Helping Students Connect with Data: Using R in Learning Introductory Statistical Concepts

R works with data structures such as vectors (one dimensional array) and data frames (two dimensional arrays). When R is started, we will see a window that is called the R console. This is where we type our commands and see the text results. Graphics appear in a separate window. The > is called the prompt, where R commands are written. The results of an R command can be assigned to a variable using <- or =. In this paper, we will use =. In R, a vector is a sequence of data values of the same type. The function, c, is used to create vectors from scalars. The following statements create a vector and display it.

```
> x <- c(2, 4, 6, 8, 10)
> x
[1]  2  4  6  8 10
```

Once we have a vector of numbers, we can apply built-in functions to get useful statistical summaries and visual displays.

We will use a csv (comma-separated values) file named ***HealthData.csv*** for introducing descriptive statistics. This file has the health data information (gender, age, height, weight, waist and pulse rate) of 80 individuals. This file has been saved in *Documents* folder of your computer.

To read the data into a data frame named *ourdata* from the csv file, type

```
> ourdata = read.csv(file.choose(), header = TRUE)
```

To open a text file, replace *read.csv* by *read.table*.

To access the data in *ourdata* data frame, type

```
> attach(ourdata)
```

To visualize the data in R window, type

```
> ourdata
```

```
      Gender Age  Height  Weight  Waist  Pulse
1       M     58   70.8   169.1   90.6    68
2       M     22   66.2   144.2   78.1    64
3       M     32   71.7   179.3   96.5    88
4       M     31   68.7   175.8   87.7    72
5       M     28   67.6   152.6   87.1    64
6       M     46   69.2   166.8   92.4    72
```

7	M	41	66.5	135.0	78.8	60
8	M	56	67.2	201.5	103.3	88
9	M	20	68.3	175.2	89.1	76
10	M	54	65.6	139.0	82.5	60
11	M	17	63.0	156.3	86.7	96
12	M	73	68.3	186.6	103.3	72

Descriptive Statistics

To compute the summary statistics of a variable (say Height in above data), use *summary* command.

```
> summary(Height)
  Min.   1st Qu.  Median    Mean   3rd Qu.   Max.
  57.00   63.08   66.15      65.77    68.30     76.20
```

Use following commands to find other useful statistics of the data:

```
> sd(Height)               for  standard deviation
[1] 3.859957
```

```
> var(Height)              for variance
[1] 14.89927
```

```
> quantile(Height)         for quartiles
   0%     25%      50%      75%      100%
57.000  63.075   66.150  68.300  76.200
```

```
> quantile(Height,0.90)           for 90th percentile
  90%
70.82
```

```
> sum(Height)              for sum
[1] 5261.2
```

```
> max(Height)              for maximum value
[1] 76.2
```

```
> min(Height)          for minimum value
[1] 57
```

```
> length(Height)       for number of values
[1] 80
```

```
> fivenum(Height)      for five number summary
[1] 57.00   63.05   66.15   68.30   76.20
```

The *table* command display the frequency table. The following is the frequency table for variable Age.

```
> table(Age)
Age
12 16 17 18 19 20 22 23 24 25 26 27 28 29 31 32 33 34 36 37 40 41 42 44 45 46
 1  1  3  3   2  4  2  4   1  3  2   2  3   4  3  5   2  2  2   3  3  4   1  1  2  1
47 48 52 53 54 55 56 57 58 59 73
 1  1   3  2  1  2   2  1  1  1  1
```

Visual Displays

The *hist* command creates a histogram.

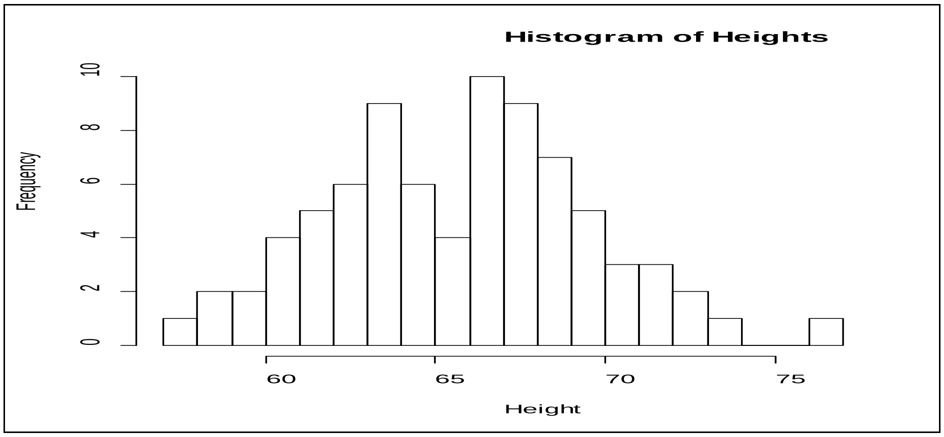
```
> hist(Height)
```

To specify the number of bars, use the option *breaks*.

```
> hist(Height, breaks = 15)
```

