# Mini project #1

**Group Member:** Chaoran Li, Wenting Wang

**Contribution of each member:**

Firstly, we discussed the mathematical models and code details together. Then, we divided the project into two part and finished our respective work. Wenting Wang mainly worked on Q1-a and Q2 while Chaoran Li worked on Q1-b and Q1-c. Then, we merged our code and solution into one report.

Each member makes contribution to each sub task of this project and combines all to finish this project, as the details shown in table 1.

| | Question1-a | Question1-b | Question1-c | Question2 |
|---|---|---|---|---|
| Chaoran li | 20% | 80% | 80% | 20% |
| Wenting wang | 80% | 20% | 20% | 80% |

Table 1: Member contribution table

**Question 1:**

**(a)** Use the above density function to analytically compute the probability that the lifetime of the satellite exceeds 15 years.

**Solution**:

According to the description of the question, the probability that the lifetime of the satellite exceeds 15 years denoted as P(T>15) can be get by:

$$P(T > 15) = 1 - P(T \leq 15)$$

We can get $P(T \leq 15)$ with the probability density function $f_T(t)$:

$$P(T \leq 15) = \int_0^{15} (t)$$

$$= \int_0^{15} 0.2(e^{-0.1t} - e^{-0.2t})$$

$$= \left[0.2\left(\frac{e^{-0.1t}}{-0.1} - \frac{e^{-0.2t}}{-0.2}\right)\right]_0^{15}$$

$$= [e^{-0.2t} - 2e^{-0.1t}]_0^{15}$$

$$= (e^{-3} - 2e^{-1.5}) - \quad (2e^{0} - e^{0})$$

$$= e^{-3} - 2e^{-1.5} + 1$$

$$= 0.6035267$$

Thus, the probability that the lifetime of the satellite exceeds 15 years is:

$$P(T > 15) = 1 - P(T \leq 15)$$

$$= 1 - 0.603527$$

$$=0.3964733$$

**(b)** Use the following steps to take a Monte Carlo approach to compute E(T) and P (T > 15)

i.) Simulate one draw of the block lifetime $X_A$ and $X_B$. Use these draws to simulate one draw of the satellite lifetime T.

**Solution**:

```
13  lambda = 1 / 10
14  xa = rexp(n=1, rate=lambda)
15  xb = rexp(n=1, rate=lambda)
16  # we can also use runif to build a Exponential Distribution
17  myExp <- function(n, l){
18    unif = runif(n, 0, 1)
19    -log(1 - unif) / l
20  }
21  # xa = myExp(n=1, l=lambda)
22  # xb = myExp(n=1, l=lambda)
23  T0 = max(xa, xb)
24  sprintf('xa = %f, xb = %f, T = %f', xa, xb, T0)
```

```
>
> lambda = 1 / 10
> xa = rexp(n=1, rate=lambda)
> xb = rexp(n=1, rate=lambda)
> # we can also use runif to build a Exponential Distribution
> myExp <- function(n, l){
+   unif = runif(n, 0, 1)
+   -log(1 - unif) / l
+ }
> # xa = myExp(n=1, l=lambda)
> # xb = myExp(n=1, l=lambda)
> T0 = max(xa, xb)
> sprintf('xa = %f, xb = %f, T = %f', xa, xb, T0)
[1] "xa = 9.033604, xb = 9.845989, T = 9.845989"
>
```

In this question, we used two methods to get Exponential Distribution. The first one used the function rexp() directly which we would use in our following code. The second one was a self-designed function myExp() which used runif() to build an Exponential Distribution based on the mathematical model taught in our class. Both of them worked and got the right result. We wrote the second version for practicing and we recommended the first one because the

R language directly provides it.

ii.) Repeat the previous step 10000 times. This will give you 10000 draws from the distribution of T. Try to avoid 'for' loop. Use 'replicate' function instead. Save these draws for reuse in later steps. [Bonus: 1 bonus point for not taking more than 1 line of code for steps (i) and (ii).]

