Project Work on

Descriptive Statistics And its Measures

Group-A

Sagnik Samanta – STA212004

Debadittya Ganguly - STA212005

Nirnoy Ghosh - STA212029

Shayani Roy – STA212033

Date - 02/02/2022

Introduction:

Our topic of project Descriptive statistics and its measures focuses on Measures of central tendencies like types of mean , median , mode ,quartiles etc. Measures of dispersion like range , variance, mean deviation and coefficient of variation etc. Measures of symmetry like skewness , kurtosis etc. The different types of methods they possess for data handling and data analysis are formulated by the help of own functions created by us through R-Programming and compiled in Statistical software R-Studio.

Measure of central tendency

Discrete frequency distribution

```
discrete_freq_table=function(x){
  y=x[1]
  for(i in 2:length(x)){
    if(sum(x[1:(i-1)]==x[i])==0)
      y=c(y,x[i])
  }
  xus=sort(y)
  xus
  freq=xus*0
  total_sum=freq*0
  for(i in 1:length(xus)){
    freq[i]=sum(x==xus[i])
    total_sum[i]=xus[i]*freq[i]
  }
  freq
  total_sum
  m=sum(total_sum)/sum(freq)
  freq.tb=cbind(xus,freq,total_sum)
  freq.tb
  return(m)
}

x<-c(42,74,40,60,82,115,41,61,75,83,63,53,110,76,84,50,67,65,78,77,56,95,68,6
9,104,80,79,79,54,73,59,81,100,66,49,77,90,84,76,42,64,69,70,80,72,50,79,52,1
03,96,51,86,78,94,71)
discrete_freq_table(x)

[1] 72.58182
```

Alternative

```
mean_1_f=function(x){
  sum=0
  for(i in 1:length(x)){
    sum=sum+x[i]
```

```
  }
  m=sum/length(x)
  return(m)
}
x=c(1,2,3,4,5,6)

mean_1_f(x)

[1] 3.5
```

Alternative

Finding Mean for Discrete Data

```
# To find the vector of distinct values
distinct=function(x){
  y=x[1]
  for(i in 2:length(x)){
    if(sum(x[1:(i-1)]==x[i])==0)
      y=c(y,x[i])
  }
  return(sort(y)) # distinct values are retunred in increasing order
}
distinct(x)

[1] 1 2 3 4 5 6

freq.table=function(x){
  xu=distinct(x) ## to count number of distinct elements
  xu
  xus=sort(xu)
  xus
  freq=xus*0
  for(i in 1:length(xus)){
    freq[i]=sum(x==xus[i]) ## counting frequency for ith distinct value
  }
  freq
  freq.tb=cbind(xus,freq)
  freq.tb
  return(freq.tb)
}
freq.table(x)

     xus freq
[1,]   1    1
[2,]   2    1
[3,]   3    1
[4,]   4    1
[5,]   5    1
[6,]   6    1
```

```
mean_2=function(x,f){
  sum=0
  freq=0
  for(i in 1:length(x)){
    sum=sum+(x[i]*f[i])
    freq=freq+f[i]
  }
  m=sum/freq
  return(m)
}
x=c(1,3,2,4,4,6,2,2,2,2,2,4,3)
y=freq.table(x)[,1]
f=freq.table(x)[,2]
mean_2(y,f)

[1] 2.846154
```

Alternative

```
distinct_1=function(x){
  y=x[1]
  for(i in 2:length(x)){
    if(sum(x[1:(i-1)]==x[i])==0)
      y=c(y,x[i])
  }
  return(sort(y))
}
Freq_1=function(x){
  a=distinct_1(sort(x))
  sum=rep(0,length(a))
  for(i in 1:length(a)){
    for(j in 1:length(x)){
      if(a[i]==x[j]){
        sum[i]=sum[i]+1
      }
    }

  }
  return(sum)
}
x=c(1,1,1,2,8,7,3,4,2,3,2,4)
Freq_1(x)

[1] 3 3 2 2 1 1

Mean_1=function(x){
  sum=0
  y=distinct_1(x)
  a=length(y)
  f=Freq_1(x)
  n=sum(f)
```

```
  for(i in 1:a) {
    sum=sum+(y[i]*f[i])
  }
  m=(sum/n)
  return(m)
}
x=c(1,1,2,2)
Mean_1(x)

[1] 1.5
```

Mean for continuous frequency distribution

```
continuous_freq_table=function(x,n){
  differ<-max(x)-min(x)
  int_differ=differ/n
  limit<-rep(0,n+1)
  limit[1]=min(x)
  for(i in 2:(n+1)){
    limit[i]=limit[1]+((i-1)*int_differ)
  }
  limit
  lower_limit<-limit[1:n]
  upper_limit<-limit[2:(n+1)]
  class_limit<-data.frame(lower_limit,upper_limit)
  class_limit
  freq<-rep(0,n)
  freq[1]=sum(x<=upper_limit[1])
  for(i in 2:n){
    freq[i]=sum(x<=upper_limit[i])-sum(freq)
  }
  freq
  cum_freq<-rep(0,10)
  cum_freq=freq[1]
  for(i in 2:n){
    cum_freq[i]=freq[i]+cum_freq[i-1]
  }
  cum_freq
  class_mark<-rep(0,n)
  for(i in 1:n){
    class_mark[i]=((lower_limit[i]+upper_limit[i])/2)
  }
  class_mark
  m<-rep(0,n)
  for(i in 1:n){
    m[i]=class_mark[i]*freq[i]
  }
  m
  m1=sum(m)/sum(freq)
  return(m1)
```

```
}
x<-c(10.2,0.5,5.2,6.1,3.1,6.7,8.9,7.2,8.9,5.4,3.6,9.2,6.1,7.3,2.0,1.3,6.4,8.0
,4.3,4.7,12.4,8.6,13.1,3.2,9.5,7.6,4.0,5.1,8.1,1.1,11.5,3.1,6.8,7.0,8.2,2.0,3
.1,6.5,11.2,12.0,5.1,10.9,11.2,8.5,2.3,3.4,5.2,10.7,4.9,6.2)
n<-10
continuous_freq_table(x,n)

[1] 6.5984
```

Alternative (Table creation)

```
continuous_freq_table1=function(x,n){
  differ<-max(x)-min(x)
  int_differ=differ/n
  limit<-rep(0,n+1)
  limit[1]=min(x)
  for(i in 2:(n+1)){
    limit[i]=limit[1]+((i-1)*int_differ)
  }
  limit
  lower_limit<-limit[1:n]
  upper_limit<-limit[2:(n+1)]
  class_limit<-data.frame(lower_limit,upper_limit)
  class_limit
  freq<-rep(0,n)
  freq[1]=sum(x<=upper_limit[1])
  for(i in 2:n){
    freq[i]=sum(x<=upper_limit[i])-sum(freq)
  }
  freq
  cum_freq<-rep(0,10)
  cum_freq=freq[1]
  for(i in 2:n){
    cum_freq[i]=freq[i]+cum_freq[i-1]
  }
  cum_freq
  class_mark=(lower_limit+upper_limit)/2
  class_mark
  tab=data.frame(lower_limit,upper_limit,class_mark,freq,cum_freq)
  return(tab)
}
x<-c(10.2,0.5,5.2,6.1,3.1,6.7,8.9,7.2,8.9,5.4,3.6,9.2,6.1,7.3,2.0,1.3,6.4,8.0
,4.3,4.7,12.4,8.6,13.1,3.2,9.5,7.6,4.0,5.1,8.1,1.1,11.5,3.1,6.8,7.0,8.2,2.0,3
.1,6.5,11.2,12.0,5.1,10.9,11.2,8.5,2.3,3.4,5.2,10.7,4.9,6.2)
n<-10
continuous_freq_table1(x,n)

   lower_limit upper_limit class_mark freq cum_freq
1         0.50        1.76       1.13    3        3
2         1.76        3.02       2.39    3        6
3         3.02        4.28       3.65    7       13
```

```
4          4.28         5.54       4.91    8        21
5          5.54         6.80       6.17    7        28
6          6.80         8.06       7.43    5        33
7          8.06         9.32       8.69    7        40
8          9.32        10.58       9.95    2        42
9         10.58        11.84      11.21    5        47
10        11.84        13.10      12.47    3        50
```

```
Continuous_Mean=function(x,f){
  sum=0
  n=sum(f)
  for(i in 1:length(x)) {
    sum=sum+(x[i]*f[i])
  }
  m=(sum/n)
  return(m)
}
x<-c(10.2,0.5,5.2,6.1,3.1,6.7,8.9,7.2,8.9,5.4,3.6,9.2,6.1,7.3,2.0,1.3,6.4,8.0
,4.3,4.7,12.4,8.6,13.1,3.2,9.5,7.6,4.0,5.1,8.1,1.1,11.5,3.1,6.8,7.0,8.2,2.0,3
.1,6.5,11.2,12.0,5.1,10.9,11.2,8.5,2.3,3.4,5.2,10.7,4.9,6.2)
n<-10
continuous_freq_table1(x,n)
```

```
   lower_limit upper_limit class_mark freq cum_freq
1         0.50        1.76       1.13    3        3
2         1.76        3.02       2.39    3        6
3         3.02        4.28       3.65    7       13
4         4.28        5.54       4.91    8       21
5         5.54        6.80       6.17    7       28
6         6.80        8.06       7.43    5       33
7         8.06        9.32       8.69    7       40
8         9.32       10.58       9.95    2       42
9        10.58       11.84      11.21    5       47
10       11.84       13.10      12.47    3       50
```

```
Continuous_Mean(continuous_freq_table1(x,n)[,3],continuous_freq_table1(x,n)[,
4])
```

```
[1] 6.5984
```

