**Mini Project 1**

**Name :**

**Dhwaniben Kaneria (drk170130)**

**Utkarsh Gandhi (usg170030)**

**Contribution of team members:**

Dhwani:

* Wrote equation and calculation in word
* Learned R Coding
* Tried different R codes
* Wrote narration for codes - Section 1
* Derived conclusions from calculations
* Wrote explanation for Section 1

Utkarsh:

* Explained the simulation
* Learned R coding
* Did debugging for R codes
* Wrote R code - Section 2
* Derived conclusions from graphs
* Wrote explanation for Section 2

**Ans 1.**

**(a)**

P(1)=½

P(2)=⅛

P(3)=⅛

P(4)=¼

E(x)=

=1\*½ +2\*⅛+3\*⅛+4\*¼

=2.125

Var(x)=P(x)

=

=1.609

P(X<=2)=P(x) = P(1) + P(2)

= ½ +⅛

=0.625

**(b)**

1. Here, X is an arbitrary discrete random variable. We can simulate a draw from uniform distribution and divide the interval [0, 1] into subintervals [0,0.5],[0.5,0.625],[0.625,0.75],[0.75,1].
2. Now we could simulate U ~ Uniform(0,1). We can find X by determining the region where the generated value of U belongs. In R, we have the function runif() to generate a random value from uniform distribution.
3. Subinterval Ai will have length 0.25; there may be a finite or infinite number of them, according to possible value of X.

**(c)**

Monte Carlo Simulation,

For 1000 draws,

# repeat draw 5 times

for (i in 1:5) {

#sample is used to take elements of x which is specified by vector (1,2,3,4) with #replacement, second argument is size which is 1 as we need to choose only one item,

# third arg is for replacement which is true and last arg is to mentioned probability vector #which is (0.5,0.125,0.125,0.25).

#To make 1000 draws, replicate function is used.

a=replicate(1000, sample(c(1, 2, 3,4), 1, TRUE, c(0.50, 0.125, 0.125,0.25)))

b=replicate(1000, sample(c(1, 2, 3,4), 1, TRUE, c(0.50, 0.125, 0.125,0.25))<=2)

e= mean(a)

v=var(a)

cdf=mean(b)

print (c(e,v,cdf))

}

|  |  |  |
| --- | --- | --- |
| **E(x)** | **Var(x)** | **P(X<=2)** |
| 2.089000 | 1.552632 | 0.627000 |
| 2.214000 | 1.653858 | 0.603000 |
| 2.180000 | 1.635235 | 0.651000 |
| 2.119000 | 1.600439 | 0.650000 |
| 2.108000 | 1.591928 | 0.633000 |

**(d)**

For 5000 draws,

for (i in 1:5) {

a=replicate(5000, sample(c(1, 2, 3,4), 1, TRUE, c(0.50, 0.125, 0.125,0.25)))

b=replicate(5000, sample(c(1, 2, 3,4), 1, TRUE, c(0.50, 0.125, 0.125,0.25))<=2)

e= mean(a)

v=var(a)

cdf=mean(b)

print (c(e,v,cdf))

}

|  |  |  |
| --- | --- | --- |
| **E(x)** | **Var(x)** | **P(X<=2)** |
| 2.145600 | 1.635128 | 0.630800 |
| 2.151000 | 1.635726 | 0.621800 |
| 2.127200 | 1.620544 | 0.621800 |
| 2.15760 | 1.64109 | 0.63800 |
| 2.151200 | 1.621863 | 0.617800 |

For 10000 draws,

for (i in 1:5) {

a=replicate(10000, sample(c(1, 2, 3,4), 1, TRUE, c(0.50, 0.125, 0.125,0.25)))

b=replicate(10000, sample(c(1, 2, 3,4), 1, TRUE, c(0.50, 0.125, 0.125,0.25))<=2)

e= mean(a)

v=var(a)

cdf=mean(b)

print (c(e,v,cdf))

}

|  |  |  |
| --- | --- | --- |
| **E(x)** | **Var(x)** | **P(X<=2)** |
| 2.124800 | 1.602985 | 0.622200 |
| 2.15110 | 1.61603 | 0.62100 |
| 2.129500 | 1.621692 | 0.622500 |
| 2.119100 | 1.600075 | 0.617000 |
| 2.094000 | 1.580122 | 0.622100 |

**(e)**

From **(a), (c), (d)** we have observed that the mean, variance and P(X<=2) are approximately equal as larger number of draws result. As the number of draws increases the mean, variance and P(X<=2) becomes more close to calculated result.