**Solution**:

```
31  Ts = replicate(10000, max(rexp(n=1, rate=lambda), rexp(n=1, rate=lambda)))
```

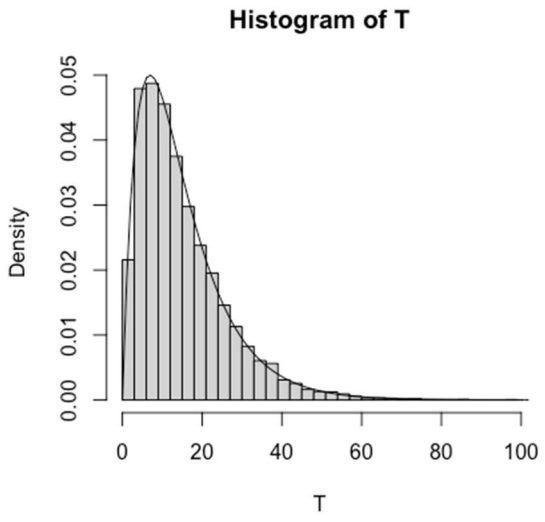
```
>
> Ts = replicate(10000, max(rexp(n=1, rate=lambda), rexp(n=1, rate=lambda)))
>
> Ts# last line has finished (i) and (ii) in one line. This line is use to show Ts
   [1]  0.1040018  3.8442511 27.8540532 25.5137443 31.7330706 21.2961336 14.0966402
   [8]  9.2737705  5.9663661  6.1934928 11.5341922  9.9413017  5.5960747  3.6377013
  [15] 13.7490294 41.1078218 24.9006392  8.2934675 31.2016822 13.2072540 25.0926645
  [22]  2.4197237 11.6540607 24.2361409 15.2140015  6.7759654 11.8436507 12.4582675
  [29] 13.4669832 13.3373543  1.2089652 19.2913334  3.6281313 30.1571391 10.1014068
```

We solved (i) and (ii) here in **one line** for **Bonus** with 'replicate' function.

iii.) Make a histogram of the draws of T using 'hist' function. Superimpose the density function given above. Try using 'curve' function for drawing the density. Note what you see.

**Solution**:

```
39  itl = 3# interval
40  hist(Ts, probability = T, breaks=seq(0, max(Ts)+itl, itl), xlab='T', main='Histogram of T')
41  pdf = function(x){
42    0.2 * exp(-0.1 * x) - 0.2 * exp(-0.2 * x)
43  }
44  curve(pdf(x), add = T, from=0, to=max(Ts)+itl, xname = 'x', xlab='T', ylab='P')
```




We drew the histogram with an interval of 3 and it fit the density function given above well. This proved that previous Exponential Distribution function worked well.

iv.) Use the saved draws to estimate E(T). Compare your answer with the exact answer given above.

**Solution**:

```
49  et = mean(Ts)# compare to 15
50  sprintf('E(T)= %f', et)
```

```
>
> et = mean(Ts)# compare to 15
> sprintf('E(T)= %f', et)
[1] "E(T)= 15.007074"
>
```

The given estimated E(T) is 15.

15.007074 > 15 (but close enough)

v.) Use the saved draws to estimate the probability that the satellite lasts more than 15 years. Compare with the exact answer computed in part(a).

**Solution**:

```
55  p15 = sum(Ts > 15) / 10000# compare answer (a)
56  sprintf('P(T>15) = %f', p15)
```

```
>
> p15 = sum(Ts > 15) / 10000# compare answer (a)
> sprintf('P(T>15) = %f', p15)
[1] "P(T>15) = 0.396300"
>
```

The probability that the satellite lasts more than 15 years this time is 0.396300 which is calculated in (a).

0.396300 < 0.3964733 (but close enough)

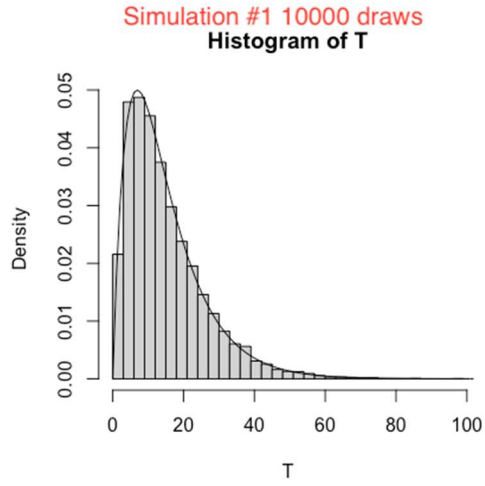
vi.) Repeat the above process of obtaining an estimate of E(T) and an estimate of the probability four more times. Note what you see.