Property 1

Arithmetic sum of the deviations of a set of values from their arithmetic mean is 0.

```
x=c(1,5,2,1,3,2,5,2,6)
y=distinct_1(x)
f=Freq_1(x)
sum=0
for(i in 1:length(f)){
  sum=sum+f[i]*(y[i]-Mean_1(x))
```

```
}
sum

[1] 0

Conclusion -So, it is proved that, arithmetic sum of the deviations of a set of values from their arithmetic mean is 0.
```

Property 2

The sum of the squares of the deviations of a set of values is minimum when taken about mean.

```
x=c(1,5,2,1,3,2,5,2,6)
y=distinct_1(x)
f=Freq_1(x)
m=Mean_1(x)
sum1=0
for(i in 1:length(y)){
  sum1=sum1+f[i]*(y[i]-m)^2
}
sum1

[1] 28

#lets take some a=2
sum2=0
a=2
for(i in 1:length(y)){
  sum2=sum2+f[i]*(y[i]-a)^2
}
sum2

[1] 37

#So , it is proved that , the sum of the squares of the deviations of a set of values is minimum when taken about mean.
```

Geometric Mean

```
G.mean._1=function(x){
  y=distinct_1(x)
  f=Freq_1(x)
  N=sum(f)
  p=1
  for(i in 1:length(f)){
    p=(p*(y[i]^f[i]))
  }
  g=(p)^(1/N)
  return(g)
}
```

```
x=c(2,2,3,2,2,3)
G.mean._1(x)

[1] 2.289428
```

Alternative

```
geometric.mean=function(x){
  a=1
  for(i in 1:length(x)){
    a=a*x[i]
  }
  a
  geometric_mean=a^(1/length(x))
  geometric_mean
  return(geometric_mean)
}
x=c(1,3,2,4,4,6,2,2,2,2,2,4,3)
geometric.mean(x)

[1] 2.577122
```

Harmonic Mean

```
H.mean._1=function(x){
  y=distinct_1(x)
  f=Freq_1(x)
  N=sum(f)
  sum=0
  for(i in 1:length(f)){
    sum=sum+(f[i]/y[i])
  }
  h=(N/sum)
  return(h)
}
x=c(5,7,5,4,4,4,7,5,4)
H.mean._1(x)

[1] 4.772727
```

Alternative

```
x=c(1,3,2,4,4,6,2,2,2,2,2,4,3)
harmonic.mean=function(x){
  p=1/x
  p
  sum=0
  for(i in 1:length(x)){
    sum=sum+p[i]
  }
  sum
```

```
  harmonic_mean=length(x)/sum
  harmonic_mean
  return(harmonic_mean)
}
harmonic.mean(x)

[1] 2.328358
```

Median

Discrete frequency distribution

```
discrete_median=function(x){
  sorted_data=sort(x)
  if(length(x)%%2==0){
    median_1=sorted_data[(length(x)/2)]
    median_2=sorted_data[(length(x)/2)+1]
    median_interval=c(median_1,median_2)
    print("median interval for even number of data")
    return(median_interval)
  }else{
    m=sorted_data[(length(x)+1)/2]
    z2=paste("median for odd numbers of data is ",m)
    return(z2)
  }
}
x<-c(42,74,40,60,82,115,41,61,75,83,63,53,110,76,84,50,67,65,78,77,56,95,68,6
9,104,80,79,79,54,73,59,81,100,66,49,77,90,84,76,42,64,69,70,80,72,50,79,52,1
03,96,51,86,78,94,71)
discrete_median(x)

[1] "median for odd numbers of data is  74"
```

Alternative

```
x=c(1,3,2,4,4,6,2,2,2,2,2,4,3)
median1=function(x){
  xu=unique(x) ## to count number of distinct elements
  xu
  xus=sort(xu)
  xus
  n=length(xus)
  freq=xus*0
  for(i in 1:length(xus)){
    freq[i]=sum(x==xus[i]) ## counting frequency for ith distinct value
  }
  freq
  cumfq=rep(0,n)
  cumfq[1]=freq[1]
  for(i in 2:n){
```

```
    cumfq[i]=freq[i]+cumfq[i-1]
  }
  cumfq
  N=sum(freq)
  med=N/2
  b=min(which(cumfq>med))
  m=xus[b]
  return(m)
}
median1(x)

[1] 2
```

Continous frequency distribution

```
continous_median_1=function(x){
  differ=max(x)-min(x)
  int_differ=differ/n
  limit<-rep(0,(n+1))
  limit[1]=min(x)
  for(i in 2:(n+1)){
    limit[i]=limit[1]+(i-1)*int_differ
  }
  limit
  lower_limit=limit[1:n]
  upper_limit=limit[2:(n+1)]
  freq<-rep(0,n)
  freq[1]=sum(x<=upper_limit[1])
  for(i in 2:n){
    freq[i]=sum(x<=upper_limit[i])-sum(freq)
  }
  freq
  cum_freq=rep(0,n)
  cum_freq[1]=freq[1]
  for(i in 2:n){
    cum_freq[i]=cum_freq[i-1]+freq[i]
  }
  cum_freq
  tab=data.frame(lower_limit,upper_limit,freq,cum_freq)
  tab
  med=length(x)/2
  z=which(cum_freq>=med)
  u=z[1]
  cum_freq[0]=0
  m=lower_limit[u]+((upper_limit[u]-lower_limit[u])/freq[u])*(med-cum_freq[u-
1])
  return(m)
}
x<-c(10.2,0.5,5.2,6.1,3.1,6.7,8.9,7.2,8.9,5.4,3.6,9.2,6.1,7.3,2.0,1.3,6.4,8.0
,4.3,4.7,12.4,8.6,13.1,3.2,9.5,7.6,4.0,5.1,8.1,1.1,11.5,3.1,6.8,7.0,8.2,2.0,3
```

```
.1,6.5,11.2,12.0,5.1,10.9,11.2,8.5,2.3,3.4,5.2,10.7,4.9,6.2)
n<-10
continous_median_1(x)

[1] 6.26
```

Alternative

```
Continuous_Median_2=function(x,f,cf,l,u){
  N=sum(f)/2
  z=which(cf>=N)
  b=z[1]
  median1=l[b]+((u[b]-l[b])/f[b])*(N-cf[b-1])
  return(median1)
}
x<-c(10.2,0.5,5.2,6.1,3.1,6.7,8.9,7.2,8.9,5.4,3.6,9.2,6.1,7.3,2.0,1.3,6.4,8.0
,4.3,4.7,12.4,8.6,13.1,3.2,9.5,7.6,4.0,5.1,8.1,1.1,11.5,3.1,6.8,7.0,8.2,2.0,3
.1,6.5,11.2,12.0,5.1,10.9,11.2,8.5,2.3,3.4,5.2,10.7,4.9,6.2)
n<-10
continuous_freq_table1(x,n)

   lower_limit upper_limit class_mark freq cum_freq
1         0.50        1.76       1.13    3        3
2         1.76        3.02       2.39    3        6
3         3.02        4.28       3.65    7       13
4         4.28        5.54       4.91    8       21
5         5.54        6.80       6.17    7       28
6         6.80        8.06       7.43    5       33
7         8.06        9.32       8.69    7       40
8         9.32       10.58       9.95    2       42
9        10.58       11.84      11.21    5       47
10       11.84       13.10      12.47    3       50

Continuous_Median_2(continuous_freq_table1(x,n)[,"class_mark"],continuous_fre
q_table1(x,n)[,"freq"],continuous_freq_table1(x,n)[,"cum_freq"],continuous_fr
eq_table1(x,n)[,"lower_limit"],continuous_freq_table1(x,n)[,"upper_limit"])

[1] 6.26
```

