Unit 3

Descriptive Statistics

* Two main concepts of statistics

(i) Population - a collection of objects, items (units) about which information is sought..

(ii) Sample - part of the population that is observed

* Without data analysis, claims would be merely anecdotal evidence. The claim may not be true because

(i) a small no. of samples

(ii) a selection bias

(iii) confirmation bias (believers more likely to contribute data)

(iv) inaccurate sources (stories subject to memory deformations)

* Data preparation

(i) Getting the data : from files | scraping the web

(ii) Parsing the data : depends on what format the data is in, plain text, fixed columns, CSV, XML, HTML.

(iii) Cleaning the data : Survey responses and other data files are almost always incomplete, have errors. Remove | ignore incomplete records

(iv) Building data structures : store data in a structure that lends itself to the analysis

in-memory storage = a data structure

out of memory data structure = a database, databases provide a mapping from keys to values (serve as dictionaries)

* Exploratory data analysis : aims to visualize and summarize the sample distribution, allowing one to make tentative assumptions about the population distribution.

* Data Cleaning: The most common steps are

(i) Sample the data: if the amount of raw data is huge, take a sample of it

(ii) Inpute missing data : done by either removing such records or by filling in the missing data based on data from other records, by computing the mean or median

(iii) Normalize numeric value: Transform numeric data into a uniform range

(iv) Reduce dimensionality : Remove irrelevant and redundant input variables

(v) Add derived features : compute additional attributes from existing attributes

(vi) Discretize numeric values into categories : sort data into buckets, based on ranges

(vii) Binarize categorical attributes : for certain ML models, that can only take binary attributes

(viii) Select, combine and aggregate data : multiple pieces of raw data are combined together, to provide stronger signals to the learning algorithm.

* Summarizing the data

A. Mean $\mu = \frac{1}{n} \sum_{i=1}^{n} x_i$

B. Variance $\sigma^2 = \frac{1}{n} \sum (x_i - \mu)^2$

$(x_i - \mu)$ = deviation

σ = standard deviation, square root of variance

Note: standard deviation is considered, because variance is hard to interpret (eg. if the unit is in grams, the variance would be gram squared).

Outliers: An item of the data that is significantly different from the rest of the data is called an outlier.

* Choosing median over mean: In the case of an outlier, the mean would drastically shift towards the outlier. Calculating the median is done by ordering all of the values, and then choosing the one in the middle.

Quantiles and Percentiles

- Take a sample with $x_1, x_2 \ldots x_i$
- Order the sample
- Choose an x_p such that the data is divided into 2 parts:
  - (i) a fraction of the data values are lesser than x_p
  - (ii) $(1-p)$ values are greater than x_p.

Here, x_p is the p^{th} quantile

Percentile = 100 x pth quantile

5-number summary: Consider the set $x_{min}, Q_1, Q_2, Q_3, x_{max}$

$\Rightarrow Q_1$ is the 25 x pth percentile

Q_2 50 x pth percentile

Q_3 75 x pth percentile

Data distributions

Histograms: A graph that shows the frequency of each value

A drawback of plotting histograms with frequencies is that if the dataset is biased, the inferences drawn may be totally incorrect. Thus, the frequencies of the histogram can be normalized by dividing by n (the no. of samples). This son of normalized histogram is called the Probability Mass Function (PMF).

* In order to plot PMF histograms, set the parameter density = True. The density parameter normalizes bin heights such that the integral of the histogram is 1. (normed can also be used, but is deprecated)

(seaborn - a python module for visualization of statistical graphics)

Cumulative Distribution Function (CDF): CDF describes the probability that a real valued random variable X, with a given probability distribution will be found to have a value <= x.

* Outlier Treatment

Some rules which can be used to detect outliers are:

(i) computing samples that are far from the median

(ii) computing samples whose values exceed the mean by 2 or 3 standard deviations.

eg In the example, where the salaries of men and women are computed, the median age is found to be 37. It is assumed that the lower limit to be able to work is 22 and the max age is 72. Anything ~~greater~~ outside these limits is considered to be an outlier.

* Measuring Asymmetry

Skewness - a statistic that measures the asymmetry of a distribution.