Consider case of Variance,

In(a) we got the variance as 1.609 when calculated analytically. In (c ) when number of draws is 1000, we can see that we got Var(X) values like 1.55, 1.65, 1.59. But in (d) when number of draws=5000, we got Var(X) as 1.63, 1.64, 1.62. Which is much closer to the calculated value. Also in (d) when the number of draws=10000, the mean value are which are much more close to calculated variance like 1.602, 1.616, 1.621.

So as the number of draws increase, the variation from calculated value decreases, as per Law of Large number theorem.

**Ans 2.**

1. According to Central Limit theorem,when n is large, X approximately follows a normal distribution. X1, X2,....Xn represent random sample size n from the Bernoulli population represented by X.

E[ **p̂** ] = p ; p = probability of getting success

Var[ **p̂** ] = p(1-p)/n [holds for all n ≥ 1]

When n is large, the approximate distribution of **p̂** will be

**p̂ ~** N[E[ **p̂** ] , var[ **p̂** ] ]

If X has normal distribution then the sample will also have normal distribution

**(b)**

n=10, 30, 50, 100

p= 0.10, 0.25, 0.50, 0.75, 0.90

Let take n=30 and p=0.10 and the plot the Q-Q plot for 500 values.

**Code in R:**

a=replicate(500,mean(rbinom(30,1,0.10)))

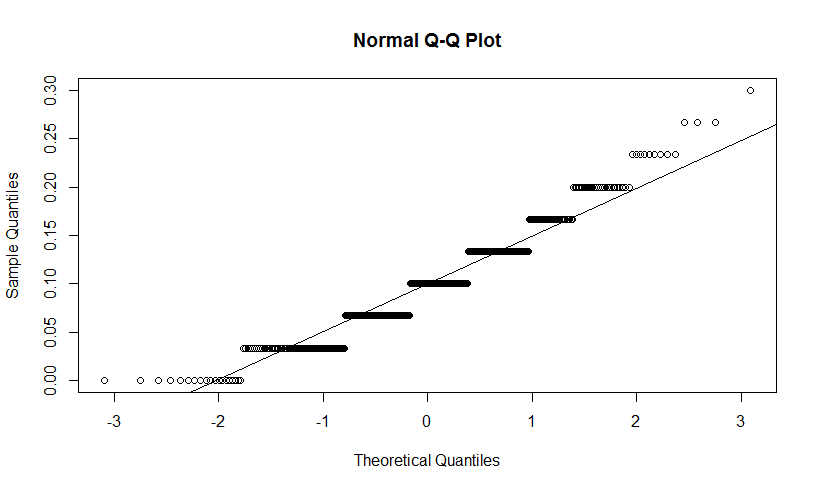
qqnorm(a)

qqline(a)

Narration of code:

1. To get value for binomial distribution, rbinom function is used in which first argument is number of observation, second argument is number of trials and third argument is probability of success on each trial.
2. After that mean of this binomial values, 500 times is done by replicate function and to q-q plot qqnorm function is used and to make line for that graph qqline function is used which draws line in the graph.

* From the below Normal Q-Q plot we can conclude that the distribution doesn’t look normal because many points doesn’t lie on the line. So for n=30 and p=0.10, the distribution is not normal.



**(c)**

n=10, 30, 50, 100

p= 0.10, 0.25, 0.50, 0.75, 0.90

**R Code:**

#this code will give 20 different Q-Q plots for different combinations of n and p

n<-c(10,30,50,100) # taking all values of n

p<-c(0.10,0.25,0.5,0.75,0.90) # taking all values of p

for (i in n)

{

for (j in p)

{

#using rbinom for getting binomial distribution

a=replicate(500,mean(rbinom(i,1,j)))

qqnorm(a) # plotting Q-Q plot

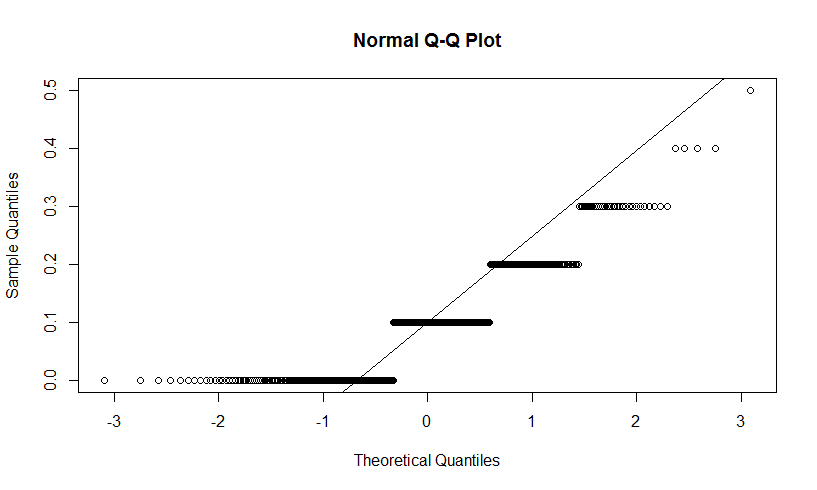
qqline(a) # getting line on the Q-Q plot

