 observations):

85.32	90.19	97.99	63.05	80.23
104.29	86.77	96.16	78.84	85.91
115.94	91.71	109.57	117.55	72.44
109.09	98.32	115.37	111.79	82.21
81.53	71.96	93.88	57.67	120.63
111.32	90.65	124.03	88.12	119.95
119.77	107.88	66.28	120.57	114.66
80.53	75.42	85.22	86.02	87.76
124.24	65.11	73.07	113.98	96.42
111.32	120.26	102.45	101.88	125.66
97.82	107.66	113.26	98.08	90.43
97.57	69.63	121.83	108.96	69.67
106.31	94.73	103.55	87.3	122.82
108.43	137.49	85.11	116.66	108.18
88.99	78.15	102.86	121.46	67.46
96.45	109.93	84.63	97.3	139.09
108.82	83.32	75.41	103.55	91.8
90.18	106.99	117.31	96.88	57.2
116.53	98.8	75.79	126.13	107
76.55	122.15	66.82	96.63	96.49

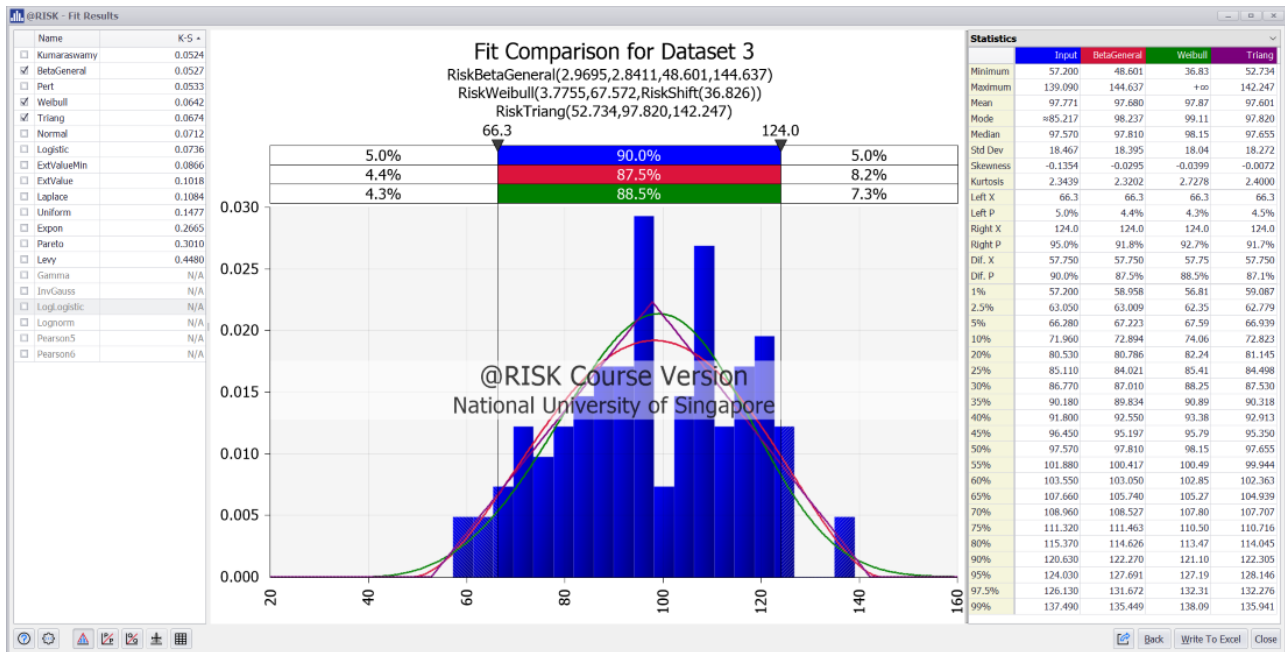
Using Excel with @Risk:

The screenshot shows an Excel spreadsheet with the @Risk add-in installed. The data from the previous table is entered into cells A4:L23. The @Risk ribbon is active, showing the 'Fit' button. The 'Fit Distributions to Data' dialog box is open, with the following settings:

- Data Set:** Name: example 1, Range: =A4:A23, Values are Dates: ☐
- Data Type:** Continuous Sample Data (selected), Discrete Sample Data, Discrete Sample Data (Counted Format), Density (X,Y) Points (Unnormalized), Density (X,Y) Points (Normalized), Cumulative (X,P) Points
- Filter:** Type: None

The dialog box has OK and Cancel buttons at the bottom right.

Results:



Selected distributions fitted sorted by KS_statistics.

1. Beta (2.9695, 2.8411, 48.601, 144.637)
2. Weibull (3.7755, 67.572, Shift(36.826))
3. Triang (52.734, 97.820, 142.247)
4. Normal (97.771, 18.467)
5. Logistic(98.126, 10.848)
6. Laplace (97.6950, 21.5416)

Using Python `scipy.stats.rv_continuous.fit` function

In [1]: `""" Fit Continuous Distributions to Data """`

```
import pandas as pd
import numpy as np
from scipy import stats
import matplotlib.pyplot as plt
```

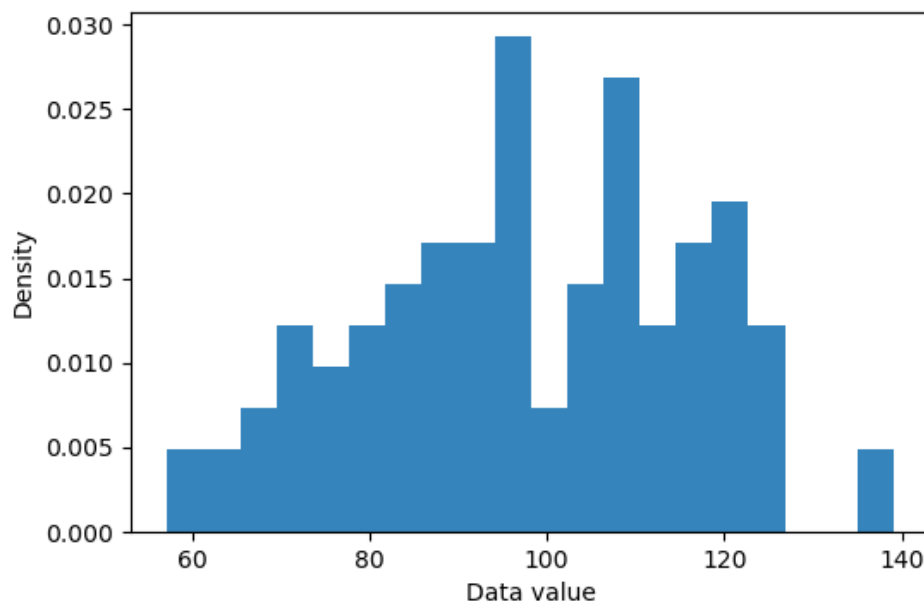
In [2]: `# Read the data from Excel file`

```
data = pd.read_excel("7.4.3_data.xlsx", sheet_name="data1", header=None)
data = data.values.flatten()
```

In [3]: `# Visualize the data with a density histogram and display description`

```
bins = 20
fig0, ax0 = plt.subplots(dpi=100)
ax0.hist(data, bins=bins, histtype='stepfilled', density=True, alpha=0.9)
ax0.set_xlabel("Data value")
ax0.set_ylabel("Density")
plt.show()

desc = stats.describe(data)
print("\nData Description:")
print(f"  size = {desc.nobs}")
print(f"  minmax = {desc.minmax}")
print(f"  mean = {desc.mean}")
print(f"  var = {desc.variance}")
print(f"  skewness = {desc.skewness}")
print(f"  kurtosis= {desc.kurtosis}")
```



Data Description:

```
size = 100
minmax = (57.2, 139.09)
mean = 97.77139999999997
var = 341.0240162020202
skewness = -0.13339778928348767
kurtosis= -0.6831564233178393
```



```
In [4]: # List of distributions to fit
Dists = ['beta','expon','gamma','lognorm','logistic', 'laplace',
         'norm', 'rayleigh','triang','uniform','weibull_min']
```

```
In [5]: # Loop through the list of distributions
res = {}
for d in Dists:
    # Fit distribution to data
    dist = getattr(stats, d)
    print(f"Fitting distribution {d}:")
    params_fit = dist.fit(data)
    # Perform the Kolmogorov-Smirnov test on fitted result.
    Dv, pv = stats.kstest(data, d, args=params_fit)
    # Keep the results
    res[d] = {'Params': params_fit,
             'KS_D' : Dv,
             'KS_pv' : pv }

# Sort the results by KS stats
results = {dist: vals for dist, vals in
           sorted(res.items(), key=lambda x: x[1]['KS_D']) }
```

Fitting distribution beta:

Parameters = (2.9694722641451423, 2.8410772308999013, 48.6010183977518, 96.0358209061009)

Fitting distribution expon:

Parameters = (57.2, 40.571399999999997)

Fitting distribution gamma:

Parameters = (535.8301275279623, -328.86401969628605, 0.7962201909442794)

Fitting distribution lognorm:

Parameters = (0.011160563881933781, -1547.1406362357357, 1644.8135449430042)

Fitting distribution logistic:

Parameters = (98.12626261718812, 10.847811913084668)

Fitting distribution laplace:

Parameters = (97.695, 15.232199999999999)

Fitting distribution norm:

Parameters = (97.771399999999997, 18.374269401529958)

Fitting distribution rayleigh:

Parameters = (55.45141355800718, 32.623572984043726)

Fitting distribution triang:

Parameters = (0.5036780338389795, 52.73408519584795, 89.51336326076714)

Fitting distribution uniform:

Parameters = (57.2, 81.89)

Fitting distribution weibull_min:

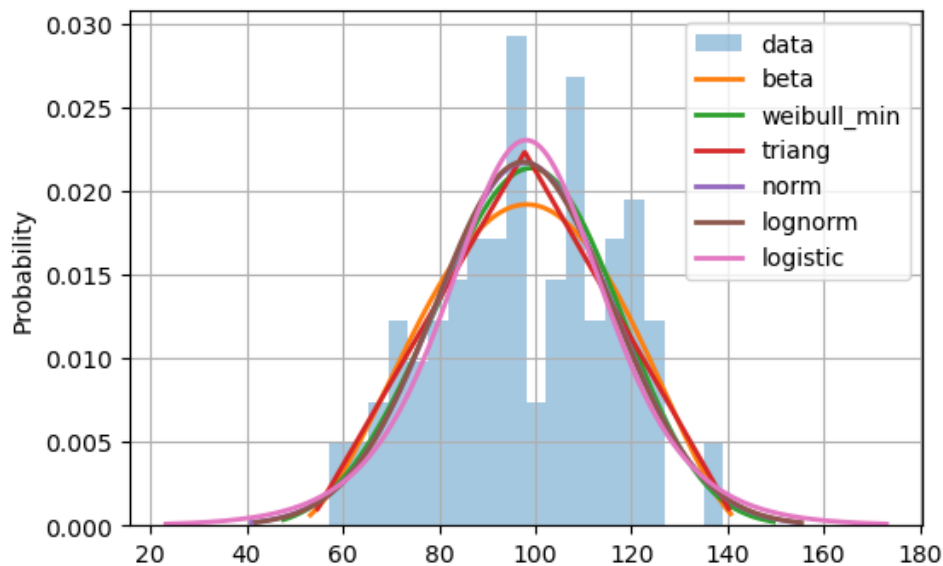
Parameters = (3.7755301524300005, 36.82589064297723, 67.57169335277156)

```

In [6]: # Compare the PDF of the fitted distributions with the data
max_to_plot = 6
num_to_plot = min(max_to_plot, len(results))

fig1, ax1 = plt.subplots(dpi=100)
ax1.hist(data, bins=bins, histtype='stepfilled', label='data',
         density=True, alpha=0.4)
ax1.set_ylabel("Probability")
for d in list(results.keys())[:num_to_plot]:
    params = results[d]['Params']
    dist = getattr(stats, d)
    x = np.linspace(dist.ppf(0.001, *params),
                    dist.ppf(0.999, *params), 500)
    pdf_fitted = dist.pdf(x, *params)
    ax1.plot(x, pdf_fitted, lw=2, label=d)
ax1.legend()
ax1.grid('dotted')
plt.show()

```

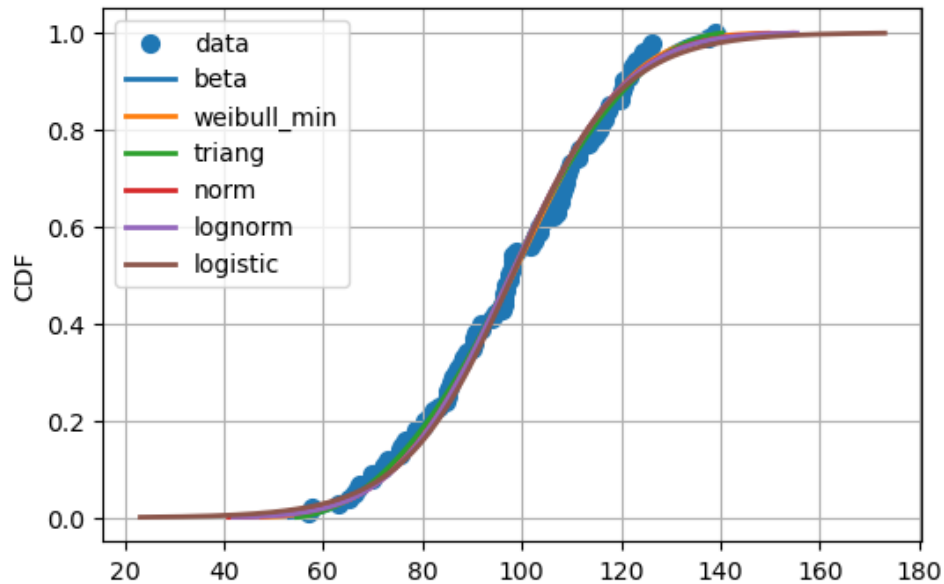


```

In [7]: # Compare the CDF of the fitted distributions with the data ECDF
def ecdf(data):
    # Compute ECDF of the data
    x = np.sort(data)
    n = x.size
    y = np.arange(1,n+1)/n
    return(x,y)

fig2, ax2 = plt.subplots(dpi=100)
x, y = ecdf(data)
ax2.scatter(x, y, lw=2, label='data' )
ax2.set_ylabel("CDF")
for d in list(results.keys())[:num_to_plot]:
    params = results[d]['Params']
    dist = getattr(stats, d)
    x = np.linspace(dist.ppf(0.001, *params),
                    dist.ppf(0.999, *params), 500)
    cdf_fitted = dist.cdf(x, *params)
    ax2.plot(x, cdf_fitted, lw=2, label=d)
ax2.legend()
ax2.grid('dotted')
plt.show()

```



```
In [8]: # Show the top fitted parameters
num_to_show = 6
num_to_show = min(num_to_show, len(results))
print(f"\nThe top {num_to_show} distributions:")

for i, (d, vals) in enumerate(list(results.items())[:num_to_show]):
    dist = getattr(stats, d)
    params = vals['Params']
    print(f"\nDistribution {i+1}: {d}")
    print(f"    KS stats = {vals['KS_D']}")
    pv = vals['KS_pv']
    print(f"    p-value = {pv}", end=" ")
    if pv < 0.05:
        print(" < 0.05")
    else:
        print("")
    print(f"    mean = {dist(*params).mean()}")
    print(f"    var = {dist(*params).var()}")
    print(f"    sd = {dist(*params).std()}")
```

The top 6 distributions:

Distribution 1: beta

```
Parameters = (2.9694722641451423, 2.8410772308999013, 48.6010183977518,
96.0358209061009)
KS stats = 0.0526932223744947
p-value = 0.9302200495921993
mean = 97.67997500732218
var = 338.3858977910069
sd = 18.39526835332953
```

Distribution 2: weibull_min

```
Parameters = (3.7755301524300005, 36.82589064297723, 67.57169335277156)
KS stats = 0.06422909438028257
p-value = 0.779443120169669
mean = 97.87435896417409
var = 325.58862495960005
sd = 18.04407451102993
```

```

Distribution 3: triang
Parameters = (0.5036780338389795, 52.73408519584795, 89.51336326076714)
KS stats = 0.06741904575124746
p-value = 0.7277895637951273
mean = 97.6005112192695
var = 333.8661136764873
sd = 18.272003548502482

Distribution 4: norm
Parameters = (97.77139999999997, 18.374269401529958)
KS stats = 0.07206510260645305
p-value = 0.6498079161282216
mean = 97.77139999999997
var = 337.61377604000006
sd = 18.374269401529958

Distribution 5: lognorm
Parameters = (0.011160563881933781, -1547.1406362357357,
1644.8135449430042)
KS stats = 0.07361252519719796
p-value = 0.6236773818384749
mean = 97.77534939305906
var = 337.0441281677288
sd = 18.358761618576803

Distribution 6: logistic
Parameters = (98.12626261718812, 10.847811913084668)
KS stats = 0.07362021344400171
p-value = 0.6235477725256537
mean = 98.12626261718812
var = 387.1353092921242
sd = 19.675754351285345

```

Example 2 (Discrete Distributions)

- Fit a distribution to the following 100 values observed from a discrete counting event.