The skewness g_1 is given by:

$$g_1 = \frac{\frac{1}{n}\sum(x_i - \mu)^2}{\frac{1}{n}\sum(x_i - \mu)^3}$$

$$g_i = \frac{1}{n}\frac{\sum(x_i - \mu)^3}{\sigma^3}$$

numerator = mean squared deviation, i.e variance

denominator = mean cubed deviation

Negative skewness => distribution skews left (extends further to the left than to the right)

Positive skewness => distribution skews right

skewness of a normal distribution and of symmetric data = 0.

Disadvantage of computing skewness : Outliers have a disproportionate effect on g_1.

Alternate method to evaluate asymmetry : observe the relationship between the mean and the median. Extreme values have more effect on the mean than on the median, so in a distribution that skews left, the mean is lesser than the median.

```
# a function to calculate skeconess
def skewness(x):
    res =0
    m = x.mean()
    s = x.std()
    for i in x:
        res += ((i-m) * (i-m) * (i-m)) / (s*s*s)
```

*Pearson's Median Skewness Coefficient : an alternative method of skewness that explicitly captures the relationship between the mean μ and the median $\mu_{1/2}$.

$$g_p = \frac{3(\mu - \mu_{1/2})}{\sigma}$$

This statistic is robust, meaning that it is less vulnerable to the effect of outliers.

```
# to calculate Pearson's median skewness coefficient
def pearson(x):
    return  3 * (x.mean() - x.median()) / x.std()
```

* Relative Risk : Relative risk is a ratio of 2 probabilities. It utilizes the probability of an event occurring in one group to the probability of an event occurring in the other group. Relative risk values are >= 0. A value of 1 indicates a neutral result, the chance of an event occurring for one group is the same as that for the other group. A value of 0 indicates that none of the cases in group 1 had the event occur, while x number of cases in group 2 had the event occur.

For eq. To calculate the relative risk of getting promoted early for men and women.
An early promotion occurs when the age of the employee is less than 41 years.

```
male_age = mi1['age']        # dataframe
female_age = fmi['age']

m_young = len(male_age[(male_age<41)])/ float( len( male_age.index ))

f_young = len(female_age[(female_age<41)]) /float(len(female_age.index))
```

Relative risk of men getting promoted early = $\frac{m\_young}{f\_young} = P(E)$

for women: $1 - P(E)$

(Multiply by 100 to get the percentage of relative risk)

* Continuous Distributions

Empirical distribution: each possible event is assigned a probability derived from an experimental observation

Continuous distribution: Distributions that are defined by a continuous function

Exponential distribution: Exponential distributions explain the inter-arrival time between events.

- If the events are equally likely to occur at any given time, the distribution of the inter-arrival time tends to an exponential distribution.

Continuous Distribution function: $CDF(x) = 1 - e^{-\lambda x}$

Probability distribution function: $PDF(x) = \lambda e^{-\lambda x}$

λ = shape of the distribution

mean = $\frac{1}{\lambda}$ variance = $\frac{1}{\lambda^2}$ median: $\frac{\ln(2)}{\lambda}$

Plotting an exponential distribution

```
import numpy as np
import matplotlib.pyplot as plt
l = 3                    (# λ value)
x = np.arrange ( <lower limit>, <upper limit>, <step> )
                  # arrange() is used to get evenly spaced values in an
                    interval

y = 1 - np.exp(-l*x)      # for CDF
(OR)
y = l * exp(-l*x)         # for PDF

plt.title('          ')
plt.xlabel('         ')
plt.ylabel('         ')
plt.show()
```

Real world events that can be described with exponential distribution

(i) radioactive decay

(ii) time before one's next telephone call

(iii) lifetime of some batteries $PDF(x) = \frac{1}{4} e^{-x/4}$

Normal Distribution

- also called the Gaussian distribution, it is the most commonly used distribution as it describes numerous phenomena.
- It has no closed-form CDF expression
- The PDF expression: $PDF(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{\frac{-(x-\mu)^2}{2\sigma^2}}$

σ describes the shape of the distribution.

```
# Plotting the normal distribution
import numpy as np
import matplotlib.pyplot as plt
u=6
s=2

x = np.arange(<lower limit>, <upper limit>, step)
y = (1/(np.sqrt(2*np.pi*s*s)))*np.exp(-((x-u)**2)/(2*s*s)))

plt.plot(x,u)
plt.title('          ')
plt.xlabel('          ')
plt.ylabel('          ')
plt.show()
```