1ST 2ND AND 3RD QUARTILE

Discrete (Ungrouped data)

```
quartile_1=function(x){
  y =sort(x)
  y
  Q1=0
  Q2=0
  Q3=0
  n=length(x)
```

```r
  if((n%%2)==0){
    Q2=((y[n/2]+y[(n/2)+1])/2)
  }else {
    Q2=y[(n+1)/2]
  }
  m=(n+1)/4
  if((n+1)%%4==0){
    Q1=y[m]
  }else{
    Q1=y[floor(m)]+(m-floor(m))*(y[ceiling(m)]-y[floor(m)])
  }
  p=3*((n+1)/4)
  if(3*(n+1)%%4==0){
    Q3=y[p]
  }else{
    Q3=y[floor(p)]+(p-floor(p))*(y[ceiling(p)]-y[floor(p)])
  }

  q=c(Q1,Q2,Q3)
  return(q)
}
x=c(18,7,15,27,22,20,24,27,30,12)
quartile_1(x)

[1] 14.25 21.00 27.00
```

Alternative

```r
quartile_1=function(x){
  y=sort(x)
  n=length(y)
  if((n%%2)==0){
    Q2=((y[n/2]+y[(n/2)+1])/2)
    Q1=y[n/4]
    Q3=y[3*n/4]
  }else{
    Q2=y[(n+1)/2]
    Q1=y[(n+1)/4]
    Q3=y[3*(n+1)/4]
  }
  q=c(Q1,Q2,Q3)
  return(q)
}
x=c(1,5,2,6,9,8,7,3,0,10,12,20)
quartile_1(x)

[1] 2.0 6.5 9.0
```

Quartile for grouped data

```
q_contineous=function(x,n){
  Range=max(x)-min(x)
  Range
  d=Range/n
  limit=rep(0,(n+1))
  limit[1]=min(x)
  for(i in 2:(n+1)){
    limit[i]=limit[1]+((i-1)*d)
  }
  limit
  lower_limit=limit[1:n]
  upper_limit=limit[2:(n+1)]
  class_interval=data.frame(lower_limit,upper_limit)
  class_interval
  frequency=rep(0,n)
  frequency[1]=sum(x<=upper_limit[1])
  frequency[1]
  for(i in 2:n){
    frequency[i]=sum(x<=upper_limit[i])-sum(frequency)
  }
  frequency
  class_mark=rep(0,d)
  for(i in 1:d){
    class_mark[i]=((lower_limit[i]+upper_limit[i])/2)
  }
  class_mark
  cumfq=rep(0,n)
  cumfq[1]=frequency[1]
  for(i in 2:n){
    cumfq[i]=frequency[i]+cumfq[i-1]
  }
  cumfq
  N=sum(frequency)
  if((N%%2)==0){
    a=N/4
  }else{
    a=(N+1)/4
  }
  if((N%%2)==0){
    b=3*N/4
  }else{
    Q3=3*(N+1)/4
  }
  i=min(which(cumfq>=a))
  j=min(which(cumfq>=b))
  Q1=lower_limit[i]+((a-cumfq[i-1])/frequency[i])*(upper_limit[i]-lower_limit
[i])
  Q3=lower_limit[j]+((b-cumfq[j-1])/frequency[j])*(upper_limit[j]-lower_limit
```

```
[j])
  return(c(Q1,Q3))
}
x=rep(c(1,5,6,8,6,4,2),c(7,9,15,23,21,10,5))
n=5
q_contineous(x,n)

[1] 4.573684 6.630435
```

Alternate

```
Continuous_Quartile=function(x,f,cf,l,u){
  N=sum(f)
  if((N%%2)==0){
    a=N/4
  }else{
    a=(N+1)/4
  }
  if((N%%2)==0){
    b=3*N/4
  }else{
    Q3=3*(N+1)/4
  }
  i=min(which(cf>=a))
  j=min(which(cf>=b))
  Q1=l[i]+((a-cf[i-1])/f[i])*(u[i]-l[i])
  Q3=l[j]+((b-cf[j-1])/f[j])*(u[j]-l[j])
  return(c(Q1,Q3))
}
x=rep(c(1,5,6,8,6,4,2),c(7,9,15,23,21,10,5))
n=5
continuous_freq_table1(x,n)

  lower_limit upper_limit class_mark freq cum_freq
1         1.0         2.4        1.7   12       12
2         2.4         3.8        3.1    0       12
3         3.8         5.2        4.5   19       31
4         5.2         6.6        5.9   36       67
5         6.6         8.0        7.3   23       90

Continuous_Quartile(continuous_freq_table1(x,n)[,"class_mark"],continuous_fre
q_table1(x,n)[,"freq"],continuous_freq_table1(x,n)[,"cum_freq"],continuous_fr
eq_table1(x,n)[,"lower_limit"],continuous_freq_table1(x,n)[,"upper_limit"])

[1] 4.573684 6.630435
```

Percentile

Discrete Percentile

```
Percentile_1=function(x,p) {
  y=sort(x)
  y
  z_p=0
  n=length(x)
  m=((n+1)*p)/100
  if(m%%2==0){
    z_p=y[m]
  }else{
    z_p=y[floor(m)]+(m-floor(m))*(y[ceiling(m)]-y[floor(m)])
  }
  return(z_p)
}
x=c(80,81,76,73,85,88,79,68,89)
p=62
Percentile_1(x,p)

[1] 81.8
```

Percentile for Grouped data

```
percentile_contineous=function(x,n,j){
  Range=max(x)-min(x)
  Range
  d=Range/n
  limit=rep(0,(n+1))
  limit[1]=min(x)
  for(i in 2:(n+1)){
    limit[i]=limit[1]+((i-1)*d)
  }
  limit
  lower_limit=limit[1:n]
  upper_limit=limit[2:(n+1)]
  class_interval=data.frame(lower_limit,upper_limit)
  class_interval
  frequency=rep(0,n)
  frequency[1]=sum(x<=upper_limit[1])
  frequency[1]
  for(i in 2:n){
    frequency[i]=sum(x<=upper_limit[i])-sum(frequency)
  }
  frequency
  class_mark=rep(0,n)
```

```
  for(i in 1:n){
    class_mark[i]=((lower_limit[i]+upper_limit[i])/2)
  }
  class_mark
  cumfq=rep(0,n)
  cumfq[1]=frequency[1]
  for(i in 2:n){
    cumfq[i]=frequency[i]+cumfq[i-1]
  }
  cumfq
  a=sum(frequency)
  p_j=(j*a)/100
  b=min(which(cumfq>=p_j))
  p=lower_limit[b]+((p_j-cumfq[b-1])/frequency[b])*(upper_limit[b]-lower_limi
t[b])
  return(p)
}
x=rep(c(1:10),c(2,3,4,5,6,3,8,9,4,2))
n=5
j=99
percentile_contineous(x,n,j)

[1] 9.862
```

Alternate

```
continuous_percentile=function(x,f,cf,l,u,j){
  a=sum(f)
  p_j=(j*a)/100
  b=min(which(cf>=p_j))
  p=l[b]+((p_j-cf[b-1])/f[b])*(u[b]-l[b])
  return(p)
}
x=rep(c(1:10),c(2,3,4,5,6,3,8,9,4,2))
n=5
j=99
continuous_freq_table1(x,n)

  lower_limit upper_limit class_mark freq cum_freq
1         1.0         2.8        1.9    5        5
2         2.8         4.6        3.7    9       14
3         4.6         6.4        5.5    9       23
4         6.4         8.2        7.3   17       40
5         8.2        10.0        9.1    6       46

continuous_percentile(continuous_freq_table1(x,n)[,"class_mark"],continuous_f
req_table1(x,n)[,"freq"],continuous_freq_table1(x,n)[,"cum_freq"],continuous_
freq_table1(x,n)[,"lower_limit"],continuous_freq_table1(x,n)[,"upper_limit"],
j)

[1] 9.862
```

Mode

Discrete frequency distribution

```
mode_1=function(x){
  xu=unique(x) ## to count number of distinct elements
  xu
  xus=sort(xu)
  xus
  freq=xus*0
  for(i in 1:length(xus)){
    freq[i]=sum(x==xus[i]) ## counting frequency for ith distinct value
  }
  freq
  freq_max=max(freq)
  i=which(freq==freq_max)
  m=xus[i]
  m
  return(m)
}
x<-c(42,74,40,60,82,115,41,61,75,83,63,53,110,76,84,50,67,65,78,77,56,95,68,6
9,104,80,79,79,54,73,59,81,100,66,49,77,90,84,76,42,64,69,70,80,72,50,79,52,1
03,96,51,86,78,94,71)
mode_1(x)

[1] 79
```