To add a title for the plot, use the *main* option.

```
> hist(Height, breaks = 15, main = "Histogram of Heights")
```



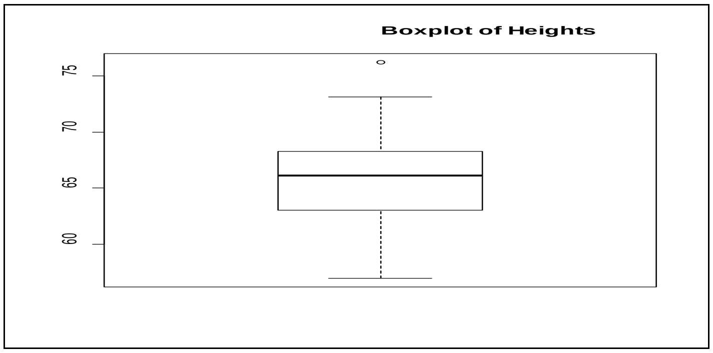
For stem and leaf plot:

```
> stem(Height)
```

```
  The decimal point is at the |
  56 | 0
  58 | 2668
  60 | 256733489
  62 | 336790123444567
  64 | 1337801466
  66 | 123345778026669
  68 | 000033335772247
  70 | 0380179
  72 | 401
  74 |
  76 | 2
```

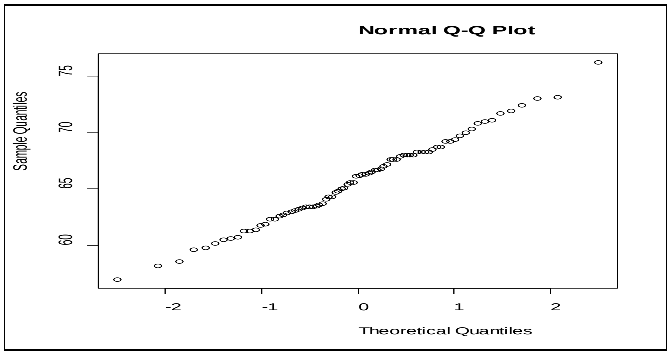
To create a boxplot:

```
> boxplot(Height, main = "Boxplot of Heights")
```

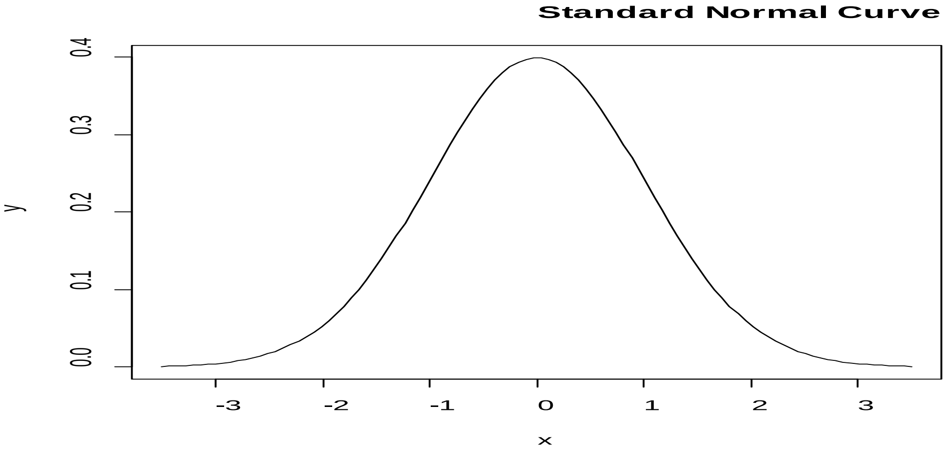


To create a normal probability plot:

```
> qqnorm(Height)
```



Normal Distribution



dnorm function

dnorm(x, μ, σ) function gives the height of the density function (pdf) at a value of x of the normal distribution with mean *μ* and standard deviation *σ*.