}

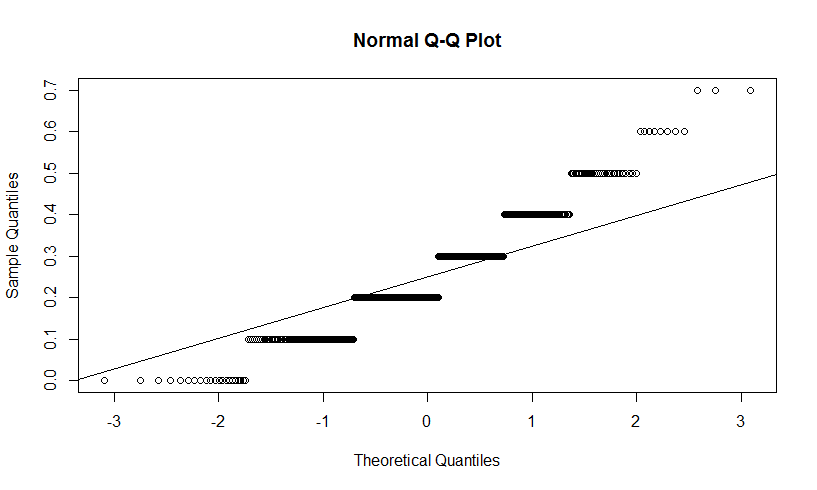
}

Plotting the Q-Q plot for different combinations of n and p.