Alternative

```
Mode_2=function(x){
  y=sort(distinct_1(x))
  m=y[which(Freq_1(x)==max(Freq_1(x)))]
  return(m)
}
x=c(1,3,2,4,4,6,2,2,2,2,2,4,3)
Mode_2(x)

[1] 2
```

Mode for continuous frequency distribution

```
mode2_freq_tab<-function(x,n){
  differ=max(x)-min(x)
  int_differ=differ/n
  limit<-rep(0,n+1)
  limit[1]=min(x)
  for(i in 2:(n+1)){
    limit[i]=limit[1]+(i-1)*int_differ
  }
  limit
```

```
  lower_limit=limit[1:n]
  upper_limit=limit[2:(n+1)]
  freq=rep(0,n)
  freq[1]=sum(x<=upper_limit[1])
  for(i in 2:n){
    freq[i]=sum(x<=upper_limit[i])-sum(freq)
  }
  freq
  u=which(freq==max(freq))
  u
  tab=data.frame(lower_limit,upper_limit,freq)
  tab
  freq[n+1]=0
  mode1=lower_limit[u]+((upper_limit[u]-lower_limit[u])*((freq[u]-freq[u-1])/
(2*freq[u]-freq[u-1]-freq[u+1])))
  if(mode1>=lower_limit[u]){
    if(mode1<=upper_limit[u]){
      return(mode1)
    }else{
      mode1=lower_limit[u]+((upper_limit[u]-lower_limit[u])*((freq[u]-freq[u-
1])/((abs(freq[u]-freq[u-1]))+abs(freq[u]-freq[u+1]))))
      return(mode1)
    }
  }else{
    mode1=lower_limit[u]+((upper_limit[u]-lower_limit[u])*((freq[u]-freq[u-1]
)/((abs(freq[u]-freq[u-1]))+abs(freq[u]-freq[u+1]))))
    return(mode1)
  }
}
x<-c(10.2,0.5,5.2,6.1,3.1,6.7,8.9,7.2,8.9,5.4,3.6,9.2,6.1,7.3,2.0,1.3,6.4,8.0
,4.3,4.7,12.4,8.6,13.1,3.2,9.5,7.6,4.0,5.1,8.1,1.1,11.5,3.1,6.8,7.0,8.2,2.0,3
.1,6.5,11.2,12.0,5.1,10.9,11.2,8.5,2.3,3.4,5.2,10.7,4.9,6.2)
n<-10
mode2_freq_tab(x,n)

[1] 4.91
```

Alternative

```
mode_contineous.data=function(x,n){
  Range=max(x)-min(x)
  Range
  d=Range/n
  d
  limit=rep(0,(n+1))
  limit[1]=min(x)
  for(i in 2:(n+1)){
    limit[i]=limit[1]+((i-1)*d)
  }
  limit
```

```
  lower_limit=limit[1:n]
  upper_limit=limit[2:(n+1)]
  class_interval=data.frame(lower_limit,upper_limit)
  class_interval
  frequency=rep(0,n)
  frequency[1]=sum(x<=upper_limit[1])
  frequency[1]
  for(i in 2:n){
    frequency[i]=sum(x<=upper_limit[i])-sum(frequency)
  }
  frequency
  class_mark=rep(0,n)
  for(i in 1:n){
    class_mark[i]=((lower_limit[i]+upper_limit[i])/2)
  }
  class_mark
  cumfq=rep(0,n)
  cumfq[1]=frequency[1]
  for(i in 2:n){
    cumfq[i]=frequency[i]+cumfq[i-1]
  }
  cumfq
  N=max(frequency)
  mode=0
  b=which(frequency==N)
  m=lower_limit[b]+((frequency[b]-frequency[b-1])/((2*frequency[b])-frequency
[b-1]-frequency[b+1]))
  m
  return(m)
}
x=rep(c(1,5,6,8,6,4,2),c(7,9,15,23,21,10,9))
n=5
mode_contineous.data(x,n)

[1] 5.766667
```

Alternative

```
Continuous_Mode2=function(x,f,l,u){
  N=max(f)
  mode1=0
  b=which(f==N)
  mode1=l[b]+((f[b]-f[b-1])/((2*f[b])-f[b-1]-f[b+1]))
  return(mode1)
}
x<-c(10.2,0.5,5.2,6.1,3.1,6.7,8.9,7.2,8.9,5.4,3.6,9.2,6.1,7.3,2.0,1.3,6.4,8.0
,4.3,4.7,12.4,8.6,13.1,3.2,9.5,7.6,4.0,5.1,8.1,1.1,11.5,3.1,6.8,7.0,8.2,2.0,3
.1,6.5,11.2,12.0,5.1,10.9,11.2,8.5,2.3,3.4,5.2,10.7,4.9,6.2)
n<-10
continuous_freq_table1(x,n)
```

```
   lower_limit upper_limit class_mark freq cum_freq
1         0.50        1.76       1.13    3        3
2         1.76        3.02       2.39    3        6
3         3.02        4.28       3.65    7       13
4         4.28        5.54       4.91    8       21
5         5.54        6.80       6.17    7       28
6         6.80        8.06       7.43    5       33
7         8.06        9.32       8.69    7       40
8         9.32       10.58       9.95    2       42
9        10.58       11.84      11.21    5       47
10       11.84       13.10      12.47    3       50
```

```
Continuous_Mode2(continuous_freq_table1(x,n)[,"class_mark"],continuous_freq_t
able1(x,n)[,"freq"],continuous_freq_table1(x,n)[,"lower_limit"],continuous_fr
eq_table1(x,n)[,"upper_limit"])
```

```
[1] 4.78
```

MEASURES OF DISPERSION

Range –

```
x=c(2,13,4,4,8,11,16,54,6)
function_1=function(x){
  xs=sort(x)
  min=xs[1]
  max=xs[length(x)]
  range=max-min
  return(range)
}
function_1(x)
```

```
[1] 52
```

Alternate

```
Max_sr=function(x){
  l=length(x)
  a=sort(x)
  b=a[l]
  return(b)
}
Max_sr(x)
```

```
[1] 54
```

```
Min_sr=function(x){
  a=sort(x)
  b=a[1]
  return(b)
```

```
}
Min_sr(x)

[1] 2

#Creating a function for Range
Range_sr=function(x){
  r=Max_sr(x)-Min_sr(x)
  return(r)
}
Range_sr(x)

[1] 52
```

Quartile deviation-

For an ungrouped data, quartiles can be obtained using the following formulas,

Quartile Deviation is given by=(Q3-Q1)/2 , where Q1 and Q3 are the 1st & 3rd quartiles of the distribution respectively

For n=odd

Q1=[(n+1)/4]th item

Q2=[(n+1)/2]th item

Q3=[3*(n+1)/4]th term

For n=even

Q1=[n/4]th term

Q2=[((n/2)+(n/2)+1)/2]th term

Q3=[3*n/4]th term

Quartile for ungrouped data

```
x=c(1,5,2,6,9,8,7,3,0,10,12,20)
quartile_deviation=function(x){
  y=sort(x)
  n=length(y)
  if((n%%2)==0){
    Q2=((y[n/2]+y[(n/2)+1])/2)
    Q1=y[n/4]
    Q3=y[3*n/4]
  }else{
    Q2=y[(n+1)/2]
    Q1=y[(n+1)/4]
    Q3=y[3*(n+1)/4]
```

```
  }
  q_deviation=(Q3-Q1)/2
  return(q_deviation)
}
quartile_deviation(x)

[1] 3.5
```

Quartile for grouped data

Q1=l+(Q_1-cf[-1])*(h/f)

Q3=l+(Q_3-cf[-1])*(h/f)