To calculate the height of pdf at x = 15 of the normal distribution with μ = 20 and σ = 4:

```
> dnorm(15,20,4)
[1]  0.04566227
```

For standard normal distribution, you do not have to specify μ and σ. You could use either *dnorm(x)* or *dnorm (x, 0, 1)*.

```
> dnorm(0.5)
[1] 0.3520653
```

```
> dnorm(0.5,0,1)
[1] 0.3520653
```

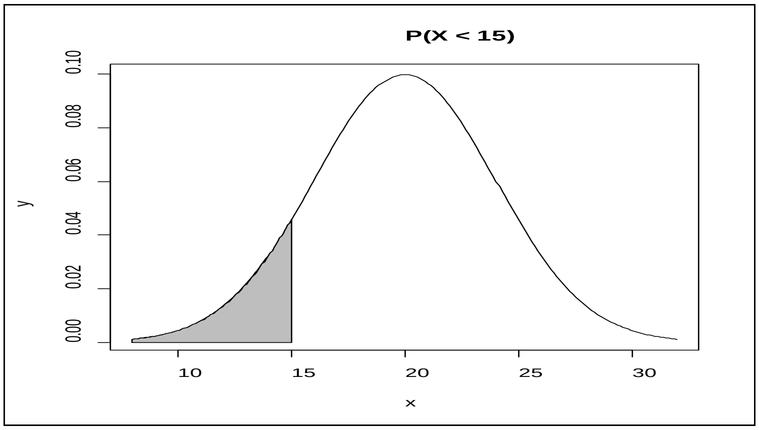
pnorm function

pnorm(x, μ, σ) function gives the area under the normal curve (cdf) with mean μ and standard deviation σ to the left of x. This is the probability that P(X ≤ x).

Consider the normal distribution with μ = 20 and σ = 4. To compute P(X ≤ 15):

```
> pnorm(15,20,4)
[1] 0.1056498
```

This probability (area under the curve) is shown in the following figure:



To compute the area above 15 , P (X > 15):

```
> 1 - pnorm(15,20,4)
[1] 0.8943502
```

To compute the area between 15 and 25, P(15 < X < 25):

```
> pnorm(25,20,4) - pnorm(15,20,4)
[1] 0.7887005
```

qnorm function

The *qnorm(p, μ, σ)* function gives the value at which the cdf (P(X ≤ x)) of the normal distribution with mean μ and standard deviation σ is *p*. In other words, it computes the *p*th quantile of the normal distribution.

To find x such that P(X ≤ x) = 0.90 in the normal distribution with μ = 20 and σ = 4:

```
> qnorm(0.90, 20, 4)
[1] 25.12621
```

rnorm function

rnorm(n, μ, σ) function generates *n* random numbers from the normal distribution with mean μ and standard deviation σ.

To generate 10 random numbers from the normal distribution with μ = 20 and σ = 4:

```
> rnorm(10,20,4)
 [1] 19.16433  21.84712  20.66090  26.43889  18.02179  13.92341  19.03123  24.49400
 [9] 21.65163  11.81368
```

Similarly, one can generate random samples from other probability distributions.

Binomial distribution

rbinom(m,n,p) generates *m* random numbers from the binomial distribution with *n* and *p* as parameters.

To generate a sample of **10** random numbers from the binomial distribution with *n* = 15, and *p* = 0.2:

```
> rbinom(10,15,0.2)
 [1] 2 6 3 4 5 2 1 2 1 2
```

Sampling Distributions and Central Limit Theorem

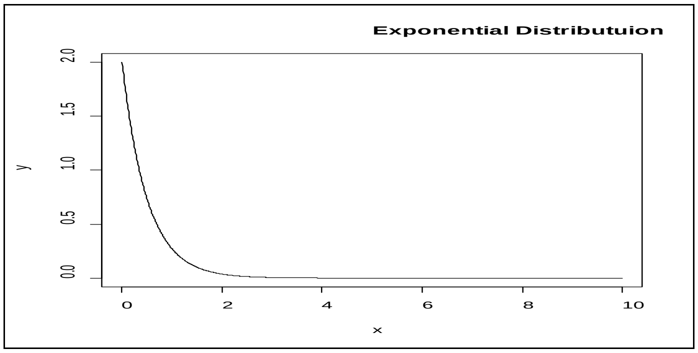
The sampling distribution of the mean is the probability distribution of the sample mean based on all possible simple random samples of the same size from the same population.

We can use simulations to understand and visualize the following properties of the sampling distributions:

- The mean of all sample means is equal to the population mean (μ).
- The standard deviation of the sample means (known as the standard error) is equal to the population standard deviation divided by square root of the sample size($\sigma/\sqrt{n}$).
- Sample means are more normal than individual observations.

The **central limit theorem** explains the shape of the sampling distribution. This theorem tells that for a population of any distribution, the distribution of the sample mean approaches a normal distribution as the sample size increases. The larger the sample size, the better the approximation.