12	11	10	15	3
7	4	11	8	4
4	7	6	9	6
6	13	10	7	5
9	8	5	9	7
9	10	10	10	13
4	6	11	5	5
13	9	6	15	10
7	7	9	15	8
6	9	12	6	4
11	7	8	8	8
7	6	3	7	6
8	6	7	9	8
10	7	11	12	8
7	7	3	7	7
5	11	8	11	6
8	13	11	7	6
13	7	7	17	5
4	10	5	8	10
6	6	11	9	11

Using Excel with @Risk

7.4.3_Fit_Discrete_Distribution_@Risk.xlsx - Excel

Poh Kim Leng

File Home Insert Draw Page Layout Formulas Data Review View Developer Add-ins Help Acrobat Log-hub @RISK Tell me Share

Iterations: 1000
Simulations: 1

Define Simulation Results

A1 12

	A	B	C	D	E
1	12	11	10	15	3
2	7	4	11	8	4
3	4	7	6	9	6
4	6	13	10	7	5
5	9	8	5	9	7
6	9	10	10	10	13
7	4	6	11	5	5
8	13	9	6	15	10
9	7	7	9	15	8
10	6	9	12	6	4
11	11	7	8	8	8
12	7	6	3	7	6
13	8	6	7	9	8
14	10	7	11	12	8
15	7	7	3	7	7
16	5	11	8	11	6
17	8	13	11	7	6
18	13	7	7	17	5
19	4	10	5	8	10
20	6	6	11	9	11

example 2

Ready Average: 8.1

@RISK - Fit Distributions to Data

Data Distributions Bootstrap Chi-Sq Binning Results

Data Set

Name: Dataset 2

Range: =A1:E20

☐ Values are Dates

Data Type

☐ Continuous Sample Data

☒ Discrete Sample Data

☐ Discrete Sample Data (Counted Format)

☐ Density (X,Y) Points (Unnormalized)

☐ Density (X,Y) Points (Normalized)

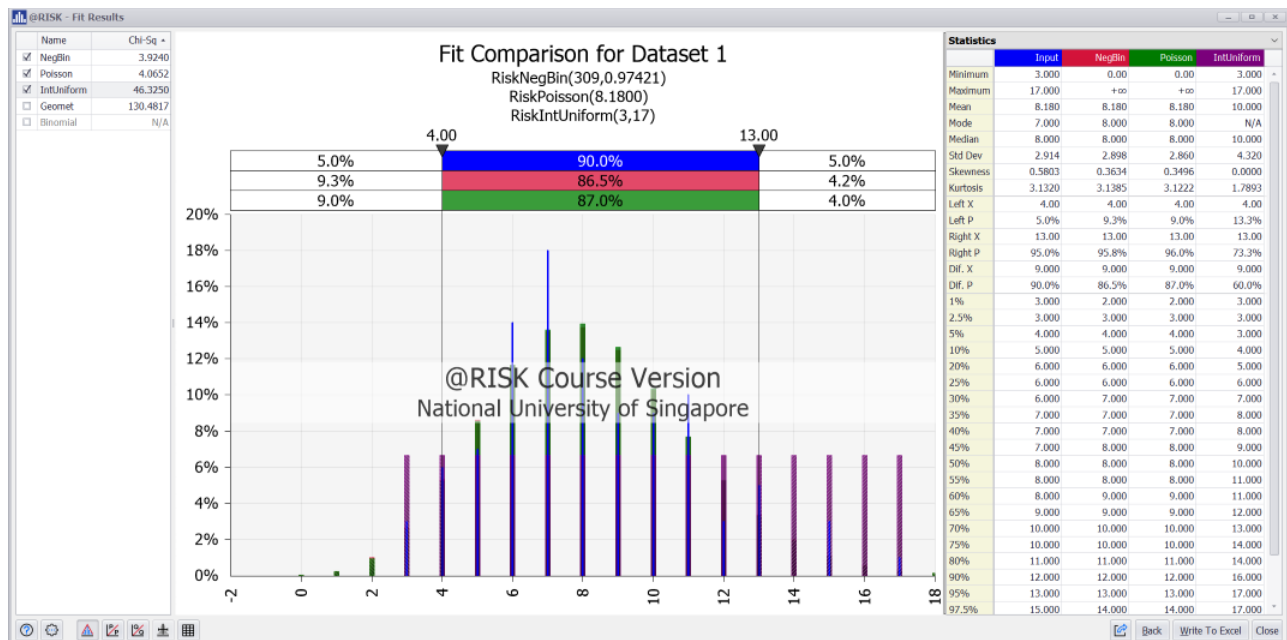
☐ Cumulative (X,P) Points

Filter

Type: None

OK Cancel

Results:



Fitted Discrete Distributions:

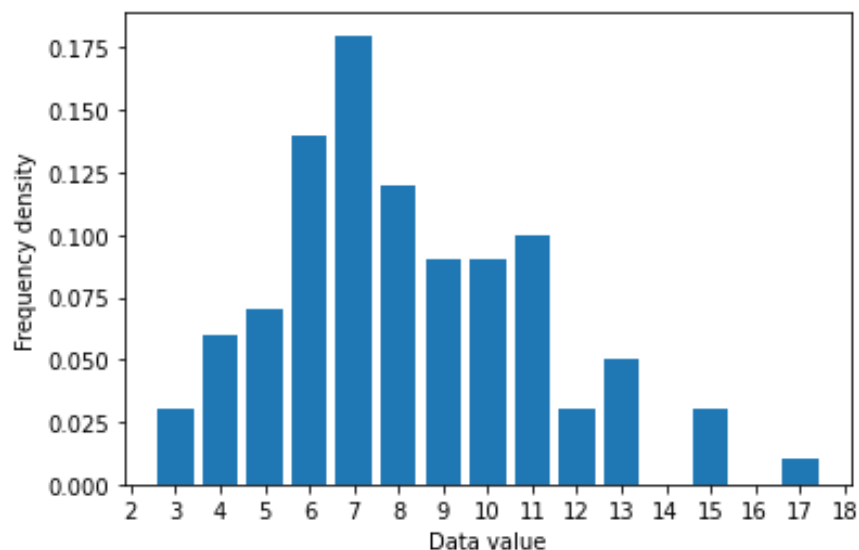
1. Poisson (8.1800)
2. NegBin (309, 0.97421)
3. IntUniform(3, 17) // chi-sq score very high
4. Geomet(0.10893) // chi-sq score very high

Python:

```
In [1]: """ Fit Discrete Distributions to Data"""  
import pandas as pd  
import numpy as np  
from scipy import stats  
from scipy.optimize import minimize  
import matplotlib.pyplot as plt
```

```
In [2]: # Read the data from Excel file  
data = pd.read_excel ("7.4.3_data.xlsx", sheet_name="data2", header=None)  
data = data.values.flatten()
```

```
In [3]: # Plot data and show description  
fig0, ax0 = plt.subplots(dpi=100)  
values, counts = np.unique(data, return_counts=True)  
ax0.bar(values, counts/counts.sum())  
ax0.set_xticks(np.arange(values.min()-1, values.max()+2))  
ax0.set_xlabel("Data value")  
ax0.set_ylabel("Frequency density")  
ax0.grid('dotted')  
plt.show()  
  
# Display description of the data  
desc = stats.describe(data)  
print("\nData Description:")  
print(f"  size = {desc.nobs}")  
print(f"  minmax = {desc.minmax}")  
print(f"  mean = {desc.mean}")  
print(f"  var = {desc.variance}")  
print(f"  skewness = {desc.skewness}")  
print(f"  kurtosis= {desc.kurtosis}")
```



```
Data Description:  
size = 100  
minmax = (3, 17)  
mean = 8.18  
var = 8.492525252525253  
skewness = 0.5715704116345816  
kurtosis= 0.0660478548040615
```

```
In [4]: # Distributions to fit and their parameter initial guess
# 'poisson' # Poisson (mu)
# 'binom' # Binomial (n, p)
# 'nbinom' # Negative Binomial (n, p)
# 'betabinom' # Beta-Binomial (n, a, b)
# 'randint' # randint (a, b)
# 'geom' # geom(p)

Dists = { 'poisson' : (10, ),
          'binom' : (20, 0.5),
          'nbinom' : (20, 0.5),
          'betabinom': (20, 10, 10),
          'randint' : (3, 17),
          'geom' : (0.5, )}
```

```
In [5]: # Negative Log-Likelihood function to minimize
def nLogLikelihood(params, dist, data):
    return -dist.logpmf(data, *params).sum()
```

```
In [6]: # Dict to keep successful results as we loop through the distributions
res = {}
for d, x0 in Dists.items():
    dist = getattr(stats, d)
    # Perform MLE Optimization
    sol = minimize(nLogLikelihood, x0=x0, args=(dist, data),
                  method='Nelder-Mead',
                  options={'maxiter':50000,'xatol':1E-8,'fatol':1E-8})
    # method='Powell',
    # options={'maxiter':10000,'xtol':1E-8,'ftol':1E-8})
    print(f"\nFitting Distribution {d}:")
    print(f" MLE {sol.message}")
    params_fit = sol.x
    print(f" Fitted Parameters = {params_fit}")
    # Keep the results if successful
    if sol.success:
        # Perform Kolmogorov-Smirnov test and store result
        Dv, pv = stats.kstest(data, d, args=params_fit)
        res[d] = {'Params' : params_fit,
                  'KS_D' : Dv,
                  'KS_pv': pv }

print(f"\nNumber of distributions fitted = {len(res)}")

# Sort the results by KS_stats
results = {dist: vals for dist, vals in
           sorted(res.items(), key=lambda x: x[1]['KS_D']) }
```

```
Fitting Distribution poisson:
MLE Optimization terminated successfully.
Fitted Parameters = [8.17999996]
```

```
Fitting Distribution binom:
MLE Optimization terminated successfully.
Fitted Parameters = [33.      0.24375]
```

```
Fitting Distribution nbinom:
MLE Optimization terminated successfully.
Fitted Parameters = [308.57602061  0.97417571]
```

```

Fitting Distribution betabinom:
  MLE Optimization terminated successfully.
  Fitted Parameters = [18.  10.5 10.5]

Fitting Distribution randint:
  MLE Maximum number of iterations has been exceeded.
  Fitted Parameters = [ 3. 17.]

Fitting Distribution geom:
  MLE Optimization terminated successfully.
  Fitted Parameters = [0.12224939]

Number of distributions fitted = 5

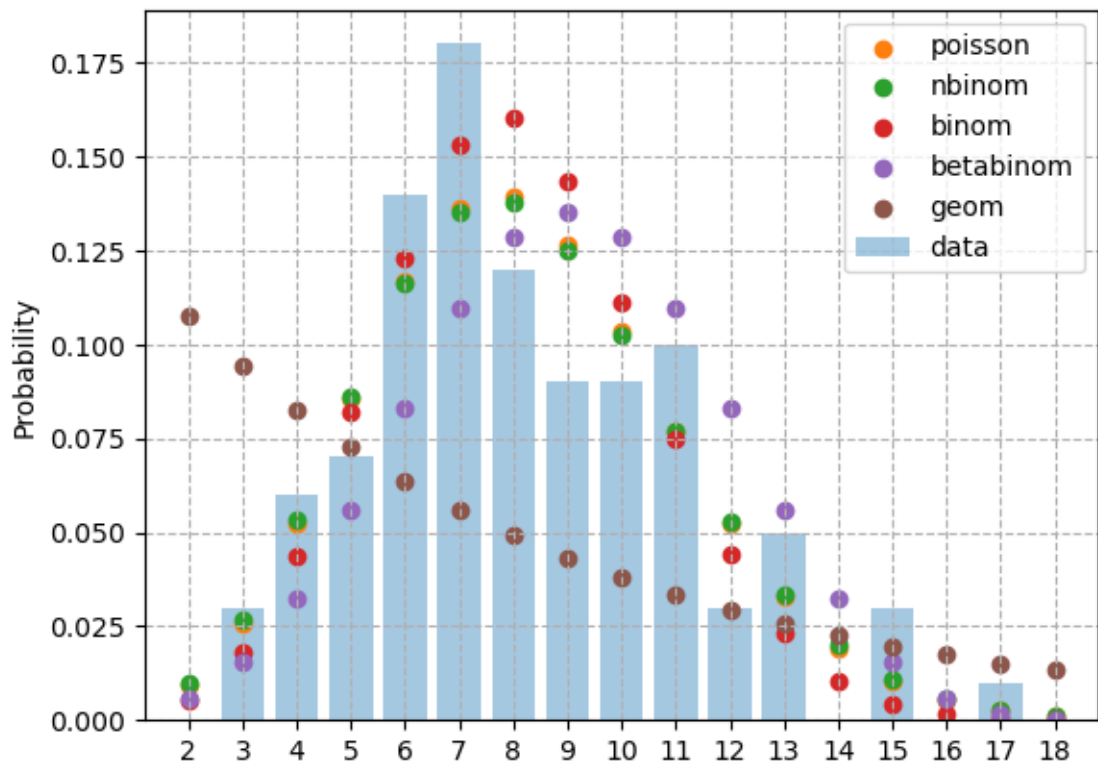
```

```

In [7]: # Plot PMF of fitted distributions and compare them with the data
num_to_plot = 5
num_to_plot = min(num_to_plot, len(results))

fig1, ax1 = plt.subplots(dpi=100)
ax1.bar(values, counts/counts.sum(), label='data', alpha=0.4)
ax1.set_ylabel("Probability")
x = np.arange(values.min()-1, values.max()+2)
for d in list(results.keys())[:num_to_plot]:
    dist = getattr(stats, d)
    params_fit = results[d]['Params']
    ax1.scatter(x, dist.pmf(x, *params_fit), label=d)
ax1.set_xticks(x)
ax1.legend()
ax1.grid(ls='--')
plt.show()