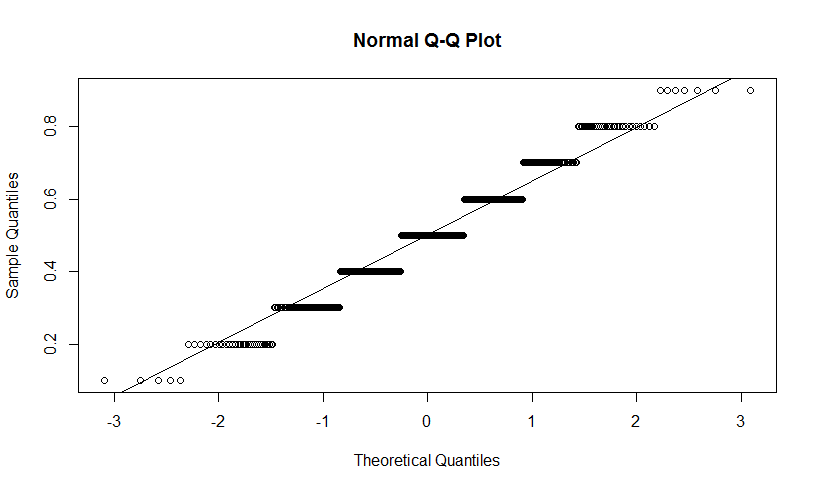
1. **(n,p) = (10, 0.10)**



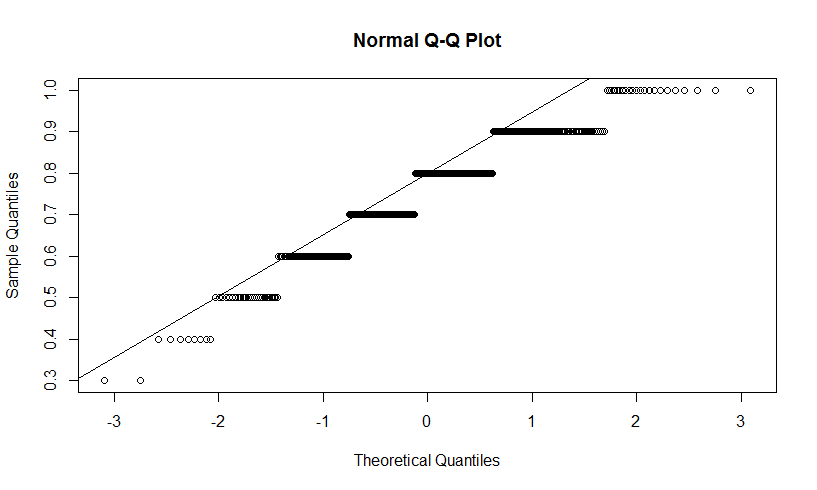
1. **(n,p) = (10, 0.25)**



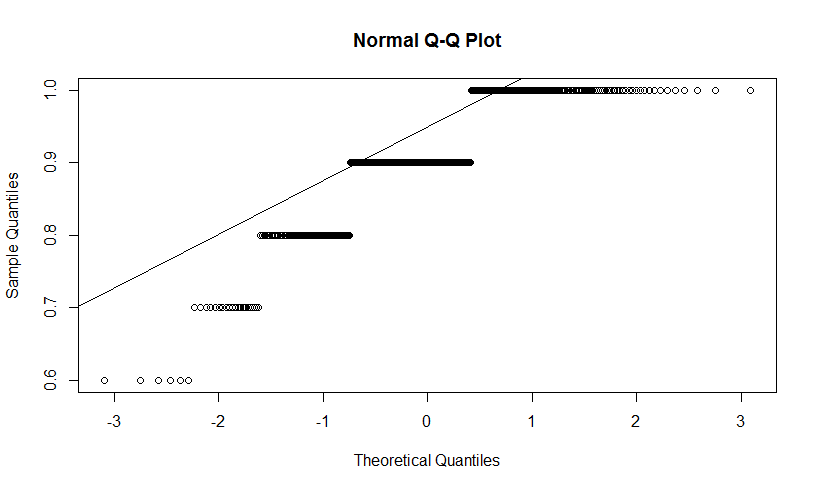
1. **(n,p) = (10, 0.50)**



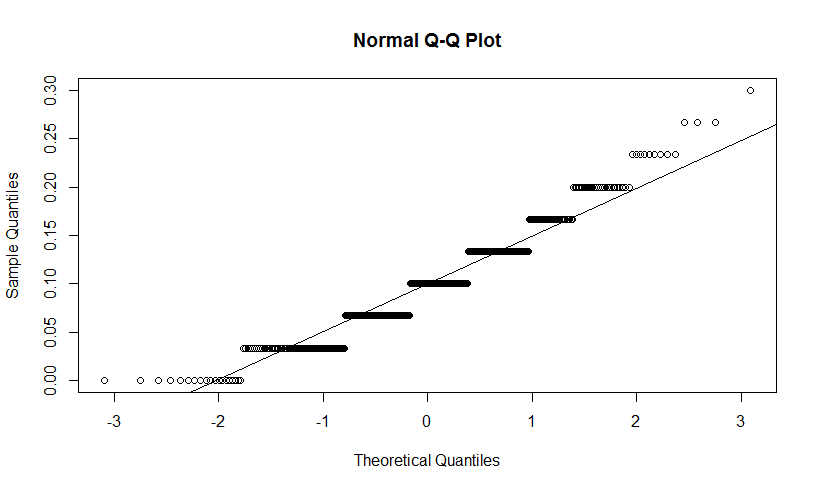
1. **(n,p) = (10, 0.75)**



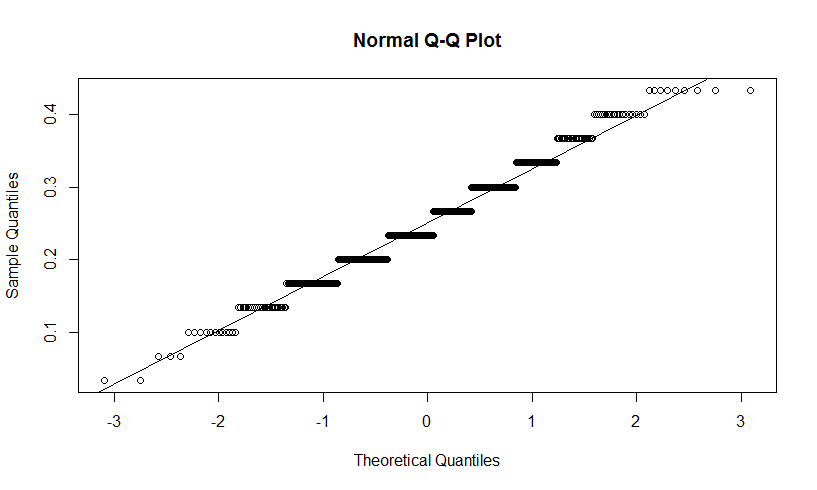
1. **(n,p) = (10, 0.90)**