For N=odd

Q_1=[(N+1)/4]th item

Q_3=[3*(N+1)/4]th item

For N=even

Q_1=[(N+1)/4]th item

Q_3=[3*N/4]th item

l=lower limit of quartile class

h=Width of the class

f=frequency of the class

cf[-1]=cumulative frequency of previous class

```
x=rep(c(1,5,6,8,6,4,2),c(7,9,15,23,21,10,5))
n=5
table_q=function(x,n){
  Range=max(x)-min(x)
  Range
  d=Range/n
  limit=rep(0,(n+1))
  limit[1]=min(x)
  for(i in 2:(n+1)){
    limit[i]=limit[1]+((i-1)*d)
  }
  limit
  lower_limit=limit[1:n]
  upper_limit=limit[2:(n+1)]
  class_interval=data.frame(lower_limit,upper_limit)
  class_interval
  frequency=rep(0,n)
  frequency[1]=sum(x<=upper_limit[1])
  frequency[1]
```

```r
  for(i in 2:n){
    frequency[i]=sum(x<=upper_limit[i])-sum(frequency)
  }
  frequency
  class_mark=rep(0,d)
  for(i in 1:d){
    class_mark[i]=((lower_limit[i]+upper_limit[i])/2)
  }
  class_mark
  cumfq=rep(0,n)
  cumfq[1]=frequency[1]
  for(i in 2:n){
    cumfq[i]=frequency[i]+cumfq[i-1]
  }
  cumfq
  table=data.frame(class_interval,class_mark,frequency,cumfq)
  return(table)
}
table_q(x,n)
```

```
  lower_limit upper_limit class_mark frequency cumfq
1         1.0         2.4        1.7        12    12
2         2.4         3.8        1.7         0    12
3         3.8         5.2        1.7        19    31
4         5.2         6.6        1.7        36    67
5         6.6         8.0        1.7        23    90
```

```r
quartile_deviation2=function(x,f,l,u,cf){
  N=sum(f)
  if((N%%2)==0){
    a=N/4
  }else{
    a=(N+1)/4
  }
  if((N%%2)==0){
    b=3*N/4
  }else{
    Q3=3*(N+1)/4
  }
  i=min(which(cf>=a))
  j=min(which(cf>=b))
  Q1=l[i]+((a-cf[i-1])/f[i])*(u[i]-l[i])
  Q3=l[j]+((b-cf[j-1])/f[j])*(u[j]-l[j])
  q_deviation=(Q3-Q1)/2
  return(q_deviation)
}
quartile_deviation2(table_q(x,n)[,"class_mark"],table_q(x,n)[,"frequency"],ta
ble_q(x,n)[,"lower_limit"],table_q(x,n)[,"upper_limit"],table_q(x,n)[,"cumfq"
])
```

```
[1] 1.028375
```

Mean Deviation

Discrete frequency distribution

```
x<-c(42,74,40,60,82,115,41,61,75,83,63,53,110,76,84,50,67,65,78,77,56,95,68,6
9,104,80,79,79,54,73,59,81,100,66,49,77,90,84,76,42,64,69,70,80,72,50,79,52,1
03,96,51,86,78,94,71)
a<-40
mean_deviation_data=function(x,a){
  sum=0
  sorted_data=sort(unique(x))
  freq=rep(0,length(sorted_data))
  for(i in 1:length(sorted_data)){
    freq[i]=sum(x==sorted_data[i])
  }
  freq
  mean_deviation=0
  for(i in 1:length(sorted_data)){
    mean_deviation=abs((sorted_data[i]-a)*freq[i])+mean_deviation
  }
  mean_point=mean_deviation/sum(freq)
  return(mean_point)
}
mean_deviation_data(x,a)

[1] 32.58182
```

Continuous frequency distribution

```
x<-c(10.2,0.5,5.2,6.1,3.1,6.7,8.9,7.2,8.9,5.4,3.6,9.2,6.1,7.3,2.0,1.3,6.4,8.0
,4.3,4.7,12.4,8.6,13.1,3.2,9.5,7.6,4.0,5.1,8.1,1.1,11.5,3.1,6.8,7.0,8.2,2.0,3
.1,6.5,11.2,12.0,5.1,10.9,11.2,8.5,2.3,3.4,5.2,10.7,4.9,6.2)
n<-10
a<-40
mean_deviation_continuous_data=function(x,n,a){
  int_differ=max(x)-min(x)
  int=int_differ/n
  limit=rep(0,(n+1))
  limit[1]=min(x)
  for(i in 1:(n+1)){
    limit[i]=limit[1]+(i-1)*int
  }
  limit
  lower_limit=limit[1:10]
  upper_limit=limit[2:11]
  freq=rep(0,n)
  freq[1]=sum(x<=upper_limit[1])
```

```r
  for(i in 2:n){
    freq[i]=sum(x<=upper_limit[i])-sum(freq)
  }
  freq
  class_mark=rep(0,n)
  for(i in 1:n){
    class_mark[i]=((lower_limit[i]+upper_limit[i])/2)
  }
  class_mark
  mean_deviation=rep(0,n)
  for(i in 1:n){
    mean_deviation[i]=(abs((class_mark[i]-a)*freq[i]))
  }
  mean_deviation
  tab=data.frame(lower_limit,upper_limit,class_mark,freq,mean_deviation)
  return(tab)
}
mean_deviation_continuous_data(x,n,a)
```

```
   lower_limit upper_limit class_mark freq mean_deviation
1         0.50        1.76       1.13    3         116.61
2         1.76        3.02       2.39    3         112.83
3         3.02        4.28       3.65    7         254.45
4         4.28        5.54       4.91    8         280.72
5         5.54        6.80       6.17    7         236.81
6         6.80        8.06       7.43    5         162.85
7         8.06        9.32       8.69    7         219.17
8         9.32       10.58       9.95    2          60.10
9        10.58       11.84      11.21    5         143.95
10       11.84       13.10      12.47    3          82.59
```

```r
mean_deviation_value=function(mean_deviation,freq){
  mean=sum(mean_deviation)/sum(freq)
  return(mean)
}
mean_deviation_value(mean_deviation_continuous_data(x,n,a)[,"mean_deviation"]
,mean_deviation_continuous_data(x,n,a)[,"freq"])
```

```
[1] 33.4016
```

Alternate (Absolute)

```r
Abs_sr=function(x){
  a=x^2
  b=sqrt(a)
  return(b)
}
Abs_sr(-6)
```

```
[1] 6
```

Mean deviation about some constant 'a'.

```
distinct_sr=function(x){
  y=x[1]
  for(i in 2:length(x)){
    if(sum(x[1:(i-1)]==x[i])==0)
      y=c(y,x[i])
  }
  return(sort(y))
}
Freq_sr=function(x){
  a=distinct_sr(sort(x))
  sum=rep(0,length(a))
  for(i in 1:length(a)){
    for(j in 1:length(x)){
      if(a[i]==x[j]){
        sum[i]=sum[i]+1
      }
    }

  }
  return(sum)
}
x=c(1,1,1,2,8,7,3,4,2,3,2,4)
Freq_sr(x)

[1] 3 3 2 2 1 1

MD_a_sr=function(x,a){
  n=length(distinct_sr(x))
  y=distinct_sr(x)
  f=Freq_sr(x)
  N=sum(f)
  sum=0
  for(i in 1:n){
    sum=sum+(Abs_sr(y[i]-a))*f[i]
  }
  m=(sum/N)
  return(m)
}
x=c(1,4,2,8,4,3,6,6,3,2,1)
MD_a_sr(x,5)

[1] 2.272727
```

Mean deviation

```
x=c(0:8)
f=c(1,8,28,56,70,56,28,8,1)
```

```
A=sum(x*f)/sum(f)
mean_deviation=function(x,f,A){
  md=sum((abs(x-A))*f)/sum(f)
  return(md)
}
mean_deviation(x,f,A)

[1] 1.09375
```

Property of mean deviation

Mean deviation is minimum when it is measured about median.

```
cumfq=cumsum(f)
cumfq

[1]   1   9  37  93 163 219 247 255 256

table=cbind(x,f,cumfq)
table

      x  f cumfq
 [1,] 0  1     1
 [2,] 1  8     9
 [3,] 2 28    37
 [4,] 3 56    93
 [5,] 4 70   163
 [6,] 5 56   219
 [7,] 6 28   247
 [8,] 7  8   255
 [9,] 8  1   256

N=sum(f)
N

[1] 256

N/2

[1] 128

median=4
mean_deviation(x,f,median)

[1] 1.09375

for(i in 1:length(x)){
  md=mean_deviation(x,f,i)
  print(md)
}
```

```
[1] 3.007812
[1] 2.078125
[1] 1.367188
[1] 1.09375
[1] 1.367188
[1] 2.078125
[1] 3.007812
[1] 4
[1] 5
```

Conclusion:- Here we have seen that mean deviation is minimum when it is meas ured about median.