```

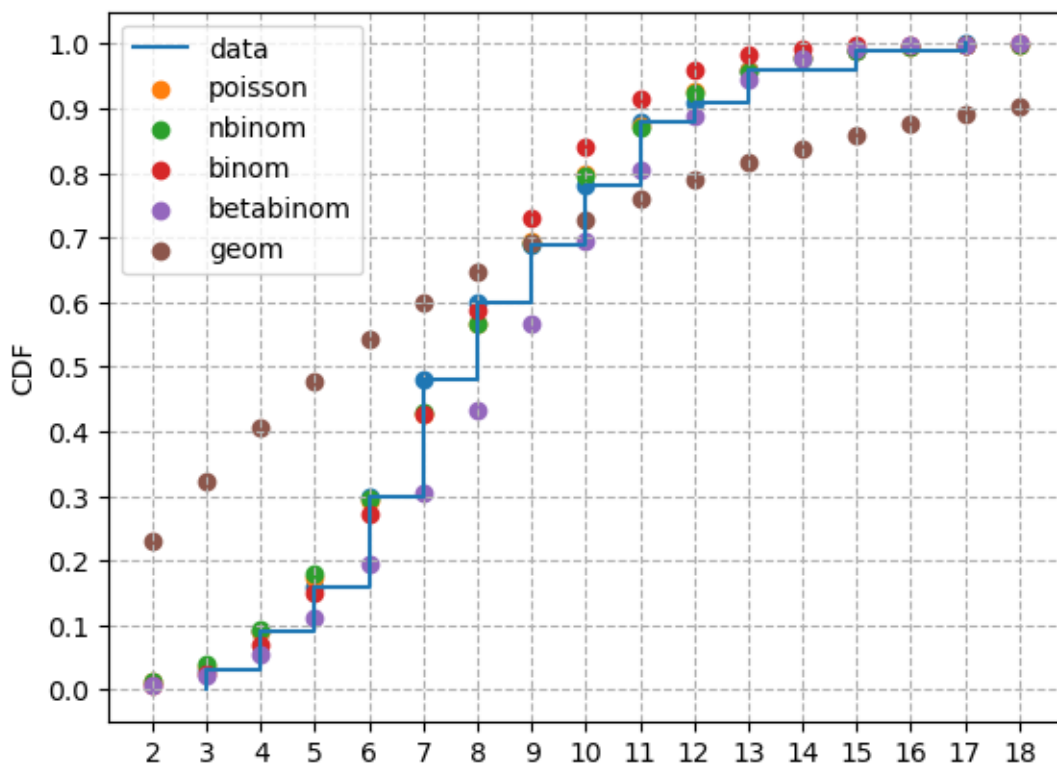



```

In [8]: # Plot CDF of fitted distributions and compare them with the data
def step_func(x, y):
    ''' Compute coordinates to line plot x and y as step function '''
    xval = []
    for item in x:
        xval.append(item)
        xval.append(item)
    yval = [0]
    for item in y[:-1]:
        yval.append(item)
        yval.append(item)
    yval.append(1)
    return (xval, yval)

fig2, ax2 = plt.subplots(dpi=100)
cdf_data = np.cumsum(counts/counts.sum())
ax2.plot(*step_func(values, cdf_data), label='data' )
ax2.scatter(values , cdf_data, label=None)
x = np.arange(values.min()-1, values.max()+2)
for d in list(results.keys())[:num_to_plot]:
    dist = getattr(stats, d)
    params_fit = results[d]['Params']
    ax2.scatter(x, dist.cdf(x, *params_fit), label=d)
ax2.set_xticks(x)
ax2.set_yticks(np.linspace(0,1,11))
ax2.set_ylabel('CDF')
ax2.legend()
ax2.grid(ls='--')
plt.show()

```



```
In [9]: # Show the top fitted parameters
num_to_show = 5
num_to_show = min(num_to_show, len(results))
print(f"\nThe top {num_to_show} distributions:")

for i, (d, vals) in enumerate(list(results.items())[:num_to_show]):
    dist = getattr(stats, d)
    params = vals['Params']
    print(f"\nDistribution {i+1}: {d}")
    print(f"  Params = {vals['Params']}")
    print(f"  KS_stats = {vals['KS_D']}")
    pv = vals['KS_pv']
    print(f"  p-value = {pv}", end=" ")
    if pv < 0.05:
        print("< 0.05")
    else:
        print("")
    print(f"  mean = {dist(*params).mean()}")
    print(f"  var = {dist(*params).var()}")
    print(f"  sd = {dist(*params).std()}")
```

The top 6 distributions:

Distribution 1: poisson

```
Params = [8.17999996]
KS_stats = 0.13188736268091658
p-value = 0.05614609245552149
mean = 8.179999962449074
var = 8.179999962449074
sd = 2.8600699226503314
```

Distribution 2: nbinom

```
Params = [308.57602061  0.97417571]
KS_stats = 0.13520999427877164
p-value = 0.04686151753816992 < 0.05
mean = 8.179999455336235
var = 8.396841940232385
sd = 2.897730480950978
```

Distribution 3: binom

```
Params = [33.  0.24375]
KS_stats = 0.150837808663566
p-value = 0.018845645860325555 < 0.05
mean = 8.043750000000006
var = 6.0830859375000035
sd = 2.466391278264664
```

Distribution 4: betabinom

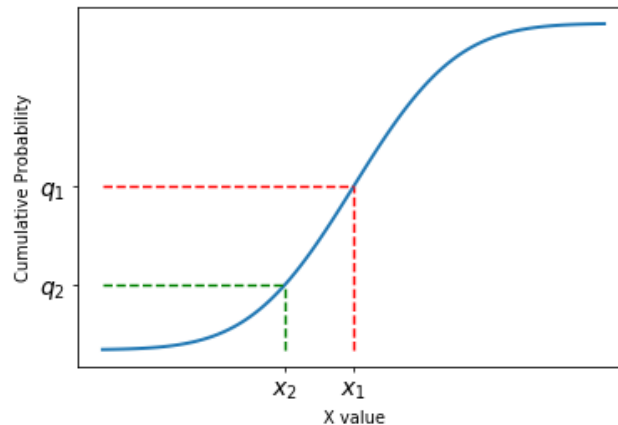
```
Params = [18.  10.5 10.5]
KS_stats = 0.17607889576275854
p-value = 0.003498486645988952 < 0.05
mean = 9.0
var = 7.9772727272727275
sd = 2.824406615073815
```