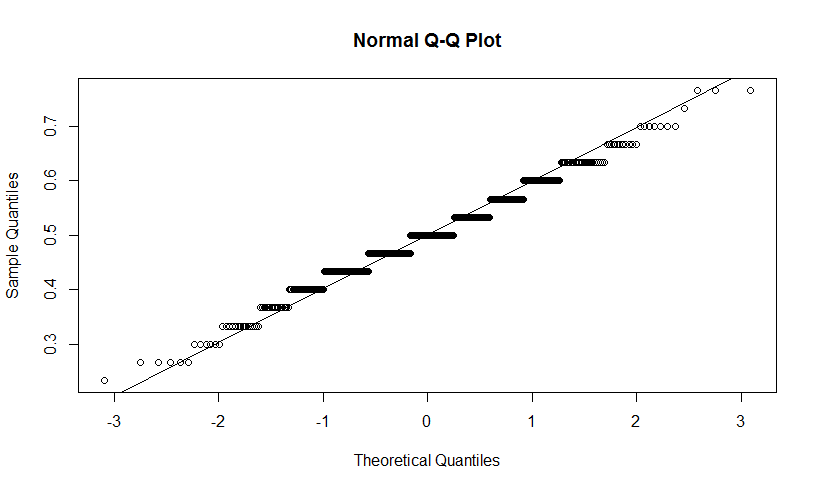
1. **(n,p) = (30, 0.10)**



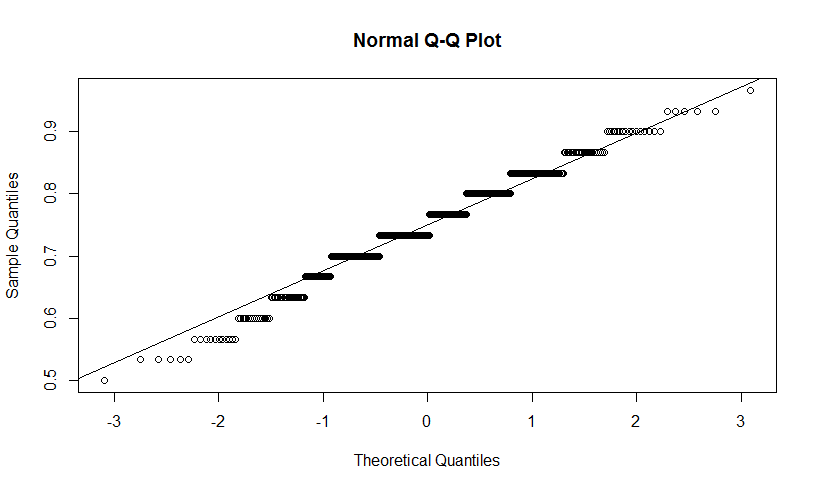
1. **(n,p) = (30, 0.25)**



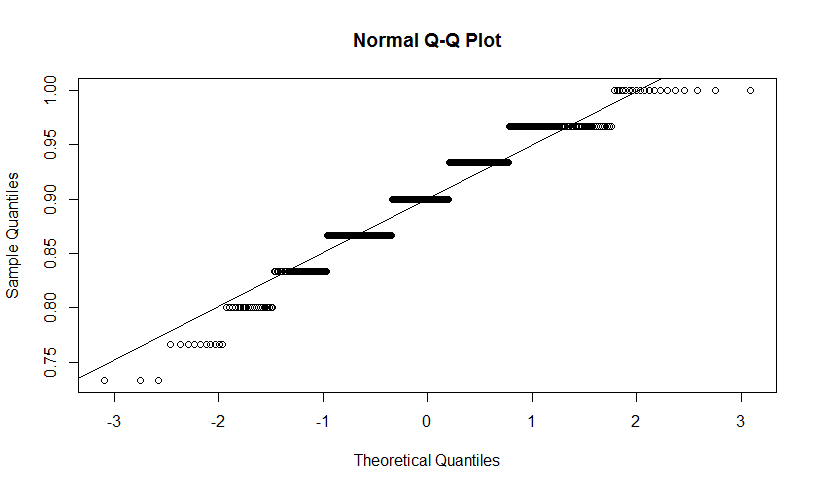
1. **(n,p) = (30, 0.50)**



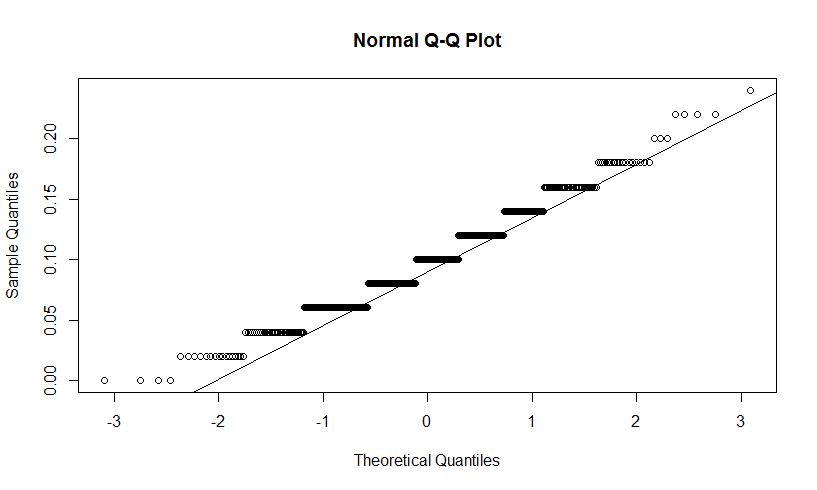
1. **(n,p) = (30, 0.75)**



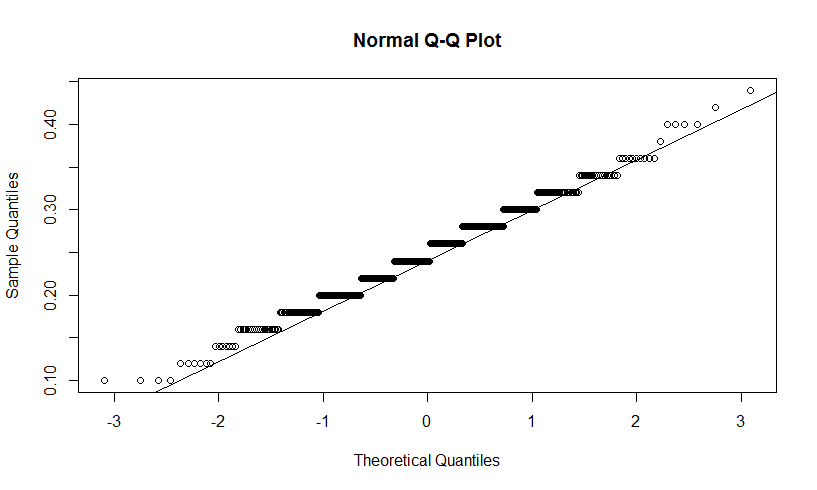
