**Mini Project 1**

**Name :**

**Maitri Dharmendrakumar Shah (mxs172030)**

**Nayana Thomas(nxt170630)**

**Contribution of team members:**

Maitri:

Wrote equation and calculation in word

Drew a graph about simulation and found the way to distribute it

Learned R Coding

Tried different R codes

Wrote narration for codes

Derived conclusions from calculations

Nayana:

Did calculation on paper

Explained the simulation

Learned R coding

Did debugging for R codes

Derived conclusions from graphs

Wrote explanation for section1

**Ans 1.**

**(a)**

P(0) = ½

P(1) = ¼

P(2) = ¼

E(x) = ∑ x P(x)

= 0 \* ½ + 1 \* ¼ + 2 \* ¼

= 0.75

Var (x) = ∑ ( x – E(x))2 P(x)

= ( 0 - ¾ )2 \* ( ½ ) + ( 1 - ¾ )2 \* ( ¼ ) + ( 2 - ¾ )2 \* ( ¼ )

= 0.6875

P ( X > 0 ) = ∑(x=1,2) P(x) = p(1)+p(2)

= 0.25 + 0.25

= 0.5

**(b)**

Section 1:

X is adiscrete random variable with probabilities P(0) = 0.5, P(1) = 0.25 and P(2) = 0.25.

We could simulate a draw from the uniform distribution on the unit interval [0,1].

We can divide the interval [0,1] into subintervals [0,0.5], [0.5,0.75] and [0.75,1]

Now we could simulate U ~ Uniform(0,1). Now we can find X by determining the region where the generated value of U belongs.

Let A0 be the number of Xi that lie in [0,0.5] and A1 be the number of Xi that lie in [0.5,0.75] and A2 be the number of Xi that lie in [0.75,1] for the 1000 draws.

P(0) = A0/1000

P(1) = A1/1000

P(2) = A2/1000

In R, we have the function runif() to generate a random value from a uniform distribution.

**(c)**

Section 1:

For 1000

|  |  |  |
| --- | --- | --- |
| E(x) | Var(x) | P(x>0) |
| 0.791 | 0.6696 | 0.488 |
| 0.764 | 0.6701 | 0.527 |
| 0.782 | 0.6949 | 0.493 |
| 0.735 | 0.6776 | 0.548 |
| 0.755 | 0.6875 | 0.467 |

Section 2:

For E(x):

for (i in 1:5) {

a <- replicate(1000, sample(c(0, 1, 2), 1, TRUE, c(0.5, 0.25, 0.25)))

e <- mean(a)

print (e)

}

Narration of the code:

Here, sample is used to take elements of x which is specified by vector (0,1,2) with replacement. So, first argument of sample is vector of x values, second argument is size which is 1 as we need to choose only one item, third argument is for replacement which is true as we are taking values for multiple times and last argument is to mention probability vector which is (0.5,0.25,0.25).

To make 1000 draws, replicate is function is used. To repeat this draw 5 times for loop is used and at the end, mean is taken by using mean function which shows expected value.

For var(x):

for (i in 1:5) {

a <- replicate(1000, sample(c(0, 1, 2), 1, TRUE, c(0.5, 0.25, 0.25)))

e <- var(a)

print (e)

}

Narration of the code:

Here, sample is used to take elements of x which is specified by vector (0,1,2) with replacement. So, first argument of sample is vector of x values, second argument is size which is 1 as we need to choose only one item, third argument is for replacement which is true as we are taking values for multiple times and last argument is to mention probability vector which is (0.5,0.25,0.25).

To make 1000 draws, replicate is function is used. To repeat this draw 5 times for loop is used and at the end, variance is taken by using var function.

For P(x>0):

for (i in 1:5) {

a <- replicate(1000, sample(c(0, 1, 2), 1, TRUE, c(0.5, 0.25, 0.25)) > 0)

e <- mean(a)

print (e)

}

Narration of the code:

Here, sample is used to take elements of x which is specified by vector (0,1,2) with replacement. So, first argument of sample is vector of x values, second argument is size which is 1 as we need to choose only one item, third argument is for replacement which is true as we are taking values for multiple times and last argument is to mention probability vector which is (0.5,0.25,0.25). As we want P(x>0), at the end we want only >0 values so >0 is mentioned at end, by doing this only 1,2 values will be get by sample function.