Distribution 5: geom

```
Params = [0.12224939]
KS_stats = 0.3889784302638508
p-value = 3.9009361387286217e-14 < 0.05
mean = 8.18000000390055
var = 58.73240005991246
sd = 7.663706678880166
```

7.4.4 Fitting Normal Distribution with Two Known Percentile Values

- Suppose an uncertain variable X follows the Normal distribution with two known percentile values as shown in the following CDF:



- That is, given $X \sim N(\mu, \sigma^2)$, and

$$F(x_1; \mu, \sigma) = q_1 \quad (1)$$

$$F(x_2; \mu, \sigma) = q_2 \quad (2)$$

$$\text{where } F(x) = \frac{1}{2} \left[1 + \operatorname{erf} \left(\frac{x - \mu}{\sigma \sqrt{2}} \right) \right] \quad \text{and} \quad \operatorname{erf}(x) = \frac{2}{\sqrt{2\pi}} \int_{-\infty}^x e^{-t^2} dt$$

- How do we find the two parameters μ and σ ?

Case 1: When the mean of the distribution is known.

- Consider the easy case when x_1 is the mean, i.e., $\mu = x_1$ and $q_1 = 0.5$
- We only need to find σ by solving equation (2):

$$F(x_2) = \frac{1}{2} \left[1 + \operatorname{erf} \left(\frac{x_2 - \mu}{\sigma \sqrt{2}} \right) \right] = q_2$$

- Therefore $\sigma = \frac{x_2 - \mu}{z_2}$

where $z_2 = \sqrt{2} \operatorname{erf}^{-1}(2q_2 - 1)$ is the inverse CDF value of the standard normal distribution $N(0,1)$ at percentile q_2 .

- The value of z_2 can be computed using:
 - Python: `z2 = scipy.stats.norm.ppf(q2)`
 - Excel: `=NORM.S.INV(q2)` or `=NORM.INV(q2, 0, 1)`
 - R: `z2 <- qnorm(q2, mean=0, sd=1)`

Example

- It is observed that the mean of a variable X is 200, and 20% of the values of X is less than or equal to 100. If X follows the normal distribution, what is its standard deviation?

We have $x_1 = 200, q_1 = 0.5$
 $x_2 = 100, q_2 = 0.2$

$$z_2 = F^{-1}(0.2) = -0.84162$$

$$\sigma = (100 - 200) / (-0.84162) = 118.818$$

- Hence $X \sim N(200, 118.818^2)$

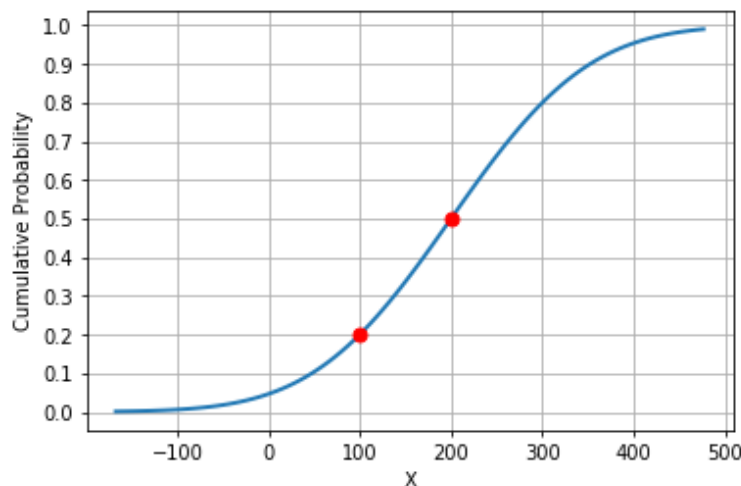
Python:

```
In [1]: """ Fit Normal Distribution with known mean and 1 percentile value """  
from scipy.stats import norm
```

```
In [2]: x1, q1 = 200, 0.5  
x2, q2 = 100, 0.2  
mu = x1  
z2 = norm.ppf(q2)  
sigma = (x2 - mu)/z2  
print(f"mu = {mu}, sigma = {sigma}")
```

mu = 200, sigma = 118.81829498938903

```
In [3]: import matplotlib.pyplot as plt  
import numpy as np  
xmin = norm.ppf(0.001, loc=mu, scale=sigma)  
xmax = norm.ppf(0.999, loc=mu, scale=sigma)  
x = np.linspace(xmin, xmax, 100)  
y = norm.cdf(x, mu, sigma)  
fig, ax = plt.subplots()  
ax.plot(x, y, lw=2)  
ax.plot([x1, x2], [q1, q2], 'ro', ms=7)  
ax.set_yticks(np.linspace(0, 1, 11))  
ax.set_ylabel("Cumulative Probability")  
ax.set_xlabel("X")  
ax.grid()  
plt.show()
```



Case 2 (When the mean is unknown)

- Consider the case when neither x_1 nor x_2 is the mean. In this case, we need to solve two equations with two unknowns μ and σ :

$$\frac{1}{2} \left[1 + \operatorname{erf} \left(\frac{x_1 - \mu}{\sigma \sqrt{2}} \right) \right] = q_1 \quad \text{and} \quad \frac{1}{2} \left[1 + \operatorname{erf} \left(\frac{x_2 - \mu}{\sigma \sqrt{2}} \right) \right] = q_2$$

Example:

- An uncertain variable follows the normal distribution. It is observed that 10% of its values are greater than or equal to 352.27 and 20% of its values are less than or equal to 100. What is the mean and standard deviation of X ?
- Given $x_1 = 352.27$ $q_1 = 1 - 0.1 = 0.9$
 $x_2 = 100$ $q_2 = 0.2$
- Solving the 2 equations, we obtain mean = 200.00, standard deviation = 118.82.

Python:

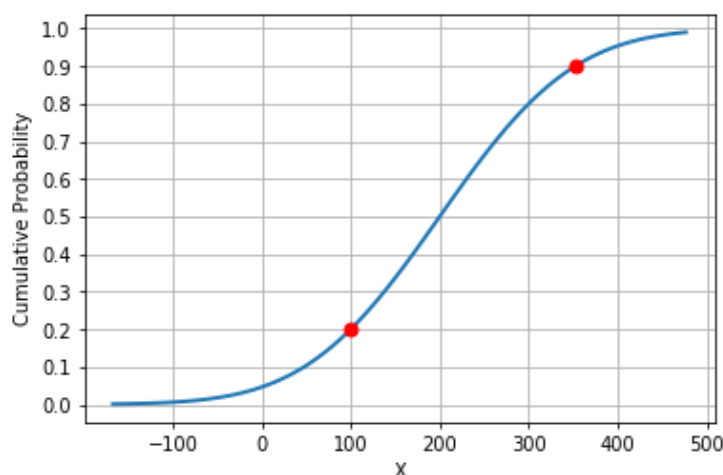
```
In [1]: """ Fit Normal Distribution with 2 known percentile values """
from scipy.stats import norm
from scipy.optimize import root
```

```
In [2]: # Equations to solve
eq = lambda param, x, q: (norm.cdf(x[0], *param) - q[0],
                          norm.cdf(x[1], *param) - q[1])
```

```
In [3]: x1, q1 = 352.27, 0.9
x2, q2 = 100, 0.2
guess = ((x1+x2)/2, abs(x1-x2)/2)
sol = root(eq, guess, args=((x1, x2), (q1, q2)))
print(sol.message)
mu, sigma = sol.x
print(f"mu = {mu}, sigma = {sigma}")
```

The solution converged.
mu = 199.99929759918126, sigma = 118.81746040871225

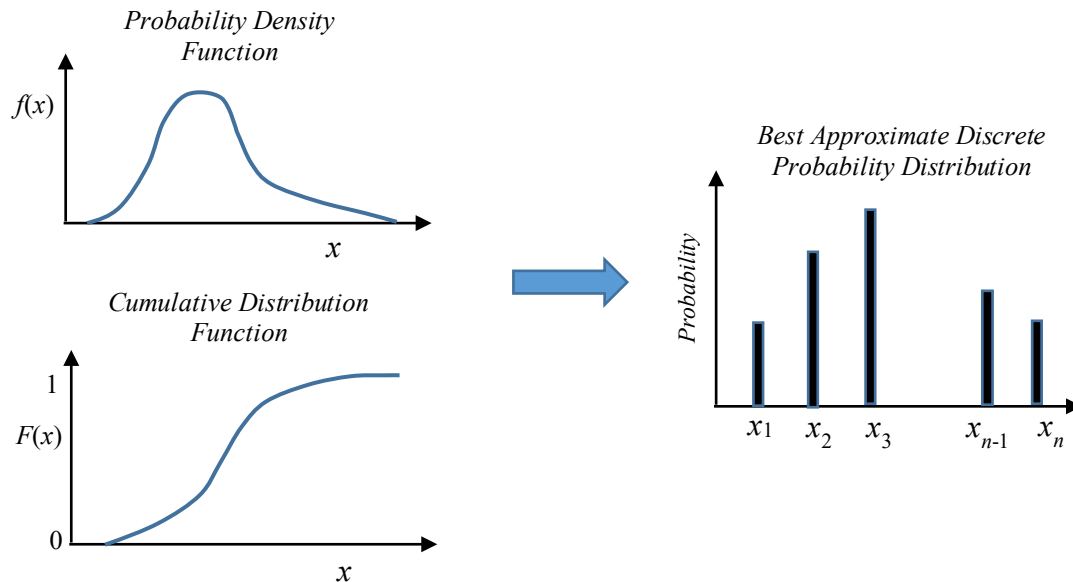
```
In [4]: # Use the same code from previous example 1 to plot:
```



7.5 Discrete Approximation of Continuous Probability Distribution

7.5.1 Introduction

- Suppose we have a continuous uncertain variable with known PDF and we wish to represent it in a decision tree or influence diagram with a discrete chance node.
- We need to find the *best approximate discrete probability distribution* with n discrete outcomes to represent it in the decision tree or influence diagram.