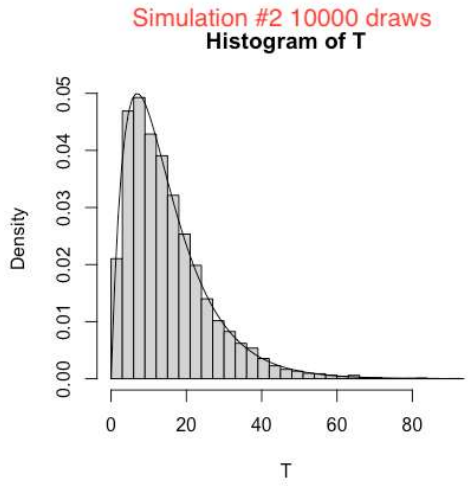
**Solution**:

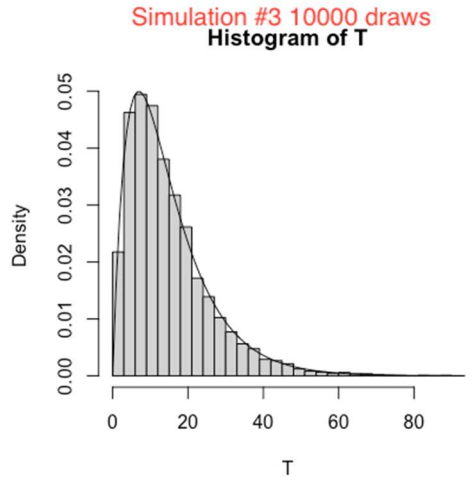
```
61 simulate = function(n, l, i){
62   Ti = replicate(n, max(rexp(n=1, rate=lambda), rexp(n=1, rate=lambda)))
63   hist(Ti, probability = T, breaks=seq(0, max(Ti)+i, i), xlab='T', main='Histogram of T')
64   curve(pdf(x), add = T, from=0, to=max(Ti)+i, xname = 'x', xlab='T', ylab='P')
65   c(mean(Ti), sum(Ti > 15) / n)
66 }
67 replicate(4, simulate(n=10000, l=lambda, i=itl))
```

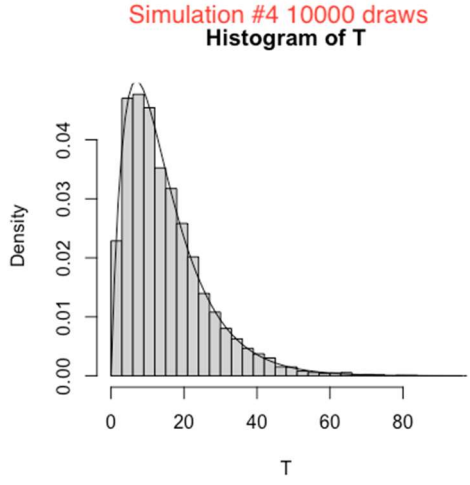
```
>
> simulate = function(n, l, i){
+   Ti = replicate(n, max(rexp(n=1, rate=lambda), rexp(n=1, rate=lambda)))
+   hist(Ti, probability = T, breaks=seq(0, max(Ti)+i, i), xlab='T', main='Histogram of T')
+   curve(pdf(x), add = T, from=0, to=max(Ti)+i, xname = 'x', xlab='T', ylab='P')
+   c(mean(Ti), sum(Ti > 15) / n)
+ }
> replicate(4, simulate(n=10000, l=lambda, i=itl))
        [,1]     [,2]     [,3]     [,4]
[1,] 15.0564 15.02805 14.84717 15.02667
[2,]  0.3962  0.40260  0.39090  0.40520
>
```

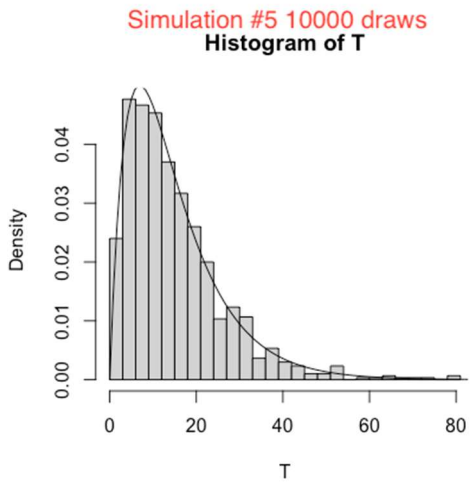
We wrote a 'simulate' function here and it could help us solve (b-iv) and (c) concisely and quickly.