Events that follow a normal distribution

(i) size of living tissue (length, height, weight)

(ii) length of inert appendages

(iii) physiological measurements like blood pressure

Central Limit Theorem

- Consider any distribution of values (random / continuous), which has a definite mean, median, variance.
- Set a sampling size of n.
- Draw a sample of size n, and compute its mean, say $\bar{x}_1$
- Draw another sample of size n, and compute its mean, say $\bar{x}_2$.
- Continue drawing samples of size n, and compute their mean.
- Plot the frequency of occurrence of each of these means, $\{\bar{x}_1, \bar{x}_2 \ldots \bar{x}_m\}$.
- The distribution approximately would look like a Gaussian distribution, even if the original distribution was far from normal.

Statement: Take the mean of n random samples from any arbitrary distribution with a well-defined standard deviation σ and mean μ. As n gets bigger and bigger, the distribution of the sample mean would always converge to a Gaussian distribution with a mean μ and standard deviation $\frac{\sigma}{\sqrt{n}}$. That is, the distribution of an average tends to (be normal) even when the distribution from which the average is computed is far from normal.

Kernel Density Estimates

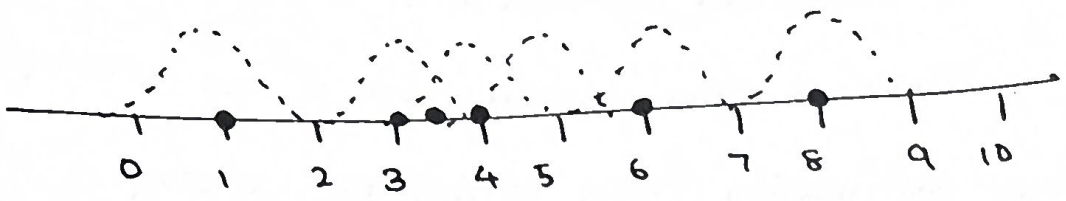
Kernel density estimation is a form of a non-parametric density estimation.

In some cases, a data sample may not resemble a common probability distribution. This often happens when the data has 2 peaks (a bimodal distribution) or multiple peaks (a multimodal distribution). Here, a parametric density estimation is not possible. Instead, an algorithm is used to approximate the probability distribution of the data, which is called a non-parametric estimation.

One such non-parametric estimation method is the Kernel Density Estimation (KDE). No assumptions are made about the underlying distribution.

A Kernel is a mathematical function that returns a probability for a given value of a random variable. The Kernel effectively smoothens the probabilities across a range of outcomes. There is a parameter, called the smoothening parameter, that controls the number of samples/window of samples used to estimate the probability for a new point.

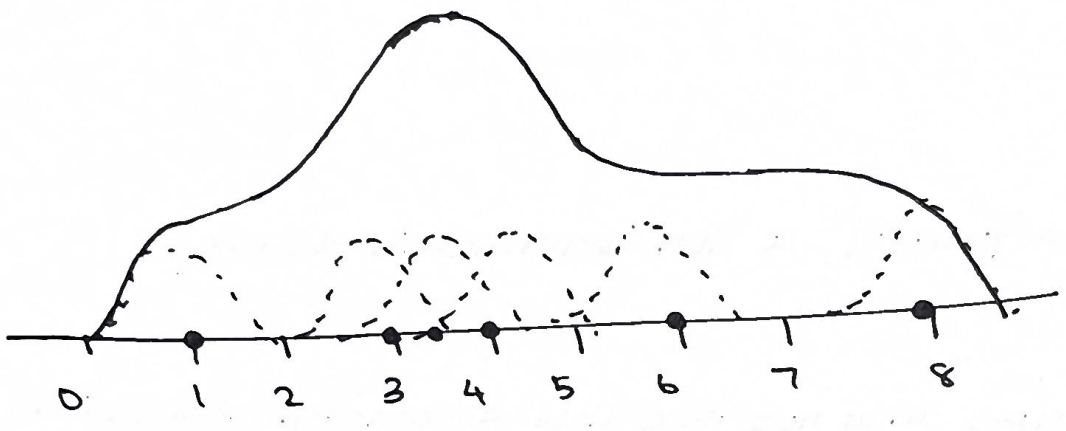
For example: consider a dataset as follows



Add a Gaussian kernel centred about each data point

(The Gausslan kernel is the default kernel, other kernel functions also exist)

These kernels are added up cumulatively



code?

