CS6313 : STATISTICAL METHODS FOR DATA SCIENCE

Mini Project-1
Group Number : 1

Names of group members: Venkatesh Sankar (VXS200014)

Manneyaa Jayasanker (MXJ180040)

Contribution of each group member: Initially ,we individually understood problem statement and each came up with possible solutions. We then discussed about approaches for all solutions and finally concurred with using the best solution out of both the codes. Since both of us are new to R language, we spent some time understanding R. **Manneyaa** wrote the documentation for 1.a, 1.b i, ii, iii, iv, v, vi and **Venkatesh** provided R code for 1.b ii, iii , 1.c and wrote documentation for 1.c and 2.

1)

a) The probability that the lifetime of the satellite exceeds 15 years can be calculated by integrating the density function over the given set of limits, as shown below :

Here, T represents the lifetime of the satellite.

$$f_T(t) = \begin{cases} 0.2e^{-0.1t} - 0.2e^{-0.2t}, & 0 \le t \le \infty \\ 0, & \text{otherwise} \end{cases}$$

$$\begin{aligned}
p(T>15) &= 1 - p(T \le 15) \\
&= 1 - \int_0^{15} f_T(t)\,dt \\
&= 1 - \int_0^{15} (0.2e^{-0.1t} - 0.2e^{-0.2t})\,dt \\
&= 1 - \left(0.2\left[\frac{e^{-0.1t}}{-0.1}\right]_0^{15} - 0.2\left[\frac{e^{-0.2t}}{-0.2}\right]_0^{15}\right) \\
&= 1 - (-2e^{-1.5} + 2 + e^{-3} - 1) \\
&= 1 - (e^{-3} - 2e^{-1.5} + 1) \\
&= 1 - [0.049 - 2(0.223) + 1] \\
&= 0.3965
\end{aligned}$$

b)

i) For an Exponential distribution we know mean is $1/\lambda = 10 \Rightarrow \lambda = 0.1$, where λ is the rate parameter.

```
> max(rexp(n=1,rate=0.1), rexp(n=1,rate=0.1))
[1] 18.494
```

ii) To simulate 10,000 draws from the distribution of T :
(Combining (i) and (ii) using replicate() which has better efficiency in memory management for large number of vector inputs compared to for loop)

t.sim10k = replicate(n = 10000, expr = max(rexp(1, 0.1), rexp(1, 0.1)))
here, n = no. of replications
expr = expression which will be executed repeatedly

Code Snippet:

```
> t.sim10k = replicate(n = 10000, expr = max(rexp(1, 0.1), rexp(1, 0.1)))
> print(t.sim10k)
   [1] 20.0239478 19.1520695 14.9676935  9.3280160 23.3199811 16.8574158  8.4160360  4.1783160 23.3103447
  [10] 34.7477544 23.0506940  1.0245704 44.8661494 27.3416373  5.1006955 16.0831656  3.4634663 18.0540198
  [19]  5.3885481 18.2166116 16.0089703 21.5915715 20.1570386  2.5237139  2.0636498 10.6436588 25.9676347
  [28] 15.2797143 12.1183418  8.0682773 12.2015230 28.5345368  8.9236123 11.6712092 17.5002404 28.6128987
  [37]  2.6414857 15.6945451  2.5414475  4.7194393  7.8559826 19.1448880 11.1501121  4.2910664  7.0014930
```

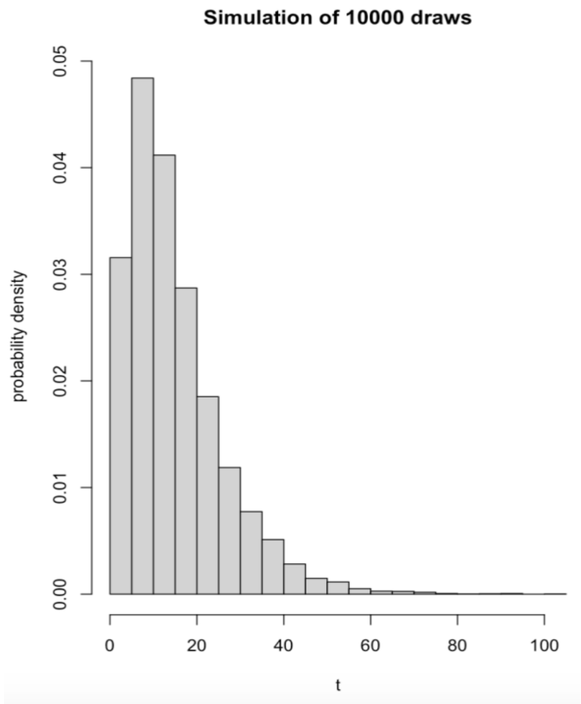
iii) To draw a histogram based on the above simulation :

hist(x = t.sim10k, freq = FALSE, main = 'Simulation of 10000 draws', xlab = 't', ylab = 'probability density')
here ,x = input vector values
freq = if FALSE, plots probability density function
main = title of histogram
xlab = x-axis label
ylab = y-axis label