To demonstrate this, we generate random samples from a skewed distribution. We use the exponential distribution (with parameter $\lambda = 2$) and show that the sampling distribution of sample mean approaches a normal distribution as the sample size increases.



We use *rexp(n, λ)* to generate a random sample of *n* values from exponential distribution with parameter λ.

We consider sample sizes of 10, 25, and 50. In each case we generate 10,000 random samples and compute the sample mean and observe the distributional shape using histograms and normal probability plots. We use a *for* loop to generate 10,000 samples and compute sample means. The following R code does the simulation of the process described above.

```
> means = c()
> for(i in 1: 10000)
{
+   y =  rexp(10,2)
+   means[i] = mean(y)
 }
```

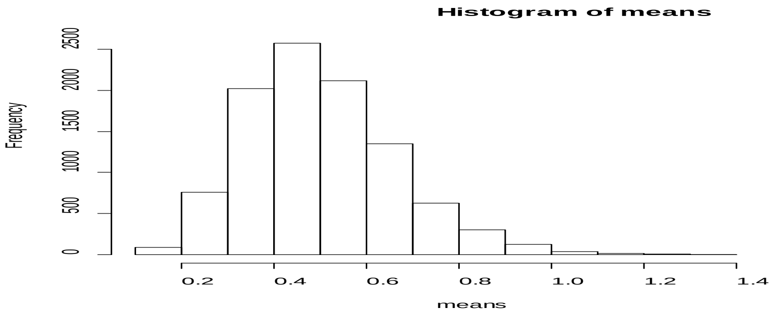
The mean and standard deviation of the sample means generated above are computed as:

```
> mean(means)
[1] 0.4963717
> sd(means)
[1] 0.1558105
```

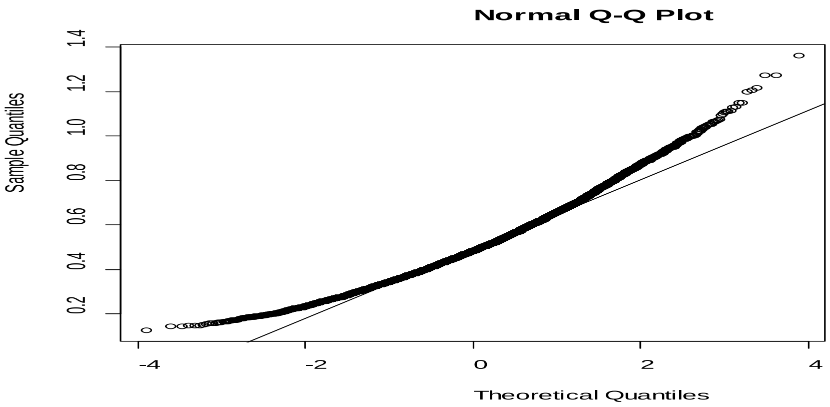
We can notice that the approximate mean and the standard deviation are close to the theoretical mean and standard deviation of the sampling distribution.
Mean $\mu = 1/\lambda = 1/2 = 0.5$
Standard deviation = $\sigma/\sqrt{n} = (1/2)/\sqrt{10} = 0.15811$.

To visualize the shape of the sampling distribution, we create histograms and normal probability plots.

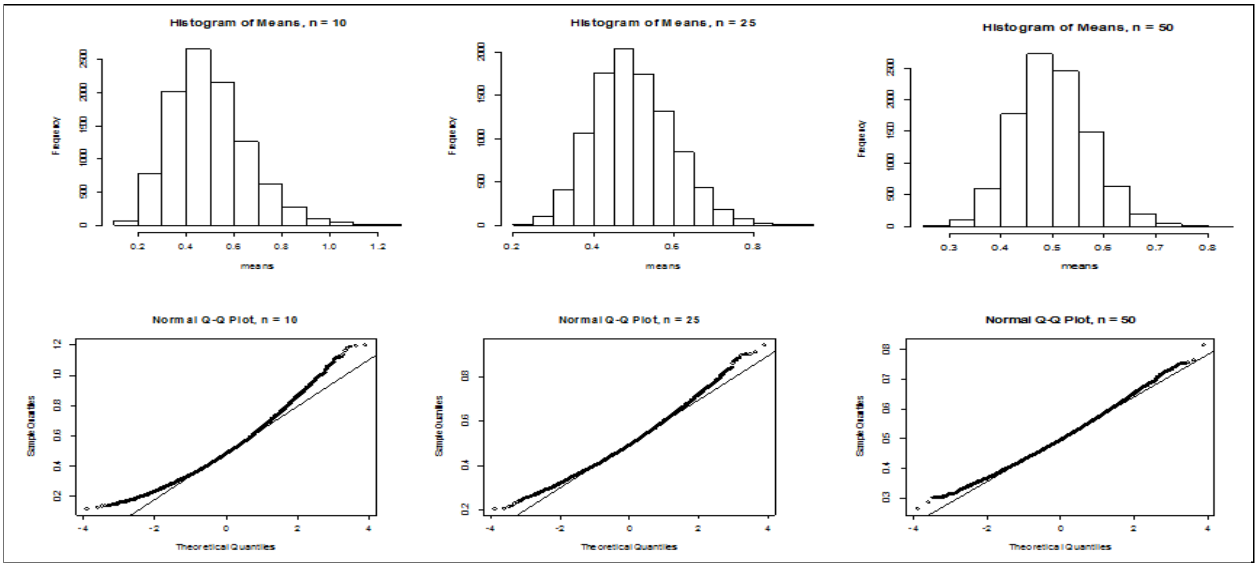
```
> hist(means)
```



```
> qqnorm(means)
> qqline(means)
```