Mean Square Deviation

```
x=c(0:8)
f=c(1,8,28,56,70,56,28,8,1)
A=sum(x*f)/sum(f)
mean_sq_deviation=function(x,f,A){
  m_sq_d=sum(((x-A)^2)*f)/sum(f)
  return(m_sq_d)
}
mean_sq_deviation(x,f,A)
```

```
[1] 2
```

Mean Square Deviation is minimum when it is measured about mean.

```
mean=sum(x*f)/sum(f)
mean_sq_deviation(x,f,mean)
```

```
[1] 2
```

```
for(i in 1:length(x)){
  m_sq_d=mean_sq_deviation(x,f,i)
  print(m_sq_d)
}
```

```
[1] 11
[1] 6
[1] 3
[1] 2
[1] 3
[1] 6
[1] 11
[1] 18
[1] 27
```

Conclusion:- Here we have seen that mean square deviation is minimum when it is measured about mean

Rsmd_a

```
RSMD_a_sr=function(x,a){
  y=distinct_sr(x)
  n=length(y)
  f=Freq_sr(x)
  N=sum(f)
  sum=0
  for(i in 1:n){
    sum=sum+((y[i]-a)^2)*f[i]
  }
  m=sqrt(sum/N)
  return(m)
}
x=c(240,241,252,265,279,283,292)
RSMD_a_sr(x,264.57)

[1] 19.33802
```

Variance and Standard Deviation

Variance

```
variance=function(x){
  sum=0
  sum_sq=0
  for(i in 1:length(x)){
    sum=sum+x[i]
    sum_sq=sum_sq+(x[i]*x[i])
  }
  mean=sum/length(x)
  var=(sum_sq/length(x))-((mean)^2)
  var
  return(var)
}
variance(x)

[1] 373.9592
```

Alternate code discrete frequency distribution

```
x<-c(42,74,40,60,82,115,41,61,75,83,63,53,110,76,84,50,67,65,78,77,56,95,68,6
9,104,80,79,79,54,73,59,81,100,66,49,77,90,84,76,42,64,69,70,80,72,50,79,52,1
03,96,51,86,78,94,71)
variance_data=function(x){
  sum=0
  for(i in 1:length(x)){
    sum=(x[i]^2)+sum
  }
```

```
  sum
  sum1=0
  for(i in 1:length(x)){
    sum1=x[i]+sum1
  }
  sum1
  var1=(sum/length(x))-((sum1/length(x))^2)
  var1
  return(var1)
}
variance_data(x)

[1] 308.6797
```

Continous frequency distribution

```
x<-c(10.2,0.5,5.2,6.1,3.1,6.7,8.9,7.2,8.9,5.4,3.6,9.2,6.1,7.3,2.0,1.3,6.4,8.0
,4.3,4.7,12.4,8.6,13.1,3.2,9.5,7.6,4.0,5.1,8.1,1.1,11.5,3.1,6.8,7.0,8.2,2.0,3
.1,6.5,11.2,12.0,5.1,10.9,11.2,8.5,2.3,3.4,5.2,10.7,4.9,6.2)
n<-10
var(x)

[1] 10.505

variance_tab=function(x,n){
  differ=max(x)-min(x)
  int_differ=differ/n
  limit<-rep(0,(n+1))
  limit[1]=min(x)
  for(i in 2:(n+1)){
    limit[i]=limit[1]+(i-1)*int_differ
  }
  limit
  lower_limit<-limit[1:10]
  upper_limit<-limit[2:11]
  freq<-rep(0,n)
  freq[1]=sum(x<=upper_limit[1])
  for(i in 2:n){
    freq[i]=sum(x<=upper_limit[i])-sum(freq)
  }
  freq
  class_mark<-rep(0,n)
  for(i in 1:n){
    class_mark[i]=((lower_limit[i]+upper_limit[i])/2)
  }
  class_mark
  ss=rep(0,n)
  square=rep(0,n)
  for(i in 1:n){
    ss[i]=(class_mark[i]*freq[i])
    square[i]=(class_mark[i]^2)*freq[i]
```

```
  }
  ss
  square
  tab=data.frame(lower_limit,upper_limit,freq,class_mark,ss,square)
  return(tab)
}
variance_tab(x,n)
```

```
   lower_limit upper_limit freq class_mark    ss   square
1         0.50        1.76    3       1.13  3.39   3.8307
2         1.76        3.02    3       2.39  7.17  17.1363
3         3.02        4.28    7       3.65 25.55  93.2575
4         4.28        5.54    8       4.91 39.28 192.8648
5         5.54        6.80    7       6.17 43.19 266.4823
6         6.80        8.06    5       7.43 37.15 276.0245
7         8.06        9.32    7       8.69 60.83 528.6127
8         9.32       10.58    2       9.95 19.90 198.0050
9        10.58       11.84    5      11.21 56.05 628.3205
10       11.84       13.10    3      12.47 37.41 466.5027
```

```
variance_data1=function(square,freq,ss){
  ssquaremean=(sum(square)/sum(freq))-((sum(ss)/sum(freq))^2)
  ssquaremean
  return(ssquaremean)
}
variance_data1(variance_tab(x,n)[,"square"],variance_tab(x,n)[,"class_mark"],
variance_tab(x,n)[,"ss"])
```

```
[1] 15.74034
```

Standard Deviation

```
standard_deviation=function(x){
  sum=0
  sum_sq=0
  for(i in 1:length(x)){
    sum=sum+x[i]
    sum_sq=sum_sq+(x[i]*x[i])
  }
  mean=sum/length(x)
  var=(sum_sq/length(x))-((mean)^2)
  sd=sqrt(var)
  return(sd)
}
standard_deviation(x)
```

```
[1] 3.208566
```

Alternate for Variance

```
Var_1=function(x){
  y=distinct_1(x)
  n=length(y)
  f=Freq_1(x)
  xbar=Mean_1(x)
  N=sum(f)
  sum=0
  for(i in 1:n){
    sum=sum+((y[i]-xbar)^2)*f[i]
  }
  m=(sum/N)
  return(m)
}
x=c(1,3,5)
Var_1(x)

[1] 2.666667
```

Alternate for Standard Deviation

```
sd_1=function(x){
  s=sqrt(Var_1(x))
  return(s)
}
sd_1(x)

[1] 1.632993
```

Coefficient of variation

```
C.v._1=function(x){
  c=(sd_1(x)/Mean_1(x))*100
  return(c)
}
x=c(12,13,14)
C.v._1(x)

[1] 6.280743
```

Coefficient of variation for comparison

```
mean_a=186
mean_b=175
variance_a=81
variance_b=100
firm_vector=c("firm a","firm b")
mean_vector=c(mean_a,mean_b)
variance_vector=c(variance_a,variance_b)
coefficient_variation_data=function(mean_vector,variance_vector,firm_vector){
  coefficient_variation=((sqrt(variance_vector)/mean_vector)*100)
  vector=data.frame(firm_vector,coefficient_variation)
```

```
  consistent_min=vector$firm_vector[which(min(coefficient_variation)==coeffic
ient_variation)]
  print("the given firm is consistent")
  return(consistent_min)
}
coefficient_variation_data(mean_vector,variance_vector,firm_vector)

[1] "the given firm is consistent"

[1] "firm a"
```

Skewness

Skewness (Moments)

Skewness means lack of symmetry. We study to have an idea about the shape of the curve Which can draw with the help of given data.

If beta_1 = 0, given distribution is symmetric

If beta_1 > 0, given distribution is positively skewed

If beta_1 < 0, given distribution is negatively skewed

Method-1

```
moment=function(x,p,r){
  m=sum(x*p)
  mtr=sum(((x-m)^r)*p)
  z=round(mtr,4)
  return(z)
}
```

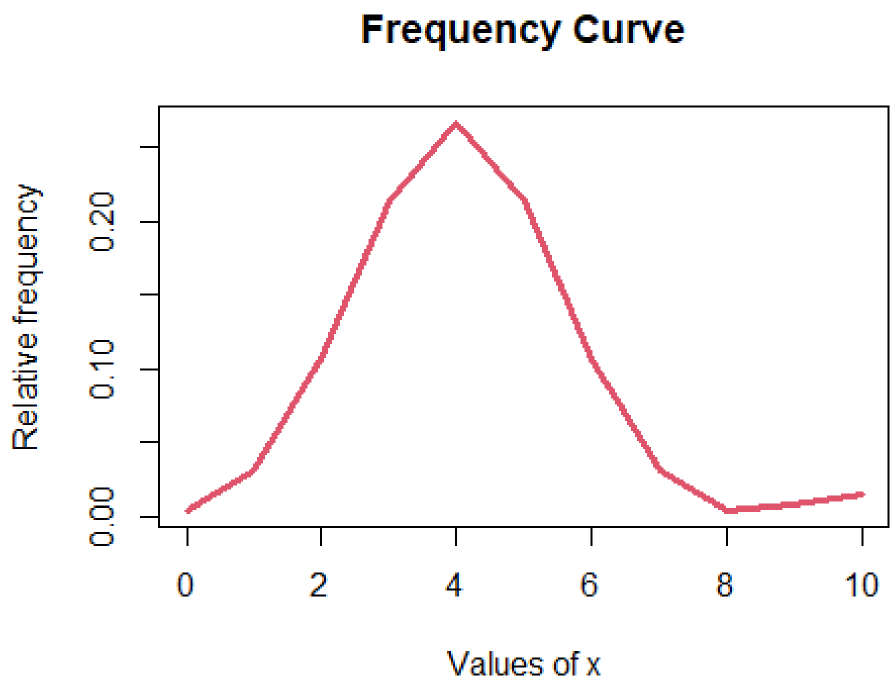
moment(x,p,r)

```
x=c(0:10)
f=c(1,8,28,56,70,56,28,8,1,2,4)
p=f/sum(f)
A=cbind(x,f,p)
A

       x  f           p
 [1,]  0  1 0.003816794
 [2,]  1  8 0.030534351
 [3,]  2 28 0.106870229
 [4,]  3 56 0.213740458
 [5,]  4 70 0.267175573
 [6,]  5 56 0.213740458
 [7,]  6 28 0.106870229
 [8,]  7  8 0.030534351
 [9,]  8  1 0.003816794
```

```
[10,]  9  2 0.007633588
[11,] 10  4 0.015267176
```