Code Snippet:

```
> hist(x = t.sim10k, freq = FALSE, main = 'Simulation of 10000 draws', xlab = 't', ylab = 'probability density')
```

Output :



To superimpose density function using curve() :

curve(expr = (0.2 * exp(-0.1 * x) - 0.2 * exp(-0.2 * x)), n=10000, from=0, to= 100 , add=TRUE, col='red')

here, expr = expression/function for which the curve should be drawn

n = no. of values

from = lower limit

to = upper limit

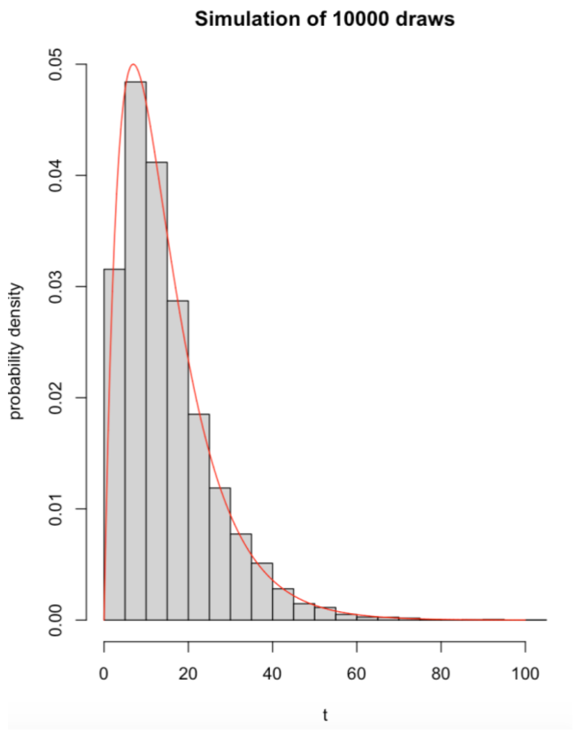
add = if TRUE , adds to an already existing plot

col = colour of the curve

Code Snippet :

```
> curve(expr = (0.2 * exp(-0.1 * x) - 0.2 * exp(-0.2 * x)), n=10000, from=0, to= 100 , add=TRUE, col='red')
```

Output :



Inference :

By superimposing the density curve over the histogram, we can visualise the distribution of data over a continuous interval. The peaks of the distribution plot help to depict the concentration of values over the interval.

iv) To estimate E(T) from saved draws:

```
> t.expectedValue = mean(t.sim10k)
> t.expectedValue
[1] 14.934
```

On printing t.expectedValue, we get E(T) = 14.934

Inference :
E(T) given in the question is 15 and the mean value obtained from the saved draws is 14.934. Thus, there is an error of ±0.0660.

v) To estimate probability of T when the lifetime T exceeds 15 from saved draws:

```
> t.probabilityValue = mean(t.sim10k>15)
> t.probabilityValue
[1] 0.3943
```

Inference :

The calculated probability of T>15 from part (a) *i.e.*, P(T > 15) = 0.3965

The estimated probability of T>15 from saved draws is P_1 (T > 15) = 0.3943

Thus, we can observe an marginal error of ±0.0022

vi) Repeating the above calculations four more times using a loop:

```
> for (i in 1:4) {
+     sim10k = replicate(n = 10000, expr = max(rexp(1, 0.1), rexp(1, 0.1)))
+     print(mean(sim10k))
+     print(mean(sim10k>15))
+
+
+ }
[1] 14.96716
[1] 0.4015
[1] 14.92354
[1] 0.3902
[1] 15.0019
[1] 0.4
[1] 14.8676
[1] 0.3925
```

Table depicting E(T) and E(T>15) for 10000 draws:

Draw #	E(T)	Error in E(T)	P(T>15)	Error in P(T>15)
1	14.9340	±0.0660	0.3943	±0.0022
2	14.9671	±0.0329	0.4015	±0.0050
3	14.9235	±0.0765	0.3902	±0.0063
4	15.0019	±0.0019	0.4000	±0.0035
5	14.8676	±0.1324	0.3925	±0.0040

Average E(T) = 14.93882
Average Error in E(T) = ±0.06194

Average P(T>15) = 0.3957
Average Error in P(T>15) = ±0.0042

Inference :

From the table above we can infer that depending on the sample of random numbers generated ,the values of expectation and probability are almost equal to the theoretically calculated values with some marginal errors.