- There are $2n$ unknowns to be determined: $p_1, x_1, p_2, x_2, \dots, p_n, x_n$.

7.5.2 Desired Properties of the Discrete Approximation

- Given a continuous variable X with PDF $f(x)$, we seek to approximate it by a probability mass function (PMF) $p_1, x_1; p_2, x_2; \dots, p_n, x_n$.
- An obvious requirement for the probabilities is $\sum_{i=1}^n p_i = 1$. But what else should also be satisfied?
- Another desired property is that the Expected Value on *any function* $g(x)$ should be preserved across the discretization. That is

$$\int_{-\infty}^{+\infty} g(x)f(x)dx = \sum_{i=1}^n p_i g(x_i) \quad \text{for all functions } g(x)$$

- If $g(x)$ is the utility function, the expected utility will be preserved and there will be no error in optimal decision policy due to the discretization.

- Assuming that $g(x)$ can be approximated by the polynomial $g(x) = a_0 + a_1x + a_2x^2 + \dots$, we therefore, require that

$$\int_{-\infty}^{+\infty} (a_0 + a_1x + a_2x^2 + \dots) f(x) dx = \sum_{i=1}^n p_i (a_0 + a_1x_i + a_2x_i^2 + \dots).$$

- This may be rewritten in terms of the moments of the continuous and the discrete distributions:

$$a_0 + a_1E[x] + a_2E[x^2] + \dots = a_0 + a_1 \sum_{i=1}^n p_i x_i + a_2 \sum_{i=1}^n p_i x_i^2 + \dots$$

- Equating term by term:

$$E[x^k] = \sum_{i=1}^n p_i x_i^k \quad \text{for } k = 0, 1, 2, \dots$$

- Hence all moments must be preserved between the continuous and the discrete distributions.

$$k=0 \Rightarrow \sum_{i=1}^n p_i = 1 \quad \text{discrete distribution must be valid.}$$

$$k=1 \Rightarrow E[x] = \sum_{i=1}^n p_i x_i \quad \text{mean must be preserved.}$$

$$k=2 \Rightarrow E[x^2] = \sum_{i=1}^n p_i x_i^2 \quad \begin{array}{l} \text{variance must be preserved} \\ \text{(if the mean is already preserved)} \end{array}$$

$$k=3 \Rightarrow E[x^3] = \sum_{i=1}^n p_i x_i^3 \quad \begin{array}{l} \text{skewness must be preserved.} \\ \text{(if the mean is already preserved)} \end{array}$$

7.5.3 Limit on the number of moments that can be preserved

- In principle, we need to achieve

$$E[x^k] = \int x^k f(x) dx = \sum_{i=1}^n p_i x_i^k \quad \text{for all } k = 0, 1, 2, \dots$$

- For any desired number of discrete branches n , we need to determine the $2n$ values

$$p_1, x_1; \quad p_2, x_2; \quad p_3, x_3; \quad \dots, \quad p_n, x_n$$

- These $2n$ values can be determined by preserving first $(2n - 1)$ moments by solving the following $2n$ simultaneous equations:

$$\sum_{i=1}^n p_i = 1 \quad (1)$$

$$\int x f(x) dx = \sum_{i=1}^n p_i x_i \quad (2)$$

$$\int x^2 f(x) dx = \sum_{i=1}^n p_i x_i^2 \quad (3)$$

$$\dots$$

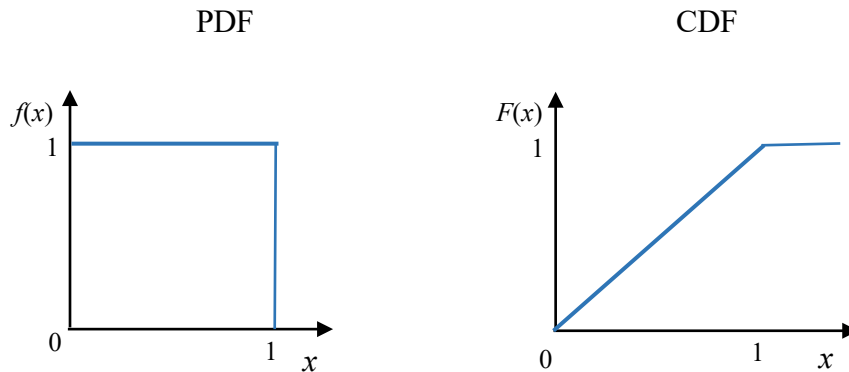
$$\int x^{2n-1} f(x) dx = \sum_{i=1}^n p_i x_i^{2n-1} \quad (2n)$$

- Hence in practice, only the first $(2n-1)$ moments can be preserved.

7.5.4 Discrete Approximation for the Uniform Distributions

Standard Uniform Distribution $U(0,1)$

- The PDF and CDF are shown below:



- If we wish to approximate it with a 2-branch probability distribution, we can only preserve the first 3 moments by solving the following 4 equations:

$$\begin{aligned}
 p_1 + p_2 &= 1 \\
 x_1 p_1 + x_2 p_2 &= \int_0^1 x dx = \frac{1}{2} \\
 x_1^2 p_1 + x_2^2 p_2 &= \int_0^1 x^2 dx = \frac{1}{3} \\
 x_1^3 p_1 + x_2^3 p_2 &= \int_0^1 x^3 dx = \frac{1}{4}
 \end{aligned}
 \quad \Rightarrow \quad
 \begin{aligned}
 x_1 &= 0.211325, & p_1 &= 0.5 \\
 x_2 &= 0.788675, & p_2 &= 0.5
 \end{aligned}$$

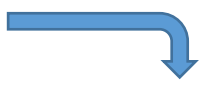
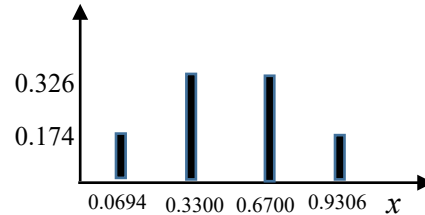
↓

- If we wish to approximate it with a 3-branch probability distribution, we can only preserve the first 5 moments by solving the following 6 equations:

$$\begin{aligned}
 p_1 + p_2 + p_3 &= 1 \\
 x_1 p_1 + x_2 p_2 + x_3 p_3 &= \int_0^1 x dx = \frac{1}{2} \\
 x_1^2 p_1 + x_2^2 p_2 + x_3^2 p_3 &= \int_0^1 x^2 dx = \frac{1}{3} \\
 x_1^3 p_1 + x_2^3 p_2 + x_3^3 p_3 &= \int_0^1 x^3 dx = \frac{1}{4} \\
 x_1^4 p_1 + x_2^4 p_2 + x_3^4 p_3 &= \int_0^1 x^4 dx = \frac{1}{5} \\
 x_1^5 p_1 + x_2^5 p_2 + x_3^5 p_3 &= \int_0^1 x^5 dx = \frac{1}{6}
 \end{aligned}
 \quad \Rightarrow \quad
 \begin{aligned}
 x_1 &= 0.112702, & p_1 &= 0.277778 \\
 x_2 &= 0.500000, & p_2 &= 0.444444 \\
 x_3 &= 0.887298, & p_3 &= 0.277778
 \end{aligned}$$

↓

- If we wish to approximate it with a 4-branch probability distribution, we can only preserve the first 7 moments by solving the following 8 equations:

$$\begin{aligned}
 p_1 + p_2 + p_3 + p_4 &= 1 \\
 x_1 p_1 + x_2 p_2 + x_3 p_3 + x_4 p_4 &= \int_0^1 x dx = \frac{1}{2} \\
 &\vdots \\
 x_1^6 p_1 + x_2^6 p_2 + x_3^6 p_3 + x_4^6 p_4 &= \int_0^1 x^6 dx = \frac{1}{7} \\
 x_1^7 p_1 + x_2^7 p_2 + x_3^7 p_3 + x_4^7 p_4 &= \int_0^1 x^7 dx = \frac{1}{8}
 \end{aligned}$$



General Uniform Distribution $U(a, b)$

- The n -branch discrete approximation for $U(a, b)$ is $(x_1, x_2, \dots, x_n; p_1, p_2, \dots, p_n)$