*<u>Estimation</u> : the process of inferring the parameters (like the mean) of a distribution from a statistic of samples drawn from a population.

Consider a sample from a population. The sample mean is $\bar{x}$. The mean of the entire population, μ can be estimated from the sample mean. This process is estimation, and here the sample mean is the estimator.

If there are no outliers, the sample mean $\bar{x}$, minimizes the mean squared error.

$$MSE = \frac{1}{n} \sum (\bar{x} - \mu)^2$$

where n is the no. of times we estimate the mean.

* Program to compute MSE

```
NTS = 200          # no. of trials
mu = 0.0
var = 1.0
err = 0.0
NPS = 1000
for i in range(NTS):
        x = np.random.normal(mu, var, NPS)
        err += (x.mean() - mu) **2
print('MSE:', err / NTS)
```

mean, std, no. of samples

Variance as an estimator

If $\bar{\sigma}^2$ is the sample variance, then

$$\bar{\sigma}^2 = \frac{1}{n} \Sigma (x_i - \bar{x})^2$$

This estimator works well only for large samples, and is biased for a ~~large~~ small no. of samples. For a smaller sample set:

$$\bar{\sigma}^2 = \frac{1}{n-1} \Sigma (x_i - \bar{x})^2$$

* Standard score: When data has to be compared, or when relations have to be established, data that has different units must be avoided. In order to compare such data, they must be standardized. This is done by subtracting the mean from x_i, and then dividing by the standard deviation.

i.e $$z_i = \frac{(x_i - \mu)}{\sigma}$$

- z_i is a dimensionless quantity. Their distribution has a mean of 0 and a variance of 1.
- It inherits the shape of the dataset, if X is normally distributed, so is Z. If X is skewed, so is Z.

Covariance : Covariance is a measure of the tendency of two variables to vary together. If there are 2 series X and Y, their deviations from the mean are:

$$dx_i = x_i - \mu_x$$

$$dy_i = y_i - \mu_y$$

where μ_x is the mean of X and μ_y is the mean of Y.

- If X and Y vary together, their deviations tend to have the same sign
- If they are multiplied together, the product is positive when the deviations have the same sign and negative when they have the opposite sign. Adding up the products gives a measure of the tendency to vary together.

$$Cov(X,Y) = \frac{1}{n} \sum dx_i dy_i$$, where n is the length of the 2 series

Disadvantages of covariance : (i) difficult to interpret

(ii) the units are the products of X and Y. (i.e covarience of weight & height would have the units kg-m, which doesn't mean much).

```
# Program to calculate covariance
def cov(X,Y)
    def _get_dvis(V)
        return [v - np.mean(V) for v in V]

    dxis = _get_dvis(X)
    dyis = _get_dvis(Y)
    return np.sum([x*y for x,y in zip(dxis, dyis)]) / len(X)
```

X = [. . .]
Y = [. . . .]

X & Y are arrays

Pearson's Correlation : If the data is normalized w.r.t their deviation, it leads to their standard scores. When they are multiplied:

$$\rho_i = \frac{(x_i - \mu_x)}{\sigma_x} \cdot \frac{(y_i - \mu_y)}{\sigma_y} \quad \text{①}$$

The mean of this product is given by $\rho = \frac{1}{n} \sum_{i=1}^{n} \rho_i$.

① can be written as $\rho = \frac{Cov(X, Y)}{\sigma_x \sigma_y}$

This is called the Pearson's correlation.

- The Pearson's correlation always lies between -1 and 1. The magnitude indicates the strength of the correlation.
- If $\rho = 1$, the variables are perfectly correlated. If you know the value of one, we can perfectly predict the value of the other.
- If $\rho = -1$, the variables are negatively correlated.
- However, if $\rho = 0$, it does not necessarily mean that the variables are not correlated. This is because Pearson's correlation works only for linear relationships.