The figure below shows histograms and normal probability plots of sample means for n = 10, 25 and 50.



Notice that as sample size increases, the sampling distribution becomes more normal.

Hypothesis testing

Hypothesis testing is a key topic in statistical inferences.

One sample t-test

We use the following example to demonstrate the *one sample t test* for mean.
Consider the *HealthData.csv* file opened earlier. We use the *Age* data to test whether the mean age is equal to 40 years.

The null and alternative hypotheses are: H_0: $\mu = 40$ years and H_1: $\mu \neq 40$ years.
In R, *t.test* command performs the t-test and produces the test statistic and the *p*-value.

```
> t.test(Age, mu = 40, alternative = 'two.sided')

        One Sample t-test

data:  Age
t = -3.8355, df = 79, p-value = 0.0002507
alternative hypothesis: true mean is not equal to 40
95 percent confidence interval:
 31.41791   37.28209
sample estimates:
mean of x
   34.35
```

To perform a left tailed or right tailed test, *'two.sided'* should be replaced with '*less*' or '*greater*'.

Since the *p*-value (0.0002507) is less than the significance level α (say 0.05), we have sufficient evidence to reject the null hypothesis and conclude that the mean age is different from 40 years.

To compute confidence intervals for different confidence levels (other than 95%), use *conf.level* option:

```
> t.test(Age, mu = 40, alternative = 'two.sided', conf.level = 0.99)

        One Sample t-test
```

```
data:  Age
t = -3.8355, df = 79, p-value = 0.0002507
alternative hypothesis: true mean is not equal to 40
99 percent confidence interval:
 30.4618  38.2382
sample estimates:
mean of x
   34.35
```

Two sample t-test

Crime Rate: A random sample of n_1 = 10 regions in New England gave the following violent crime rates (per million population)

x_1 values: 3.5 3.7 4.0 3.9 3.3 4.1 1.8 4.8 2.9 3.1

Another random sample of n_2 = 12 regions in Rocky Mountain areas gave the following violent crime rates (per million population)

x_2 values: 3.7 4.3 4.5 5.3 3.3 4.8 3.5 2.4 3.1 3.5 5.2
2.8

Do the data indicate that the average violent crime rate in New England region is same as that in Rocky Mountain area? Use α = 0.01.

Before we perform the test, it is necessary to perform F-test for equality of variance.

```
> x1 = c(3.5, 3.7, 4.0, 3.9, 3.3, 4.1, 1.8, 4.8, 2.9, 3.1)
> x2 = c(3.7, 4.3, 4.5, 5.3, 3.3, 4.8, 3.5, 2.4, 3.1, 3.5, 5.2, 2.8)

> var.test(x1,x2)

        F test to compare two variances

data:  x1 and x2
F = 0.74295, num df = 9, denom df = 11, p-value = 0.6662
alternative hypothesis: true ratio of variances is not equal to 1
95 percent confidence interval:
 0.207071 2.906474
```

```
sample estimates:
ratio of variances
        0.7429496
```

We obtained *p*-value greater than 0.05, then we can assume that the two variances are equal.

Then we perform the *t-test* for equality of means assuming equal variance.

```
> t.test(x1, x2, alternative = 'two.sided',  var.equal = TRUE, paired = FALSE)

        Two Sample t-test
data:  x1 and x2
t = -0.93911, df = 20, p-value = 0.3589
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 -1.1489034   0.4355701
sample estimates:
mean of x   mean of y
 3.510000   3.866667
```

We obtained *p*-value greater than 0.05, then we can conclude that the averages crime rates of two regions are the same.