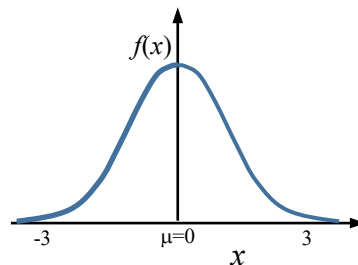
where $x_i = a + (b - a) z_i$ for $i = 1$ to n

and $(z_1, z_2, \dots, z_n; p_1, p_2, \dots, p_n)$ is the n -branch discrete approximation for $U(0,1)$

7.5.5 Discrete Approximation for Normal Distributions

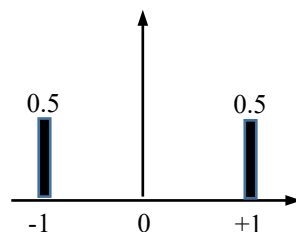
Standard Normal Distribution $N(0, 1)$

- The PDF is:



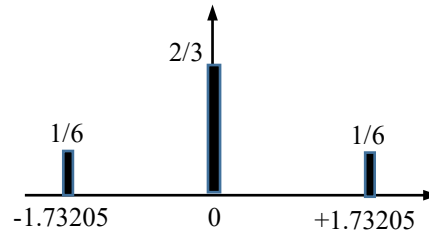
- 2-branch discrete approximation preserving the first 3 moments:

$$\begin{aligned}
 x_1 &= -1, & p_1 &= 0.5 \\
 x_2 &= +1, & p_2 &= 0.5
 \end{aligned}$$



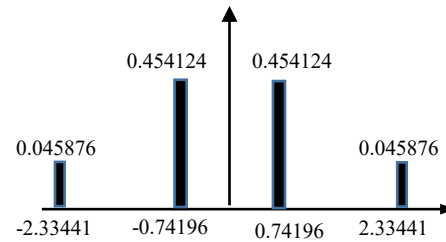
- 3-branch discrete approximation preserving the first 5 moments:

$$\begin{aligned} x_1 &= -1.73205, & p_1 &= 1/6 \\ x_2 &= 0, & p_2 &= 2/3 \\ x_3 &= +1.73205, & p_3 &= 1/6 \end{aligned}$$



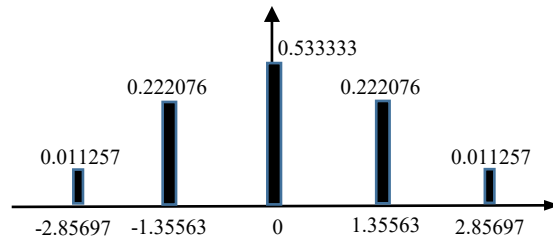
- 4-branch discrete approximation preserving the first 7 moments:

$$\begin{aligned} x_1 &= -2.33441, & p_1 &= 0.045876 \\ x_2 &= -0.741964, & p_2 &= 0.454124 \\ x_3 &= +0.741964, & p_3 &= 0.454124 \\ x_4 &= +2.33441, & p_4 &= 0.045876 \end{aligned}$$



- 5-branch discrete approximation preserving the first 9 moments:

$$\begin{aligned} x_1 &= -2.85697, & p_1 &= 0.011257 \\ x_2 &= -1.35563, & p_2 &= 0.222076 \\ x_3 &= 0, & p_3 &= 0.533333 \\ x_4 &= 1.35563, & p_4 &= 0.222076 \\ x_5 &= 2.85697, & p_5 &= 0.011257 \end{aligned}$$



General Normal Distribution $N(\mu, \sigma^2)$

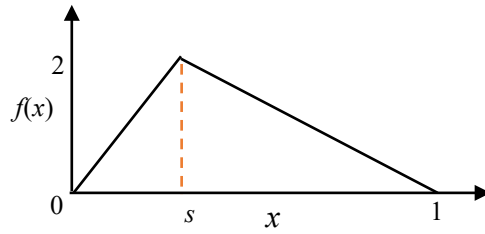
- The n -branch discrete approximation for $N(\mu, \sigma^2)$ is $(x_1, x_2, \dots, x_n; p_1, p_2, \dots, p_n)$

where $x_i = \mu + \sigma z_i$ for $i = 1$ to n

and $(z_1, z_2, \dots, z_n; p_1, p_2, \dots, p_n)$ is the n -branch discrete approximation for $N(0,1)$

7.5.6 3-Branch Discretization of the Asymmetric Triangular Distribution

- The asymmetric distribution $\text{Triangular}(0, 1, s)$ bounded between 0 and 1 has the PDF below:



- The 3-branch discretization for $\text{Triangular}(0, 1, s)$ for some values of s are given below:

s	x value	Probability
0.1	0.12793794	0.38778583
	0.43298739	0.46905473
	0.79603109	0.14315944
0.2	0.15441205	0.34663662
	0.44738893	0.50021458
	0.80108143	0.15314880
0.25	0.16315709	0.31965103
	0.45364411	0.52050614
	0.80321892	0.15984284
0.3	0.16952605	0.29192121
	0.46060361	0.54076271
	0.80546846	0.16731608
1/3	0.17277872	0.27426476
	0.46591734	0.55302152
	0.80708707	0.17271373
0.4	0.17779237	0.24308553
	0.47831381	0.57211756
	0.81059898	0.18479691
0.5	0.18377223	0.20833333
	0.5	0.58333333
	0.81622777	0.20833333

s	Value	Probability
0.6	0.189401	0.18479691
	0.5216862	0.57211756
	0.8222076	0.24308553
2/3	0.1929129	0.17271373
	0.5340827	0.55302152
	0.8272213	0.27426476
0.7	0.1945315	0.16731608
	0.5393964	0.54076271
	0.830474	0.29192121
0.75	0.1967811	0.15984284
	0.5463559	0.52050614
	0.8368429	0.31965103
0.8	0.1989186	0.1531488
	0.5526111	0.50021458
	0.845588	0.34663662
0.9	0.2039689	0.14315944
	0.5670126	0.46905473
	0.8720621	0.38778583

General Triangular (a, b, m) $a \leq m \leq b$

- The n -branch discrete approximation for $\text{Triangular}(a, b, m)$ is $(x_1, x_2, \dots, x_n; p_1, p_2, \dots, p_n)$

where $x_i = a + (b - a) z_i$ for $i = 1$ to n

and $(z_1, z_2, \dots, z_n; p_1, p_2, \dots, p_n)$ is the n -branch discrete approximation for $\text{Triangular}(0, 1, s)$
 $s = (m - a)/(b - a)$.

- s , a and $(b - a)$ are also called the shape, location, and scale parameters of the distribution, respectively.

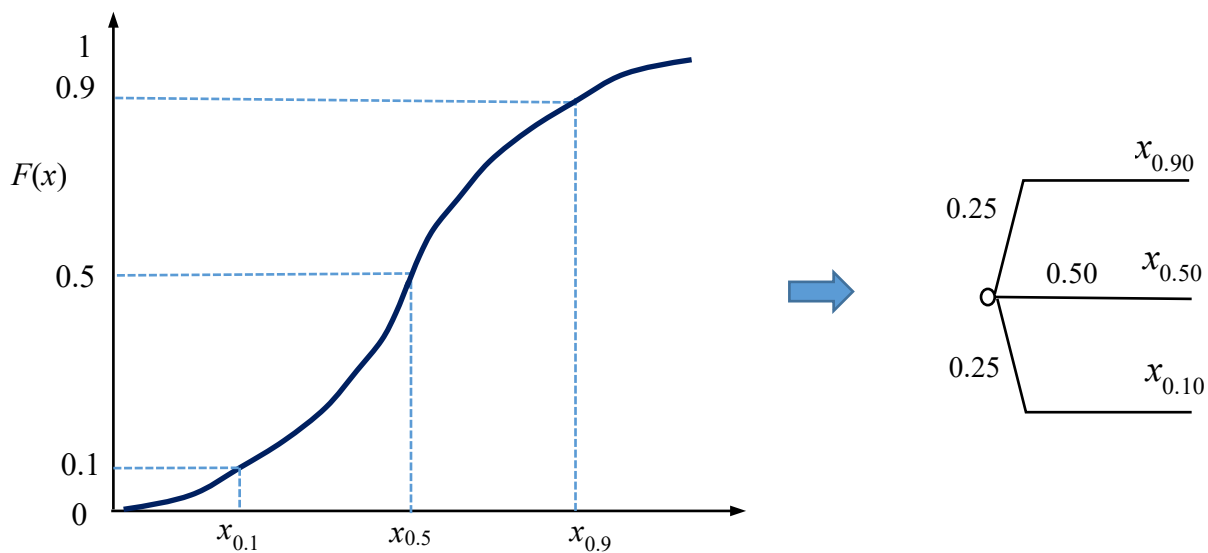
7.5.7 Discrete Approximation for Continuous Distributions in DPL

- In DPL, if a *discrete chance node* with n branches is created and a known standard continuous distribution is specified, DPL will automatically perform the $(2n - 1)$ moments matching and find the approximate branch values and probabilities for the decision tree. These values will be automatically used in creating the decision tree and generating the optimal policy tree.
- In DPL, if a *continuous chance node* is created and a known standard continuous distribution is specified, the resulting decision model can only be solved using Monte Carlo Simulation. There is no discrete optimal decision policy tree.