To make 1000 draws, replicate is function is used. To repeat this draw 5 times for loop is used and at the end, mean is taken by using mean function which is expected value for P(x>0).

**(d)**

Section 1:

For 5000 draws

|  |  |  |
| --- | --- | --- |
| E(x) | Var(x) | P(x>0) |
| 0.768 | 0.6982 | 0.506 |
| 0.7498 | 0.6888 | 0.4912 |
| 0.76 | 0.6923 | 0.4974 |
| 0.7682 | 0.6797 | 0.495 |
| 0.7504 | 0.6869 | 0.503 |

For 10000 draws

|  |  |  |
| --- | --- | --- |
| E(x) | Var(x) | P(x>0) |
| 0.7575 | 0.6872 | 0.4952 |
| 0.7387 | 0.677 | 0.4925 |
| 0.7465 | 0.6868 | 0.4971 |
| 0.7521 | 0.6983 | 0.5011 |
| 0.7506 | 0.6906 | 0.5018 |

Section 2:

For 5000 draws:

For E(x):

for (i in 1:5) {

a <- replicate(5000, sample(c(0, 1, 2), 1, TRUE, c(0.5, 0.25, 0.25)))

e <- mean(a)

print (e)

}

For var(x):

for (i in 1:5) {

a <- replicate(5000, sample(c(0, 1, 2), 1, TRUE, c(0.5, 0.25, 0.25)))

e <- var(a)

print (e)

}

For P(x>0):

for (i in 1:5) {

a <- replicate(5000, sample(c(0, 1, 2), 1, TRUE, c(0.5, 0.25, 0.25)) > 0)

e <- mean(a)

print (e)

}

Narration of code:

Here, same code as 1000 draws is done only replication digit is changed to 5000.

For 10000 draws:

For E(x):

for (i in 1:5) {

a <- replicate(10000, sample(c(0, 1, 2), 1, TRUE, c(0.5, 0.25, 0.25)))

e <- mean(a)

print (e)

}

For var(x):

for (i in 1:5) {

a <- replicate(10000, sample(c(0, 1, 2), 1, TRUE, c(0.5, 0.25, 0.25)))

e <- var(a)

print (e)

}

For P(x>0):

for (i in 1:5) {

a <- replicate(10000, sample(c(0, 1, 2), 1, TRUE, c(0.5, 0.25, 0.25)) > 0)

e <- mean(a)

print (e)

}

Narration of code:

Here, same code as 1000 draws is done only replication digit is changed to 5000.

**( e )**

From (a), (c) and (d) we can see that the mean, variance and P(X>0) are approximately equal for any number of draws and larger the number of draws, more accurate result is obtained. As the number of draws increases the mean, variance and P(X>0) became more close to the mean, variance and P(X>0) obtained analytically

We can consider the case of mean:

In (a) we got the mean as 0.75 when calculated analytically. In (c) when number of draws is 1000, we can see that we got values like 0.791, 0.782, 0.735 etc. But in (d) when number of draws = 5000, we got E(X) values as 0.7498, 0.76, 0.7682, etc. which is much more closer to the analytically computed value. Also in (d) when the number of draws = 10000, the mean values are 0.7575, 0.7521, 0.7506 which are much more close to the analytical mean.

Same is the case for variance and P(X>0).

So as the number of draws increases, the variation from the analytically computed value decreases.

**Ans 2.**

**(a)**

(a) According to Central Limit Theorem, when n is large, X approximately follows a normal distribution.

X1, X2, … represent random samples of size n from the exponential population represented by X,

The sample sum T =

E(T) = nμ where μ = 1/ λ

Var(T) = nσ 2

Sample average = T/n

Var()= σ 2 /n

If n is large T ~ N[E(T), var(T)]

ie, ~ N[E(T), var(T)]

If X has normal distribution, then the sample also have normal distribution.

Also, the sample mean will be approximately normal with the mean of the population μ and the

variance σ 2 /n.