Measure of skewness

```
meu_1=moment(x,p,1)
meu_2=moment(x,p,2)
meu_3=moment(x,p,3)
meu_4=moment(x,p,4)
meu2=meu_2-(meu_1)^2
meu2

[1] 2.6778

meu3=meu_3-(3*meu_2*meu_1)+(meu_1)^3
meu3

[1] 3.2072

beta_1=((meu3)^2)/((meu2)^3)
beta_1

[1] 0.5356951

gamma_1=sqrt(beta_1)
gamma_1

[1] 0.7319119

## Since gamma_1>0,then this is a positively skewed
plot(x,p,col=2,type="l",xlab="Values of x",ylab="Relative frequency",main="Fr
equency Curve",lwd=3)
```



Others Measures of Skewness

Karl Pearson's coefficients of skewness

s_k_1=(mean-mode)/sd

Karl Pearson's coefficients of skewness

s_k_2=3*(mean-median)/sd ,where Q2=median

Bowley's measure of skewness

s_k_3=(Q3+Q1-2*median)/(Q3-Q1) ,where Q2=median

```
x=c(18,7,15,27,22,20,24,27,30,12,1)
skewness=function(x){
  sum=0
  for(i in 1:length(x)){
    sum=sum+x[i]
  }
  mean_data=sum/length(x)
  sum_sq=0
  for(i in 1:length(x)){
    sum_sq=sum_sq+(x[i]*x[i])
  }
  var=(sum_sq/length(x))-((mean_data)^2)
  sd=sqrt(var)
  y=sort(x)
```

```
  n=length(y)
  if((n%%2)==0){
    Q2=((y[n/2]+y[(n/2)+1])/2)
    Q1=y[n/4]
    Q3=y[3*n/4]
  }else{
    Q2=y[(n+1)/2]
    Q1=y[(n+1)/4]
    Q3=y[3*(n+1)/4]
  }
  xu=unique(x) ## to count number of distinct elements
  xus=sort(xu)
  freq=xus*0
  for(i in 1:length(xus)){
    freq[i]=sum(x==xus[i]) ## counting frequency for ith distinct value
  }
  freq
  freq_max=max(freq)
  i=which(freq==freq_max)
  mode=xus[i]
  mode
  s_k_1=(mean_data-mode)/sd
  s_k_2=3*(mean_data-Q2)/sd
  s_k_3=(Q3+Q1-2*Q2)/(Q3-Q1)
  s=c(s_k_1,s_k_2,s_k_3)
  if(s_k_1>0){
    print("asymmetrical distribution:positively skewed distribution")
  }else if(s_k_1<0){
    print("asymmetrical distribution:negatively skewed distribution")
  }else{
    print("symmetrical distribution")
  }
  if(s_k_2>0){
    print("asymmetrical distribution:positively skewed distribution")
  }else if(s_k_2<0){
    print("asymmetrical distribution:negatively skewed distribution")
  }else{
    print("symmetrical distribution")
  }
  if(s_k_3>0){
    print("asymmetrical distribution:positively skewed distribution")
  }else if(s_k_3<0){
    print("asymmetrical distribution:negatively skewed distribution")
  }else{
    print("symmetrical distribution")
  }
  return(s)
}
skewness(x)
```

```
[1] "asymmetrical distribution:negatively skewed distribution"
[1] "asymmetrical distribution:negatively skewed distribution"
[1] "asymmetrical distribution:negatively skewed distribution"

[1] -0.99294544 -0.53872572 -0.06666667
```

Method-2

Skewness for discrete data

```
x<-c(42,74,40,60,82,115,41,61,75,83,63,53,110,76,84,50,67,65,78,77,56,95,68,6
9,104,80,79,79,54,73,59,81,100,66,49,77,90,84,76,42,64,69,70,80,72,50,79,52,1
03,96,51,86,78,94,71)
skewness_moments_data=function(x){
  sorted_data=sort(unique(x))
  freq<-rep(0,length(sorted_data))
  for(i in 1:length(sorted_data)){
    freq[i]=sum(x==sorted_data[i])
  }
  freq
  sum=0
  for(i in 1:length(sorted_data)){
    sum=sorted_data[i]*freq[i]+sum
  }
  mean_data=sum/sum(freq)
  sum_mu_four=0
  sum_mu_three=0
  sum_mu_two=0
  for(i in 1:length(sorted_data)){
    sum_mu_four=(((sorted_data[i]-mean_data)^4)*freq[i])+sum_mu_four
    sum_mu_two=(((sorted_data[i]-mean_data)^2)*freq[i])+sum_mu_two
    sum_mu_three=(((sorted_data[i]-mean_data)^3)*freq[i])+sum_mu_three
  }
  sum_mu_four
  sum_mu_three
  sum_mu_two
  beta_two=(sum_mu_four/sum(freq))/(((sum_mu_two)/sum(freq))^2)
  beta_one=((sum_mu_three/sum(freq))^2)/((sum_mu_two/sum(freq))^3)
  skew=(((sqrt(beta_one))*(beta_two+3))/(2*(5*beta_two-6*beta_one-9)))
  if(skew>0){
    print("asymmetrical distribution:positively skewed distribution")
  }else if(skew<0){
    print("asymmetrical distribution:negatively skewed distribution")
  }else{
    print("symmetrical distribution")
  }
  return(skew)
}
skewness_moments_data(x)

[1] "asymmetrical distribution:positively skewed distribution"
```

```
[1] 0.117566
```

Skewness for continuous data

```
x=c(10.2,0.5,5.2,6.1,3.1,6.7,8.9,7.2,8.9,5.4,3.6,9.2,6.1,7.3,2.0,1.3,6.4,8.0,
4.3,4.7,12.4,8.6,13.1,3.2,9.5,7.6,4.0,5.1,8.1,1.1,11.5,3.1,6.8,7.0,8.2,2.0,3.
1,6.5,11.2,12.0,5.1,10.9,11.2,8.5,2.3,3.4,5.2,10.7,4.9,6.2)
n=10
table_s=function(x,n){
  limit<-rep(0,n)
  differ=max(x)-min(x)
  int_differ=differ/n
  limit[1]=min(x)
  for(i in 1:(n+1)){
    limit[i]=limit[1]+(i-1)*int_differ
  }
  limit
  lower_limit=limit[1:n]
  upper_limit=limit[2:(n+1)]
  freq<-rep(0,n)
  freq[1]=sum(x<=upper_limit[1])
  for(i in 2:n){
    freq[i]=sum(x<=upper_limit[i])-sum(freq)
  }
  freq
  class_mark<-rep(0,n)
  for(i in 1:n){
    class_mark[i]=((lower_limit[i]+upper_limit[i])/2)
  }
  class_mark
  total=rep(0,n)
  for(i in 1:n){
    total[i]=sum(class_mark[i]*freq[i])
  }
  total
  table=data.frame(lower_limit,upper_limit,class_mark,freq,total)
  return(table)
}
table_s(x,n)
```

```
   lower_limit upper_limit class_mark freq total
1         0.50        1.76       1.13    3  3.39
2         1.76        3.02       2.39    3  7.17
3         3.02        4.28       3.65    7 25.55
4         4.28        5.54       4.91    8 39.28
5         5.54        6.80       6.17    7 43.19
6         6.80        8.06       7.43    5 37.15
7         8.06        9.32       8.69    7 60.83
8         9.32       10.58       9.95    2 19.90
```

```
9        10.58       11.84      11.21    5 56.05
10       11.84       13.10      12.47    3 37.41
```

```
skewness_contineous=function(l,u,x,f,s){
  mean_data=s/sum(f)
  for(i in 1:n){
    sum_mu_four=sum(((x[i]-mean_data)^4)*f[i])
    sum_mu_three=sum(((x[i]-mean_data)^3)*f[i])
    sum_mu_two=sum(((x[i]-mean_data)^2)*f[i])
  }
  sum_mu_four
  sum_mu_three
  sum_mu_two
  beta_two=(sum_mu_four/sum(f))/(((sum_mu_two)/sum(f))^2)
  beta_one=((sum_mu_three/sum(f))^2)/((sum_mu_two/sum(f))^3)
  skew=(((sqrt(beta_one))*(beta_two+3))/(2*(5*beta_two-6*beta_one-9)))
  if(skew>0){
    print("asymmetrical distribution:positively skewed distribution")
  }else if(skew<0){
    print("asymmetrical distribution:negatively skewed distribution")
  }else{
    print("symmetrical distribution")
  }
  return(skew)
}
skewness_contineous(table_s(x,n)[,"lower_limit"],table_s(x,n)[,"upper_limit"]
,table_s(x,n)[,"class_mark"],table_s(x,n)[,"freq"],table_s(x,n)[,"total"])
```

```
[1] "asymmetrical distribution:negatively skewed distribution"

[1] -0.2832641
```