7.5.8 Three-Point Quick Approximation Methods

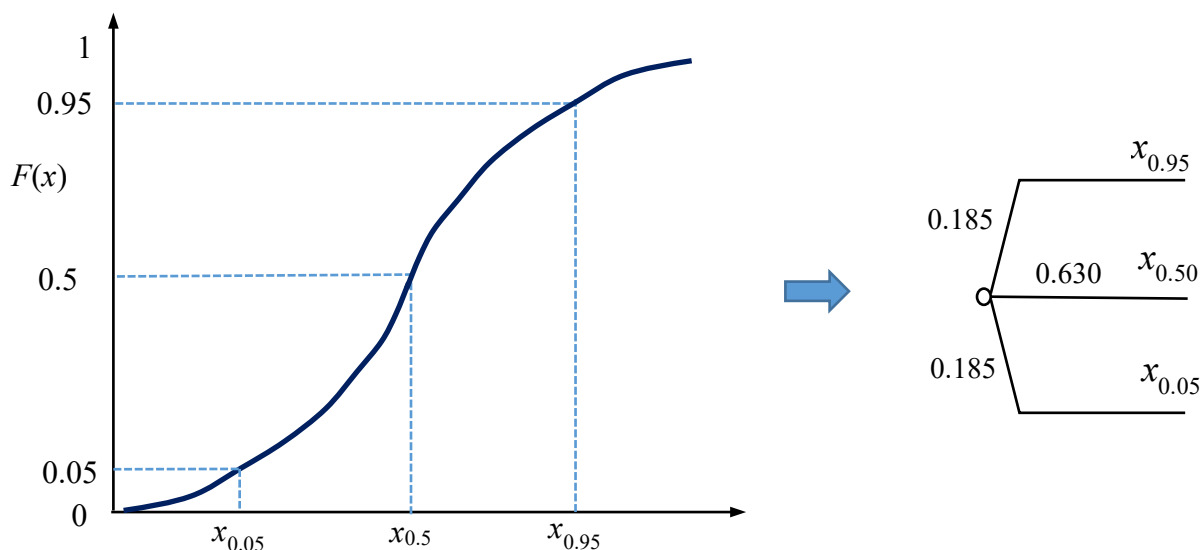
Stanford / SDG Approximation

- When the probability distribution is symmetrical, the 10, 50, and 90 percentiles provide a good approximation for a PMF with probabilities [0.25, 0.50, 0.25].



Pearson-Tukey Approximation

- When the probability distribution is symmetrical, the 5, 50 and 95 percentiles provide a good approximation for a PMF with probabilities [0.185, 0.630, 0.185].



References

1. A. Tversky and D. Kahneman (1974). Judgment under uncertainty: Heuristic and Biases. *Science* **185**:1124-1131. (Article 47 of Howard & Matheson, 1983).
2. C.S. Spetzler and C.A.S. Stael von Holstein (1972). Probability encoding in decision analysis. In *Proceedings of the ORSA-TIMS-AIEE 1972 Joint National Meeting*. (Article 33 of Howard & Matheson, 1983).
3. C.S. Spetzler and C.A.S. Stael von Holstein (1975). Probability Encoding in Decision Analysis. *Management Science* **22**(3):340-358.
4. T.S. Wallsten and D.V. Budescu (1983). Encoding Subjective Probabilities: A Psychological and Psychometric Review, *Management Science* **29**(2):151-173.
5. T.S. Wallsten and R.G. Whitefield (1986). *Assessing the Risks in Young Children of Three Effects Associated with Elevated Blood-lead Levels*. ANL/AA-32, Argonne National Laboratory, Argonne, Ill.
6. M.G. Morgan, S.C. Morris, M. Henrion, D.A.L. Amaral, and W.R. Rish (1984). Technical Uncertainty in Quantitative Policy Analysis: A Sulfur Air Pollution Example, *Risk Analysis* **4** (September) 201-216.
7. A.C. Miller III and T.R. Rice (1983). Discrete Approximations of Probability Distributions, *Management Science* **29**(3):352-362.
8. D.L. Keefer and S.E. Bodily (1983). Three-Point Approximation for Continuous Random Variables, *Management Science* **29**(5):595-609.

Exercises

P7.1 Given the following data:

5.3299	4.2537	3.1502	3.7032	1.6070	6.3923
3.1181	6.5941	3.5281	4.7433	0.1077	1.5977
5.4920	1.7220	4.1547	2.2799		

- (a) Plot a histogram representing the data.
- (b) Fit a normal distribution to the data using any suitable method.
- (c) Plot the PDF of the fitted normal distribution and compare it with the plot in (a).

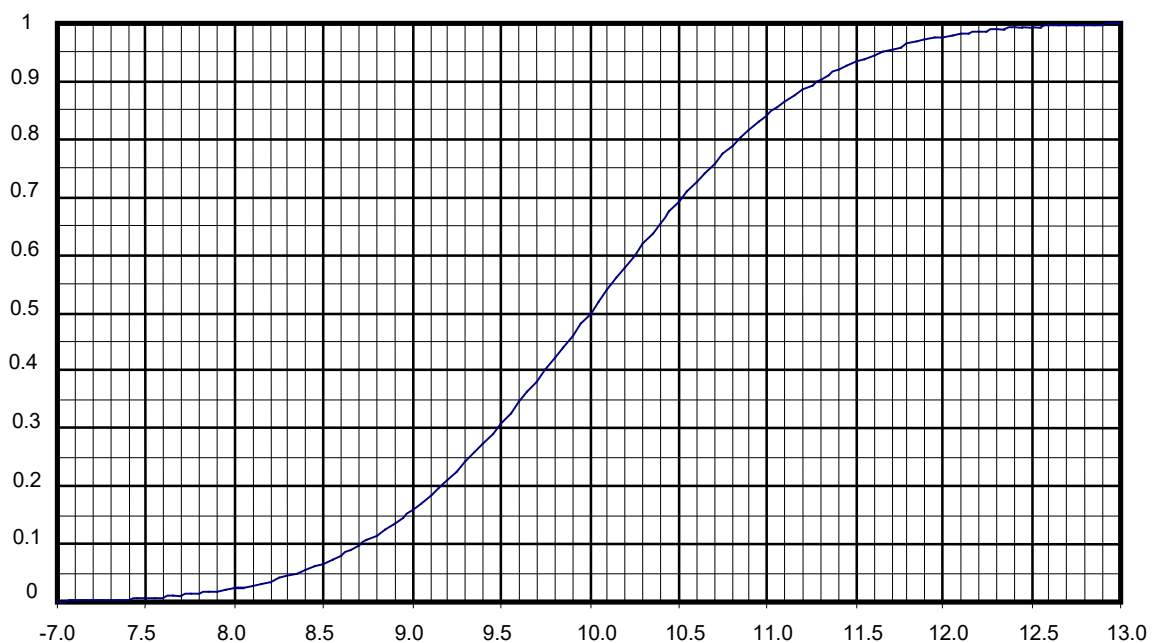
P7.2 Use DPL to find the discrete approximation of the following distributions with 3 branches:

- (a) A triangular distribution with $min=0$, $max=10$, $mode=5$.
- (b) An exponential distribution with $mean=10$.
- (c) A uniform distribution with $min=10$ and $max=20$.

P7.3 William faces an investment alternative whose net present value has a normal distribution with a mean of \$10 million and a standard deviation of \$1 million. William follows the Delta property and has a risk tolerance of \$5 million. Use DPL to determine and compare the certainty equivalent of this investment proposal using:

- (a) Discrete 3-branch approximation (moments matching).
- (b) Standard/SDG 3-branch quick approximation.
- (c) Pearson-Tukey approximation.

The CDF for the distribution Normal (10, 1) is given below:



P7.4 A scientist collected the following weights (in grams) of laboratory animals:

9.79	9.23	9.11	9.62
8.73	11.93	10.39	8.68
9.76	9.59	11.49	9.86
11.41	9.60	7.24	

- (a) Fit a probability distribution to the data using any suitable method.
- (b) Plot the PDF of the fitted distribution and compare it with the observed data.
- (c) Use the fitted distribution to estimate the probability that an animal's weight will be less than 9.5 grams.

P7.5 (Clement and Reilly 2001, Problem 10.11, modified)

A retail manager in a discount store wants to establish a policy regarding the number of cashiers to have on hand and also when to open a new cash register. The first step in this process is to determine the rate at which customers arrive at the cash register. One day, the manager observes the following times (in minutes) between arrivals of customers at the cash registers:

0.1	2.6	2.9	0.5
1.2	1.8	4.8	3.3
1.7	0.2	1.5	2.0
4.2	0.6	1.0	2.6
0.9	3.4	1.7	0.4

Assuming that customers' arrival is a poison process, fit an exponential distribution with one parameter λ to the data.

P7.6 (Clement and Reilly 2001, Problem 10.12, modified)

An ecologist studying the breeding habits of birds sent volunteers from the local chapter of the Audubon Society into the field to count nesting sites for a particular species. Each team was to survey five acres of land carefully. Because she was interested in studying the distribution of nesting sites within a particular kind of ecological system, the ecologists were careful to choose survey sites that were as similar as possible. In total, 24 teams survey five acres each and reported the following numbers of nesting sites in each of the five-acre parcels:

7	12	6	9
5	2	9	9
7	3	9	9
5	1	7	10
1	8	6	3
4	5	3	13

- (a) Plot a histogram for these data.
- (b) Fit a probability distribution to the data using any suitable method.
- (c) Compare the PMF of the fitted distribution with the histogram plotted in (a).

This page is blank.