**(b)**

Section 1:

n = 30

λ = 1

for 500 values

Section 2:

s <- replicate(500, mean(rexp(30, 1)))

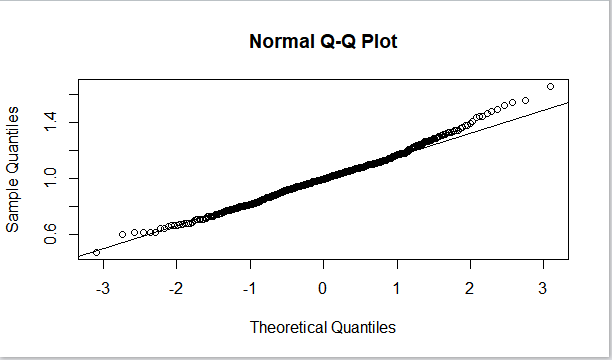
qqnorm(s)

qqline(s)

Narration of code:

To get value for exponential distribution, rexp function is used in which first argument is number of observation and second argument is rate which is λ. This will generate 100 different values.

After that mean of this exponential 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.



**(c)**

Section 1:

n = 30

λ = 10

Section 2:

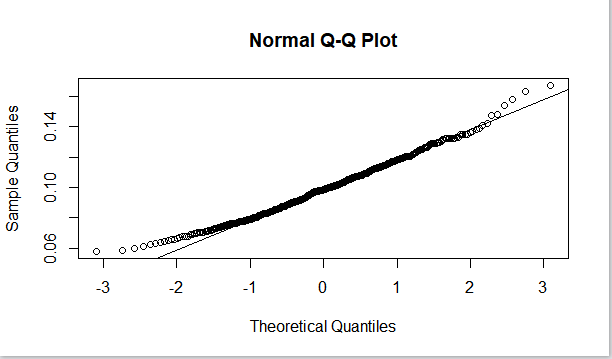
Narration of code:

By taking different values for n and λ, different q-q plot graphs are made.

s <- replicate(500, mean(rexp(30, 10)))

qqnorm(s)

qqline(s)



Section 1:

n = 30

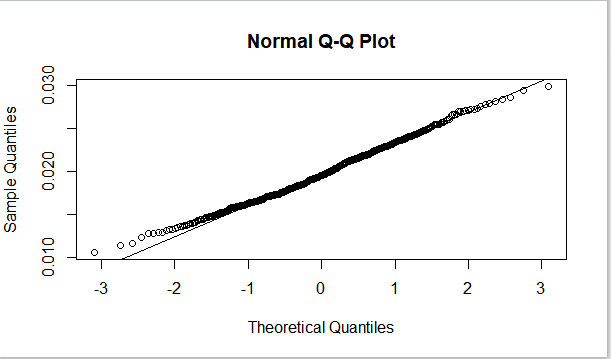
λ = 50

Section 2:

s <- replicate(500, mean(rexp(30, 50)))

qqnorm(s)

qqline(s)



Section 1:

n = 50

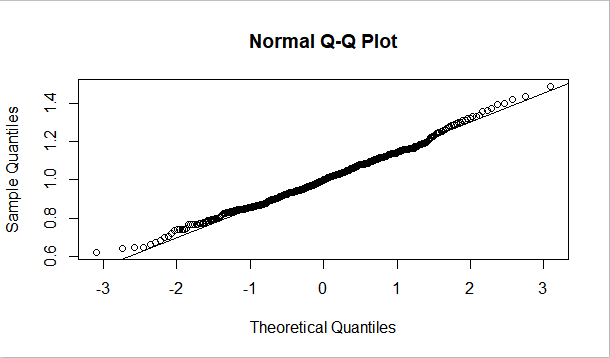
λ = 1

Section 2:

s <- replicate(500, mean(rexp(50, 1)))

qqnorm(s)

qqline(s)



Section 1:

n = 50

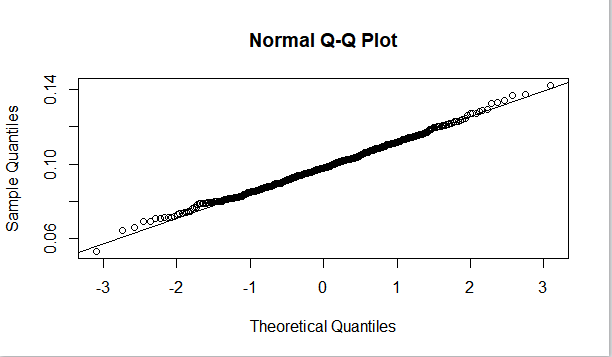
λ = 10

Section 2:

s <- replicate(500, mean(rexp(50, 10)))

qqnorm(s)

qqline(s)