| | Theoretical Value | Simulation #1 | Simulation #2 | Simulation #3 | Simulation #4 | Simulation #5 |
|---|---|---|---|---|---|---|
| E(T) | 15 | 15.007074 | 15.0564 | 15.02805 | 14.84717 | 15.02667 |
| P(T>15) | 0.3964733 | 0.3963 | 0.3962 | 0.4026 | 0.3909 | 0.4052 |

Table 2: Results of 10000 times of draw

The theoretical value and simulation value fit well for both E(T) and P(T>15). Simulation values change because of the existence of error. But all simulation values are close to the theoretical value which support the Law of Large Numbers (LLN).

**(c)** Repeat part (vi) five times using 1000 and 100000 Monte Carlo replications instead of 10000. Make a table of results. Comments on what you see and provide an explanation.

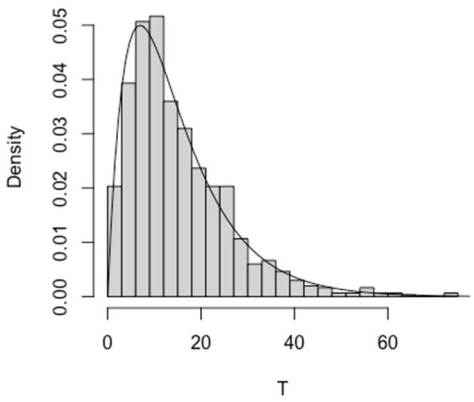
**Solution**:

Use the 'simulate' function we designed in (b-vi), we can solve (c) concisely and quickly.

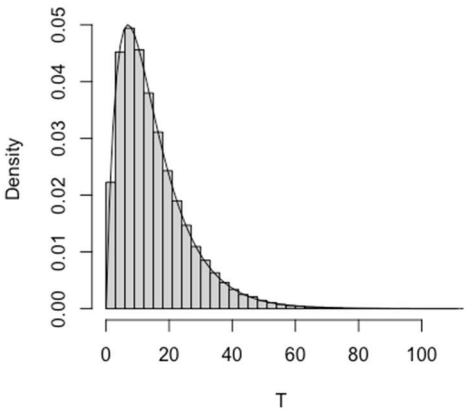
```
replicate(5, simulate(n=1000, l=lambda, i=itl))
replicate(5, simulate(n=100000, l=lambda, i=itl))
```

```
>
> replicate(5, simulate(n=1000, l=lambda, i=itl))
         [,1]     [,2]     [,3]    [,4]     [,5]
[1,] 14.90466 15.16567 14.71393 15.4954 15.40355
[2,]  0.39800  0.40600  0.39800  0.4080  0.41400
> replicate(5, simulate(n=100000, l=lambda, i=itl))
         [,1]     [,2]     [,3]     [,4]     [,5]
[1,] 15.06696 14.98300 14.97385 14.98567 15.02063
[2,]  0.39882  0.39675  0.39661  0.39465  0.39658
>
```