1. **(n,p) = (30, 0.90)**



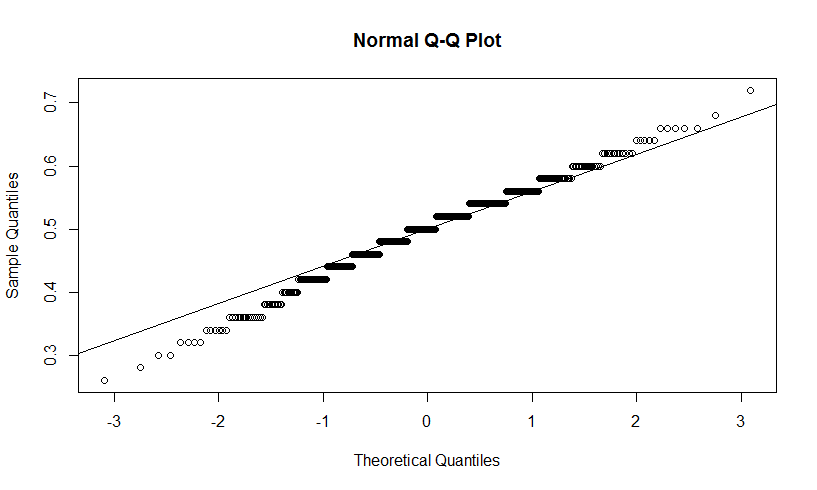
1. **(n,p) = (50, 0.10)**



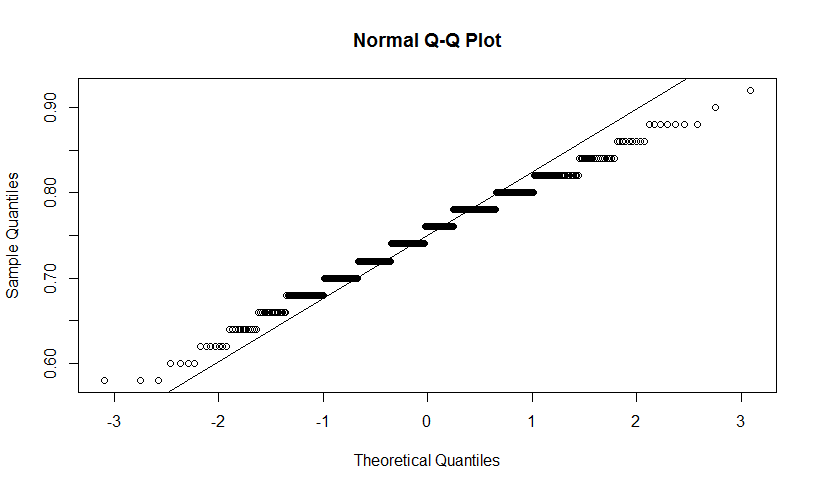
1. **(n,p) = (50, 0.25)**



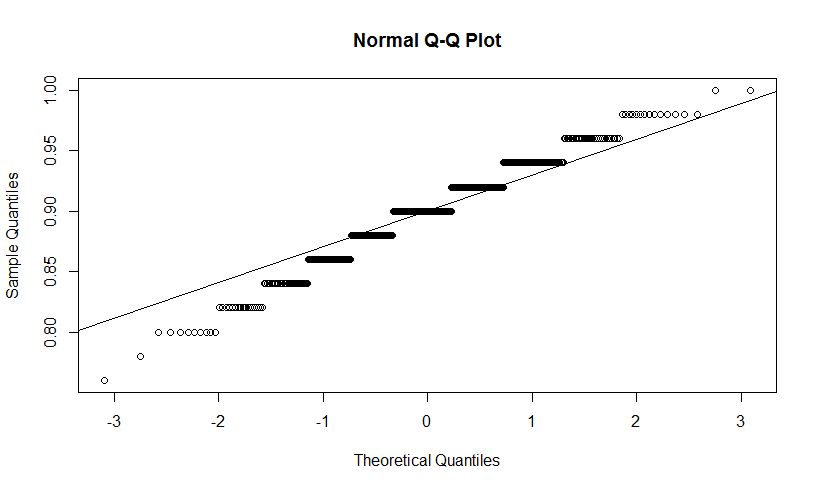
1. **(n,p) = (50, 0.50)**



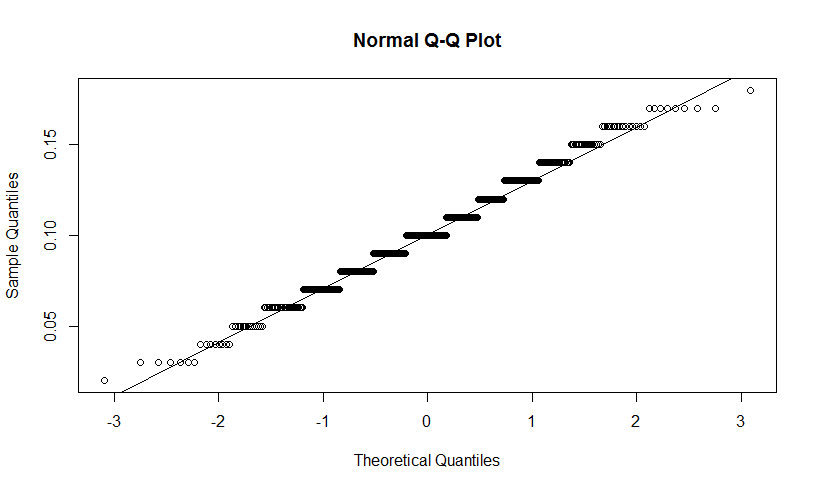
1. **(n,p) = (50, 0.75)**



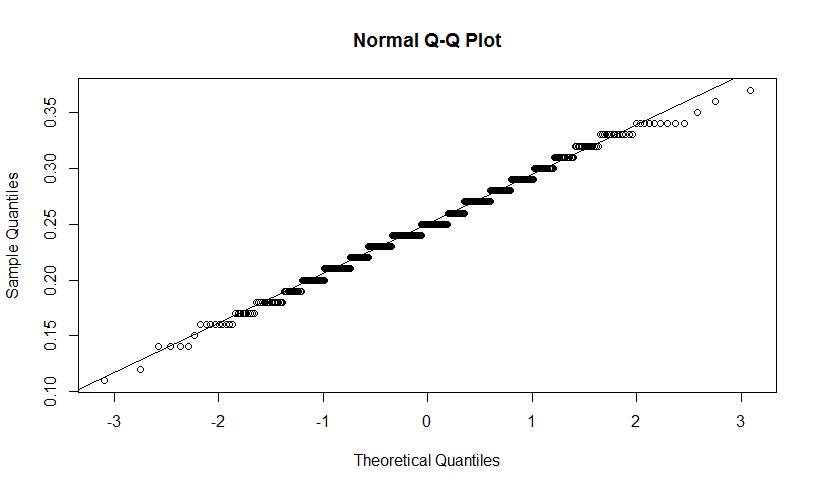