(c) To simulate using 1000 and 100000 Monte carlo replications for 5 times.

Code Snippet :

```
> calculatemeanandprob.fun = function(totaldraws) {
+ simnk = replicate(n = totaldraws, expr = max(rexp(1, 0.1), rexp(1, 0.1)))
+ ans = c(mean = mean(simnk), probability = mean(simnk > 15))
+ return (ans)
+ }
>
> # calling the function for 1000 draws 5 times using replicate()
> replicate( n = 5, expr = calculatemeanandprob.fun(1000))
                [,1]     [,2]     [,3]     [,4]     [,5]
mean        14.79705 14.92766 15.20368 14.83765 14.88107
probability  0.38700  0.38400  0.39800  0.41200  0.38500
>
>
> # calling the function for 100000 draws 5 times using replicate()
> replicate( n = 5, expr = calculatemeanandprob.fun(100000))
                [,1]     [,2]     [,3]     [,4]     [,5]
mean        15.06888 15.01038 15.02662 15.02656 15.00771
probability  0.39688  0.39703  0.39964  0.39744  0.39795
> |
```

Table depicting E(T) and E(T>15) for 1000 draws:

Draw #	E(T)	Error in E(T)	P(T>15)	Error in P(T>15)
1	14.79705	±0.2030	0.38700	±0.0095
2	14.92766	±0.0723	0.38400	±0.0125
3	15.20368	±0.2037	0.39800	±0.0015
4	14.83765	±0.1624	0.41200	±0.0155
5	14.88107	±0.11893	0.38500	±0.0115

Average E(T) = 14.92942
Average Error in E(T) = ±0.15207

Average P(T>15) = 0.3932
Average Error in P(T>15) =± 0.0101

Table depicting E(T) and E(T>15) for 100000 draws:

Draw #	E(T)	Error in E(T)	P(T>15)	Error in P(T>15)
1	15.06888	±0.0689	0.39688	±0.0004
2	15.01038	±0.0104	0.39703	±0.0006
3	15.02662	±0.0266	0.39964	±0.0031
4	15.02656	±0.0266	0.39744	±0.0009
5	15.00771	±0.0077	0.39795	±0.0014

Average E(T) = 15.02803
Average Error in E(T) = ±0.02804

Average P(T>15) = 0.3978
Average Error in P(T>15) = ± 0.0013

Summarizing, all the 5 different trails for simulation of 1000, 10000, 100000 in a tabular form.

Total replications	Average E(T)	Average Error in E(T)	Average P(T>15)	Average Error in P(T>15)
1000	14.92942	±0.15207	0.3932	±0.0101
10000	14.93882	±0.06194	0.3957	±0.0042
100000	15.02803	±0.02804	0.3978	±0.0013

Inference :
Based on the above tabular column, we can infer that as the no.of replications are increased, the error rate is getting reduced which is based on the law of large numbers.

2. To estimate the value of π using Monte Carlo approach based on 10,000 replications. Based on the given hint from the questions i.e considering a unit square inscribed in a circle with center (0.5, 0.5), we can find the probability of a randomly selected point X to be inside the circle as below,

$$P(X) = \frac{\textit{Area of the circle}}{\textit{Area of the square}} = \frac{\pi * r^2}{a^2}$$
$$= \frac{\pi * (\frac{1}{2})^2}{(1)^2}$$
$$= \frac{\pi}{4}$$

So, to compute the value of π, we can find P(X) using R and multiply it by 4.

π = 4 * P(X)

For a point to be inside a circle with center (0.5, 0.5) and radius 0.5, we know that

$$(x - 0.5)^2 + (y - 0.5)^2 <= (0.5)^2$$

Combining both the equations, we generate random values using runif() in R and compute the probability for a point to be inside a circle from the list of randomly generated points. Then using that , we calculate the value of π as given below

```
> calculatepi.fun = function(nooftrails) {
+ pointx = runif(nooftrails)
+ pointy = runif(nooftrails)
+ targetpoints = (pointx-0.5)^2 + (pointy - 0.5)^2 <= (0.5)^2
+ prob = mean(targetpoints)
+ return (prob * 4) # pi=prob*4
+ }
> calculatepi.fun(10000)
[1] 3.1528
```

The actual value of pi in R is 3.1416. So based on 10,000 draws we calculated the value of π with a difference of ±0.0112.