(Note: It also does not work when there are outliers)

Code:

```
if len(X) <= len(Y):
    return Cov(X, Y) /
        np.prod([np.std(v) for v in
            [X, Y]])
```

* Spearman's Rank Correlation

Spearman's rank correlation is an alternative that mitigates the effect of outliers and skewed distributions.

To compute the Spearman correlation, the ranks of each value must be computed, which is ~~the~~ its index in the sorted sample.

For eg. if $X = \{7, 1, 2, 5\}$, the rank of 5 is 3, because it appears third if we sort the elements.

Then, Pearson's correlation is calculated, but for the ranks.

Code:

X = [. .] .

Y = [. .]

```
def list2rank(l):
    # l is a list of numbers
    # The function returns a list of 1-based index, and the mean when
     there are multiple instances
     return [np.mean([i+1 for i, sorted_el in enumerate(sorted(l)) if
                      sorted_el == el]) for el in l ]


def spearmanrank(x, y)
    return corr(list2rank(x), list2rank(y))
```

Data Science

CAT-2

Descriptive Statistics

2 marks

- Central Limit Theorem
- PDF of a normal distribution: $\frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$
- CDF of an exponential distribution: $1 - e^{-\lambda x}$
- PDF of an exponential distribution: $\lambda e^{-\lambda x}$

mean $= \frac{1}{\lambda}$ variance: $\frac{1}{\lambda^2}$ median $= \frac{\ln(2)}{\lambda}$

Code:

```
import matplotlib.pyplot as plt
import numpy as np
l = 3                    (# shape of the distribution, λ)
x = np.arrange(0, 2.5, 0.1)      # parameters are lowerlimit, upper limit & step value
CDF = 1 - np.exp(-l * x)
PDF = l * np.exp(-l * x)
plt.plot(x, CDF, '-')
(or)
plt.plot(x, PDF, '-')
plt.titel('          ')
plt.xlabel('x')
plt.ylabel('e    ')
plt.show()
```

```
# plotting a Gaussian/normal distribution
PDF(x) = 1/√(2πσ²) e^(-(x-μ)²/2σ²)
m
mean = 6
s
std = 2
x = np.arrange(0, 20, 0.1)
y = (1/(np.sqrt(2*np.pi*s*s))) *
    np.exp(-((x-m)**2)/(2*ss))
plt.plot(x, y, '-')
plt.title(    )
plt.xlabel(    )
plt.ylabel(    )
plt.show()
```

- What is a continuous distribution ? when is it used ?

  → Distributions that are defined by a continuous fn.

  → exponential fn : describes inter - arrival time.

  → another eg : normal distribution

  → used when there is a range of values

- Kernel density estimates, when is it used ?

  → a form of non - parametric density estimation

  → the data does not resemble any common probability fn.

  → used for data with multiple peaks, (bimodal or multimodal distribution)

- relative risk

  → ratio of 2 probabilities

  → compares the probability of occurrence of an event in 1 group to that in another

  → relative risk >= 0.

  if 1 : event is equally likely to happen in both groups

  if 0 : event did not occur in 1 group, occurred x no. of times in another group.

Example: To calculate the relative risk of early promotion in men & women. Early promotion is when the age is lesser than 41

code: import pandas as pd

&

```
male_age = ml['age']
female_age = fml['age']
```

ml & fml are dataframes with info about men & women respectively.

```
male_young = len(male_age[male_age<41])/
             float(len(male_age))

female_young = len(female_age[female_age<41])/
               float(len(female_age))

relativerisk = male_young/female_young
```

relative risk of men being promoted early

```
#for women
relative_women = 1- relativerisk
```

* Difference between Pearson's Correlation & Spearman's Rank Correlation

  - PC works with raw data, while SRC works with rank-order data.
  - PC fails if the data is non-linear, Spearman's Rank Correlation works for non-linear data too.
  - PC is less robust towards outliers, SRC is more robust towards outliers

* Why is Spearman's rank correlation preferred sometimes?

  → more robust towards outliers

  → works on non-linear data

* relation between mean & median

→ mean = median for symmetric distributions

* What is the point estimate of the population mean ?

→ sample mean

* What is the difference between PMF & CDF ?

→ PMF describes the probability distribution of a discrete random variable.