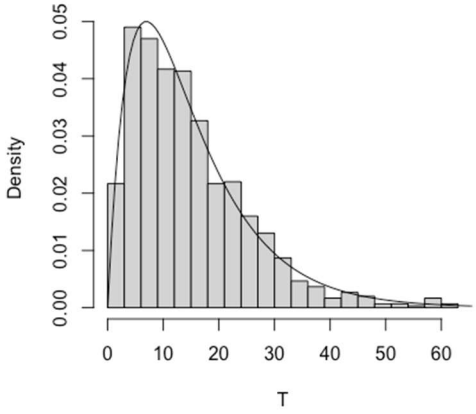
Simulation #6 1000 draws



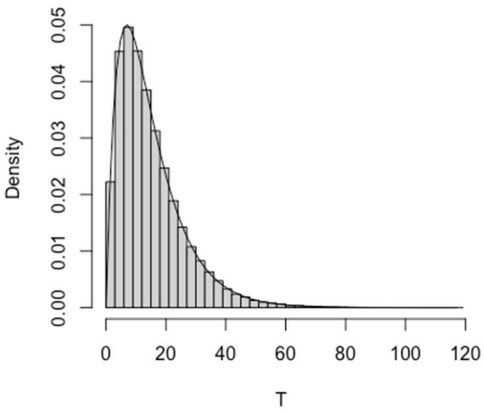

Simulation #11 100000 draws



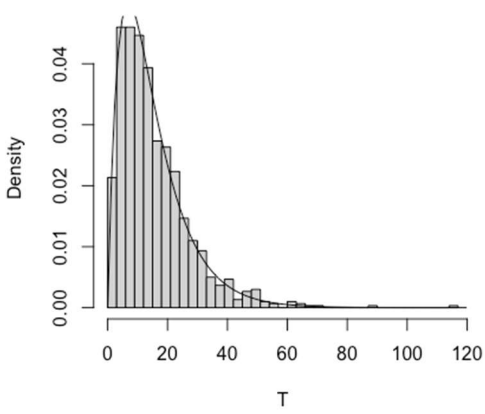

Simulation #7 1000 draws



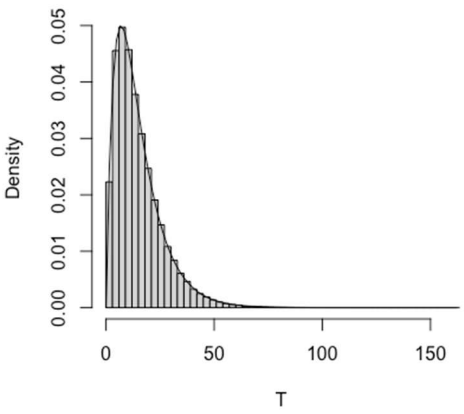

Simulation #12 100000 draws

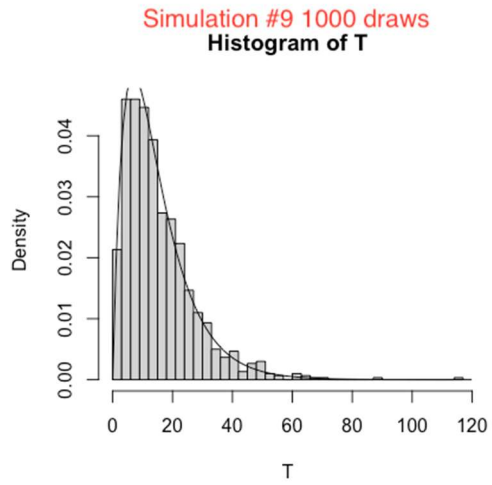


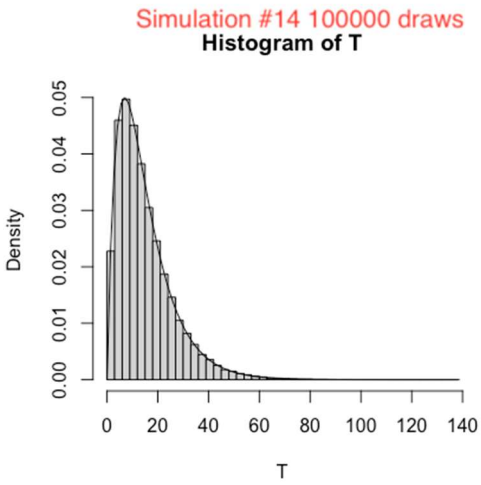

Simulation #8 1000 draws

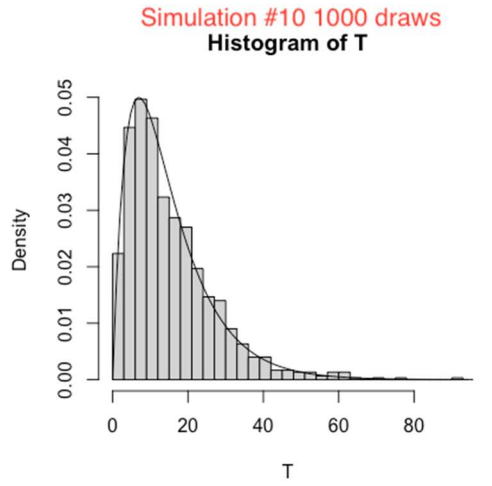


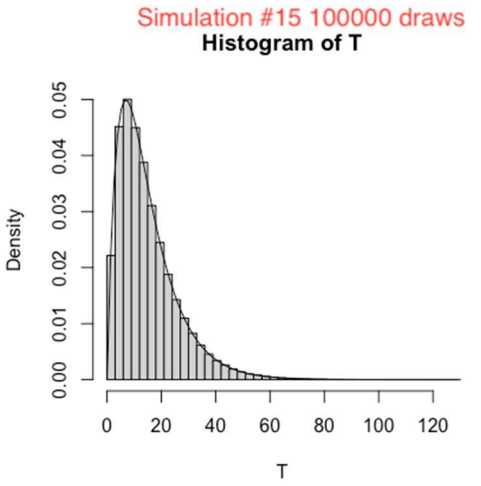

Simulation #13 100000 draws






| 1000 times | Theoretical Value | Simulation #6 | Simulation #7 | Simulation #8 | Simulation #9 | Simulation #10 |
|---|---|---|---|---|---|---|
| E(T) | 15 | 14.89791 | 14.53424 | 15.00756 | 14.74868 | 14.76678 |
| P(T>15) | 0.3964733 | 0.41 | 0.373 | 0.402 | 0.373 | 0.383 |

Table 3: Results of 1000 times of draw

| 100000 times | Theoretical Value | Simulation #11 | Simulation #12 | Simulation #13 | Simulation #14 | Simulation #15 |
|---|---|---|---|---|---|---|
| E(T) | 15 | 14.96408 | 14.98647 | 15.07583 | 14.9321 | 15.06806 |
| P(T>15) | 0.3964733 | 0.3947 | 0.3966 | 0.39963 | 0.39343 | 0.39838 |

Table 4: Results of 100000 times of draw

If the time of draw is big enough, we will get some really 'large' results which cause the whole histogram becomes wider. These results will not affect the overall distribution, but they would not be observed in small time of draw simulation.

If we use $Error(x) = 1/n \sum_{i=1}^{n}((x_{simulation} - x_{theoretica})/x_{theoretica})^2$ to evaluate the influence of times of draw on simulation results, we can have the table below:

| Times of Draw | 1000 | 10000 | 100000 |
|---|---|---|---|
| Error(E(T)) | 3.066E-4 | 2.496E-5 | 1.464E-5 |
| Error(P(T>15)) | 1.905E-3 | 1.843E-4 | 3.311E-5 |