Kurtosis-

Measure of Kurtosis

Kurtosis enables us to have an idea about the “flatness or pickiness” of the frequency distribution. It is measured by the coefficient beta_2=meu4/(meu2)^2 its derivation is given by gamma_2=beta2-3

If beta_2 = 3 , given distribution is mesokurtic.

If beta_2 > 3 , given distribution is leptokurtic.

If beta_2 < 3 , given distribution is platykurtic.

Method-1

```
moment=function(x,p,r){
  m=sum(x*p)
```

```
  mtr=sum(((x-m)^r)*p)
  z=round(mtr,4)
  return(z)
}
# moment(x,p,r)
x=c(0:10)
f=c(1,8,28,56,70,56,28,8,1,2,4)
p=f/sum(f)
A=cbind(x,f,p)
A

       x  f           p
 [1,]  0  1 0.003816794
 [2,]  1  8 0.030534351
 [3,]  2 28 0.106870229
 [4,]  3 56 0.213740458
 [5,]  4 70 0.267175573
 [6,]  5 56 0.213740458
 [7,]  6 28 0.106870229
 [8,]  7  8 0.030534351
 [9,]  8  1 0.003816794
[10,]  9  2 0.007633588
[11,] 10  4 0.015267176

meu_1=moment(x,p,1)
meu_2=moment(x,p,2)
meu_3=moment(x,p,3)
meu_4=moment(x,p,4)
meu2=meu_2-(meu_1)^2
meu4=meu_4-(4*meu_3*meu_1)+(6*meu_2*(meu_1)^2)-3*(meu_1)^4
beta_2=meu4/(meu2)^2
beta_2

[1] 4.653675

## Since beta_2>3,the given distribution is leptokurtic,that is the frequency
curve is more peaked than the normal curve.
# Excess of kurtosis
gamma_2=beta_2-3
gamma_2

[1] 1.653675
```

Method-2

Kurtosis for Discrete Distribution

```
x<-c(42,74,40,60,82,115,41,61,75,83,63,53,110,76,84,50,67,65,78,77,56,95,68,6
9,104,80,79,79,54,73,59,81,100,66,49,77,90,84,76,42,64,69,70,80,72,50,79,52,1
03,96,51,86,78,94,71)
kurtosis_data=function(x){
```

```
  sorted_data=sort(unique(x))
  freq<-rep(0,length(sorted_data))
  for(i in 1:length(sorted_data)){
    freq[i]=sum(x==sorted_data[i])
  }
  freq
  sum=0
  for(i in 1:length(sorted_data)){
    sum=sorted_data[i]*freq[i]+sum
  }
  mean_data=sum/sum(freq)
  sum_mu_four=0
  sum_mu_two=0
  for(i in 1:length(sorted_data)){
    sum_mu_four=(((sorted_data[i]-mean_data)^4)*freq[i])+sum_mu_four
    sum_mu_two=(((sorted_data[i]-mean_data)^2)*freq[i])+sum_mu_two
  }
  sum_mu_four
  sum_mu_two
  beta_two=(sum_mu_four/sum(freq))/(((sum_mu_two)/sum(freq))^2)
  if(beta_two>3){
    print("leptokurtic curve")
  }else if(beta_two<3){
    print("platykurtic curve")
  }else{
    print("mesokurtic curve")
  }
  return(beta_two)
}
kurtosis_data(x)

[1] "platykurtic curve"

[1] 2.688814
```

Kurtosis for Continuous Distribution

```
x=c(10.2,0.5,5.2,6.1,3.1,6.7,8.9,7.2,8.9,5.4,3.6,9.2,6.1,7.3,2.0,1.3,6.4,8.0,
4.3,4.7,12.4,8.6,13.1,3.2,9.5,7.6,4.0,5.1,8.1,1.1,11.5,3.1,6.8,7.0,8.2,2.0,3.
1,6.5,11.2,12.0,5.1,10.9,11.2,8.5,2.3,3.4,5.2,10.7,4.9,6.2)
n=10
table_k=function(x,n){
  limit<-rep(0,n)
  differ=max(x)-min(x)
  int_differ=differ/n
  limit[1]=min(x)
  for(i in 1:(n+1)){
    limit[i]=limit[1]+(i-1)*int_differ
  }
  limit
  lower_limit=limit[1:n]
```

```
  upper_limit=limit[2:(n+1)]
  freq<-rep(0,n)
  freq[1]=sum(x<=upper_limit[1])
  for(i in 2:n){
    freq[i]=sum(x<=upper_limit[i])-sum(freq)
  }
  freq
  class_mark<-rep(0,n)
  for(i in 1:n){
    class_mark[i]=((lower_limit[i]+upper_limit[i])/2)
  }
  class_mark
  total=rep(0,n)
  for(i in 1:n){
    total[i]=sum(class_mark[i]*freq[i])
  }
  total
  table=data.frame(lower_limit,upper_limit,class_mark,freq,total)
  return(table)
}
table_k(x,n)
```

```
   lower_limit upper_limit class_mark freq total
1         0.50        1.76       1.13    3  3.39
2         1.76        3.02       2.39    3  7.17
3         3.02        4.28       3.65    7 25.55
4         4.28        5.54       4.91    8 39.28
5         5.54        6.80       6.17    7 43.19
6         6.80        8.06       7.43    5 37.15
7         8.06        9.32       8.69    7 60.83
8         9.32       10.58       9.95    2 19.90
9        10.58       11.84      11.21    5 56.05
10       11.84       13.10      12.47    3 37.41
```

```
kurtosis_contineous=function(l,u,x,f,s){
  mean_data=s/sum(f)
  for(i in 1:n){
    sum_mu_four=sum(((x[i]-mean_data)^4)*f[i])
    sum_mu_two=sum(((x[i]-mean_data)^2)*f[i])
  }
  sum_mu_four
  sum_mu_two
  beta_two=(sum_mu_four/sum(f))/(((sum_mu_two)/sum(f))^2)
  if(beta_two>3){
    print("leptokurtic curve")
  }else if(beta_two<3){
    print("platykurtic curve")
  }else{
    print("mesokurtic curve")
  }
```

```
  return(beta_two)
}
kurtosis_contineous(table_k(x,n)[,"lower_limit"],table_k(x,n)[,"upper_limit"]
,table_k(x,n)[,"class_mark"],table_k(x,n)[,"freq"],table_k(x,n)[,"total"])

[1] "platykurtic curve"

[1] 1.672917
```

Conclusion:

Our topic of interest which has been covered through our project upholds that statistical measures can give reliability in data analysis when programmed in R and specially becomes reliable and interesting when own functions are created and compiled through it.

Acknowledgement:

We would like to express our special thanks of gratitude to our Prof. Biman Chakraborty who have given us the golden opportunity to do this wonderful project on descriptive statistics and its measures. Who also helped us in completing our project. We came to know about so many new things and we are really grateful to him.

Reference:

Gupta S.C, Kapoor V.K,2020, FUNDAMENTALS OF MATHEMATICAL STATISTICS, DESCRIPTIVE MEASURES. Educational Publishers.23, Daryaganj, New Delhi-110002: SULTAN CHAND & SONS.p.2.1-2.74.

Thanking You