→ PMF can be computed by dividing the frequency by the total no of samples (n)

→ CDF — can be for random vars either discrete or continuous.

→ It is the probability that a random variable X with a given distribution has a value $\leq x$.

6mark Questions

① measuring asymmetry

→ Skewness is a measure of asymmetry

→ $g = \frac{1}{n} \frac{\Sigma(x_i - \mu)^3}{\sigma^3}$

→ left skewness & right skewness

→ skewness for a symmetric distribution

code: def skewness(x):

```
    res = 0
    m = x.mean()
    s = x.std()
    for i in x:
        res += (((i-m)*(i-m)*(i-m)) / (s*s*s))
    print(res)
```

→ Pearson's median skewness coeff : $3(\mu - \mu_{1/2}) / \sigma$

(a relationship between mean & median)

mean
variance
std. scores
cov
pearson's correlation
Spearman's rank correlation

PART-C

12 mark questions

(1) Estimation

Estimation is a method by which the parameters of a distribution can be computed by calculating statistical values of samples drawn from the distribution.

Mean, std. deviation etc. can be used as possible estimators.

Mean as an Estimator

Let μ be the mean of the population.

Consider a sample from the population. Let its mean be $\bar{x}$.

The mean of the population, μ can be estimated from the mean of the sample.

If the population doesn't have too many outliers, the mean squared error is calculated as:

$$MSE = \frac{1}{n} \Sigma (\bar{x} - \mu)^2$$

n = no. of times the mean is estimated.

Code to calculate mean squared error

```
import numpy as np.

mu=0
var=1
err=0
samples = 1000
trials = 200
```

```
for i in range(trials):
    x=np.random.normal(mu,var,samples)
    err += (x.mean()-mu)**2

mse = err/trials

print('Mean squared error:', mse)
```

Variance as an estimator

Let $\overline{\sigma}^2$ be the sample variance, then

$$\overline{\sigma}^2 = \frac{1}{n} \Sigma (xi - \overline{x})^2$$ # applicable only for large samples

for smaller samples

$$\overline{\sigma}^2 = \frac{1}{n-1} \Sigma (xi - \overline{x})^2$$

Standard scores : → to compare data that has diff units

→ must be in a standardized format.

→ Standard scores are calculated as

$$zi = \frac{(xi - \mu)}{\sigma}$$

→ zi is dimensionless

mean = 0

variance = 1

→ takes the shape of the original distribution, if X is normal, z is normal, if X is skewed, z is skewed.

Covariance : a measure of the tendency of 2 variables to vary together. For 2 variables x & y

$dxi = xi - \mu x$

$dyi = xi - \mu y$

If dxi and dyi have the same sign, they vary together

If they vary together, their product dxidyi would ~~have~~ ~~be~~ ~~be~~ be +ve, if not, the product would be negative.

Summing up of all of these products gives the covariance.

$$\text{covariance}(x,y) = \frac{1}{n} \sum dx_i dy_i$$

where n is the no. of elements / length of series.

Code : X & Y are 2 arrays

```
import numpy as np
def get_difference(arr):    # a function to calculate the values of dxi and dyi
    return [val - np.mean(arr) for val in arr]
def cov(x,y):
    dxi = get_difference(X)
    dyi = get_difference(Y)
    cov = np.sum([x*y for x,y in zip(dxi,dyi)]) / len(X)
```

Pearson's correlation : If data is normalized by calculating their deviation, standard scores are obtained. The product of the standard scores would be

$$P_i = \frac{(x_i - \mu_x)}{\sigma_x} \cdot \frac{(y_i - \mu_y)}{\sigma_y}$$

Their overall sum : $P = \frac{1}{n} \sum_{i=1}^{n} P_i$

This can be rewritten as $\frac{cov(X,Y)}{\sigma_x \sigma_y}$

This is called the Pearson's correlation.

- Pearson's correlation takes values between -1 & 1
- $1 \rightarrow$ completely correlated. If one value is known, the value of the other is known for sure
- $-1 \rightarrow$ negatively correlated
- 0 does not mean no correlation at all, as Pearson's correlation works only for linear relationships

Code: def corr(X,Y): # X,Y are 2 arrays

    if len(X) <= len(Y):

        return cov(x,y) / np.prod[np.std(i) for i in [X,Y]]

Spearman's rank correlation

→ a better alternative – mitigates outliers & skewed distributions

→ rank of each value is computed

if $X = \{7, 2, 1, 5\}$ rank of 5 is 3.