Table 5: The Influence of times of draw on Simulation results

This result proved the Law of Large Numbers (LLN) again. While the times of draw increasing, the simulation value will be close to the theoretical value.

In addition, we found that to obtain a sufficiently accurate E(T) is always easier than P(T>15). This means that we need a relatively small time of draw if we are only interested in E(T).

**Question 2:**

**Solution**:

To estimate the value of pi, first we draw a circle with center (0.5, 0.5) and radius=0.5. Then we can get the relation between $\pi$ and the probability

that a randomly selected point in a unit square with coordinates (0, 0), (0, 1),

(1, 0) and (1, 1) falls the circle is :

$$P = \text{(Area of circle)} / \text{(Area of square)} = \frac{\pi}{4}$$

Thus, $\pi = 4 * P$

Next, we use RStudio to generate 10000 pairs of number x and y between 0 and 1. The code and output show below:

```
>
> r <- 10000                  #set up 10000 replications, the number of total random points
> x <- runif(r,min=0,max=1)  #generate 10000 random values x between 0 and 1
> y <- runif(r,min=0,max=1)   #generate 10000 random values y between 0 and 1
> circle <- (x-0.5)**2+(y-0.5)**2 <= 0.5**2   #points fall into the circle
> pi <- (sum(circle)/r)*4       #pi= 4* (the probalility that points fall into the circle)
> pi
[1] 3.1408
>
```

| values | |
|---|---|
| circle | logi [1:10000] FALSE TRUE TRUE TRUE FALSE TRUE ... |
| pi | 3.1408 |
| r | 10000 |
| x | num [1:10000] 0.8351 0.893 0.8027 0.6899 0.0171 ... |
| y | num [1:10000] 0.0452 0.6339 0.5592 0.6539 0.87 ... |

The result shows that the simulated value of $\pi$ is 3.1408 which is very close to 3.1415926 in mathematical value.