1. **(n,p) = (50, 0.90)**



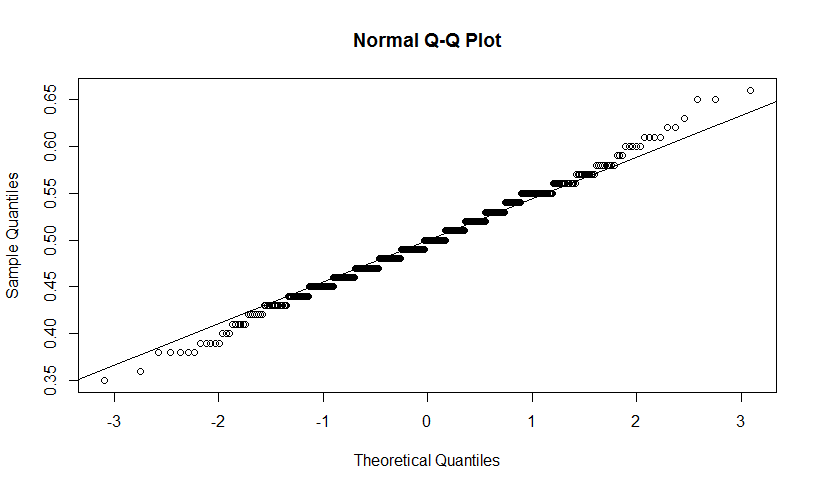
1. **(n,p) = (100, 0.10)**



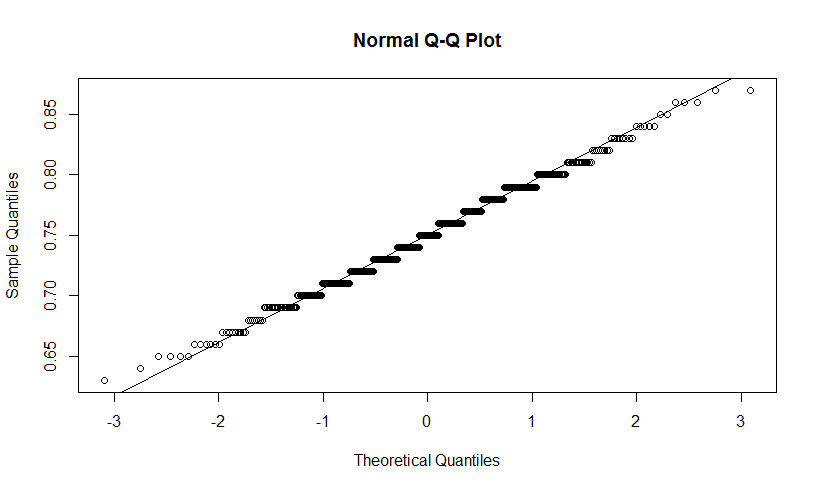
1. **(n,p) = (100, 0.25)**



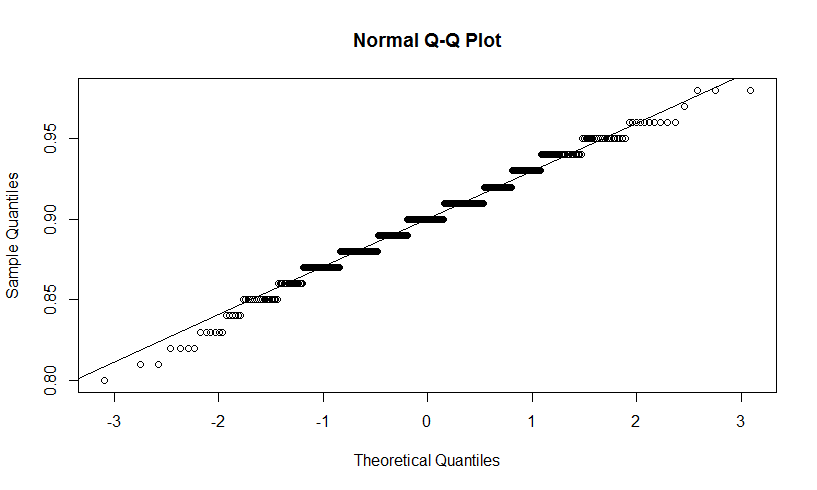
1. **(n,p) = (100, 0.50)**



1. **(n,p) = (100, 0.75)**



1. **(n,p) = (100, 0.90)**



**(d)** The value of **n** should be at least **1000** for the distribution to be approximately following normal distribution for all the given values of **p** unless the value of p is not very low or not very high. In case where p is very low or p is very high then the value of n should be much more larger. So yes it depends on p. Plot for n=1000 and p=0.50