Pearson's correlation is then calculated.

Code: def list_to_rank(l): # l is a list of nos.

    return [np.mean(i+1 for i, sorted_el in enumerate(sorted(l))

    if sorted_el == el) for el in l]

2. Data Distributions

Data distributions depict how often each value in a population/a sample appears.

The most common representation is a histogram, that shows the frequency of each value.

Consider a dataset with info on working men & women.

In order to study the distribution of ages, a frequency histogram can be plotted.

Code:

```
import matplotlib.pyplot as plt
male-age = ml['age']        # ml is the dataframe var
male-age.hist(density = False, histtype = 'stepfilled', bins = 20)
plt.xlabel('Age')
plt.ylabel('No. of men')
plt.show()
```

Illy for women

```
female-age = fml['age']
female-age hist(density = False, histtype = 'stepfilled', bins = 20)
plt.xlabel('Age')
plt.ylabel('No. of women')
plt.show()
```

However, plotting histograms on the basis of frequency may not be accurate, if there is a bias. (For eg, the no. of men may be larger that of women).

In this case, the frequencies of the histogram have to be normalized.

This is done by dividing the frequencies of the histogram by n, the no. of samples. The normalized histogram is called the Probability Mass Function.

Code

```
male_age.hist (density=True, histtype = 'stepfilled', bins = 20)
plt.xlabel ('Age')
plt.ylabel ('Probability')
plt.show()
# Ily for women
female_age.hist (density=True, histtype= 'stepfilled', bins =20)
plt.xlabel ('Age')
plt.ylabel ('Probability for women')
plt.show()
```

Another method to show the distribution of the data is using the CDF - Cumulative Distribution function. The CDF describes the probability that a real-valued random variable X with a given probability distribution would have a value <= x.

For eg, a CDF can be plotted for the ages of men & women

Code:

```
male_age.hist(density=True, cumulative =True, histtype = 'step',
        linewidth = 3, bins=20, color ='red')
female_age.hist(density = True, cumulative =True, histtype = 'step',
        linewidth =3, bins = 20, color = 'blue')
```

3 Outliers and their treatment

Outliers - An item of the data that is significantly different from the rest of the data is an outlier. That is, their value is far from the central tendency.

Detecting outliers in computing samples that are far away from the median

(i) computing samples whose values exceed the mean by 2 or 3 standard deviations.

For eg. in order to compute the outliers in a dataset with the ages of working men & women, initially some domain knowledge is used to identify the min & max working age.

The median of the data is then computed. All those values that are outside the limits are dropped.

For eg. if the median age is 37

and lower limit = 22,

upper limit = 78.

Let the original dataframe = df

Let the new dataframe = ndf

```
ndf = df.drop (df.index (df['age'] > df['age'].median()+35)
&(df['age'] < df['age'].median()))
```

Dataframes for men & women with the outliers present

```
male-age   = m1['age']
female-age = fm1['age']
```

Cleaning out the outliers from these 2 dataframes

```
male_new = male-age.drop( male-age.index [male-age >
  df.['age'].median()+35) & ( male-age < df['age'].median()
                                        -15 )] ])
```

```
female-new = female-age.drop (female-age.index [female-age>
  af,['age'].median()+35) & (female-age <df['age'].median()
                                        -15)] )
```

The change in mean, std deviation and median before & after removing the outliers

For men

```
male-old-mean = male_age.mean()
male-old-median = male_age.median()
male-old-std = male-age.std()

male-new-mean = male_new.mean()
male-new-mean = male_new.median()
male-new-std = male-new.std()
```

* The same can be shown for women as well

- To visualize the data before & after removing outliers, a histogram can be plotted.

```
import matplotlib.pyplot as pyplot
plt.figure ( figsize = (10,5))
af.age[(af.income == '>50K')].plot (color='blue')
df2.age [(df2.income =='>50K')].plot(color='red')
```

- The difference in mean age with & w/o outliers

```
# w/ outliers
print(male_age.mean() - female_age'.mean())

# w/o outliers
print(male_new.mean()- female_new.mean())
```