Section 1:

n = 50

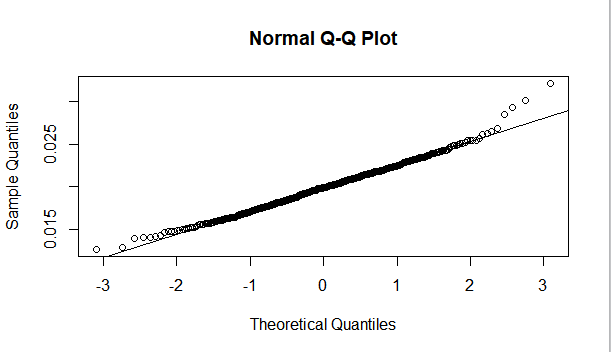
λ = 50

Section 2:

s <- replicate(500, mean(rexp(50, 50)))

qqnorm(s)

qqline(s)



Section 1:

n = 100

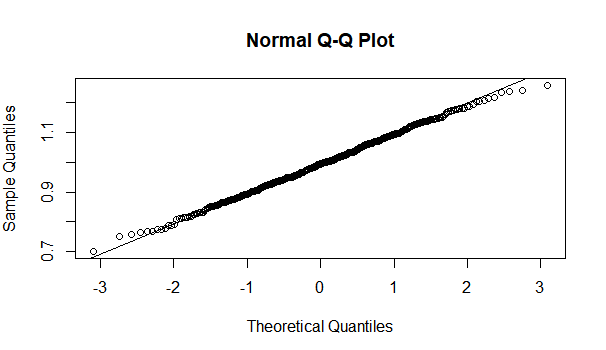
λ = 1

Section 2:

s <- replicate(500, mean(rexp(100, 1)))

qqnorm(s)

qqline(s)



Section 1:

n = 100

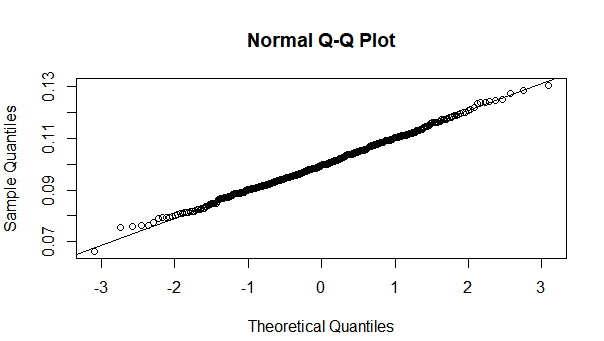
λ = 10

Section 2:

s <- replicate(500, mean(rexp(100, 10)))

qqnorm(s)

qqline(s)



Section 1:

n = 100

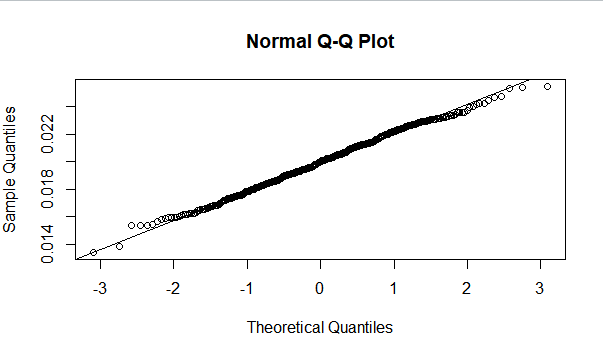
λ = 50

Section 2:

s <- replicate(500, mean(rexp(100, 50)))

qqnorm(s)

qqline(s)



**(d)**

From (b) and (c) we can see that with n = 30 itself the distribution looks approximately normal. So n can be as small as 30 for the approximation to be good.

As λ increases the curve become more concentrated and as λ decreases the curve become more spread out. But this does not affect the distribution. The distribution stays normal and hence does not depend on value of λ. This can be since from the different plots shown above, with different values of λ.