**Mini Project 1 (Solution)**

**Mini Project Duo Group # 12**

**Contribution of each group member**

Chetan Siddappareddy – 50%

Ankit Sahu – 50%

Both of us have contributed equally to the project. We learnt R through collaboration and then write the R scripts for the corresponding and report all the findings.

**Section 1**

**a)** GivenfT(t) in the question, the probability that the lifetime of the satellite exceeds 15 years can be computed by integrating fT(t) over 15 to ∞ as follows:

P(T > 15) =  fT(t)dt = (0.2exp(-0.1t) – 0.2exp(-0.2t))dt

=0.2exp(-0.1t)dt – 0.2exp(-0.2t)dt

=(0.2\*exp(-0.1\*15)/0.1) – exp(-0.2\*15)

=0.44626 – 0.04978

=0.39647

almost equal to0.3965

**b)**

> ##########################

**> # R code for Section 1b #**

> ##########################

>

**> # i. below is probability density function**

> density.fun <- function(x){ return(0.2\*exp(-0.1\*x) - 0.2\*exp(-0.2\*x))}

> xOne <- replicate(1, max(rexp(1, 0.1), rexp(1, 0.1)))

> set.seed(100)

**> # ii. simulating 10000 draws from T**

> xtenK <- replicate(10000, max(rexp(1, 0.1), rexp(1, 0.1)))



**> # iii. creating a histogram**

> hist(xtenK, prob = T, ylim = c(0, 0.05), xlab = "Time T", ylab = "Probability", main = "histogram-10000 draws")

> # superimposing the density function of T

>

> curve(density.fun, from = 0, to = max(xtenK), add = T)

>

Chart, histogram

Description automatically generated

**Both the curve and histogram sees the peak at time 15, which happens to be the expected value of T.**

**> # iv. Estimating E[T] using saved draws**

> mean(xtenK)

[1] 15.04825

>

The simulated value 15.04825 is close to the actual value of E[T], which is 15.

**> # v. Estimating prob using saved draws**

> 1-pexp(15, rate = 1 / mean(xtenK))

[1] 0.3690609

>

The simulated value 0.3690609 is close to the calculated value of probability that the satellite lasts more than 15 years, which is 0.3965.

**> # vi. Repeating the process four more times**

> **# Test 1 – 10000 draws**

> xtenK <- replicate(10000, max(rexp(1, 0.1), rexp(1, 0.1)))

> mean(xtenK)

[1] 14.95094

> 1-pexp(15, rate = 1 / mean(xtenK))

[1] 0.3666743

>

> **# Test 2 - 10000 draws**

> xtenK <- replicate(10000, max(rexp(1, 0.1), rexp(1, 0.1)))

> mean(xtenK)

[1] 14.90976

> 1-pexp(15, rate = 1 / mean(xtenK))

[1] 0.3656596

>

> **# Test 3 - 10000 draws**

> xtenK <- replicate(10000, max(rexp(1, 0.1), rexp(1, 0.1)))

> mean(xtenK)

[1] 15.1616

> 1-pexp(15, rate = 1 / mean(xtenK))

[1] 0.3718214

>

> **# Test 4** **– 10000 draws**

> xtenK <- replicate(10000, max(rexp(1, 0.1), rexp(1, 0.1)))

> mean(xtenK)

[1] 14.88354

> 1-pexp(15, rate = 1 / mean(xtenK))

[1] 0.3650121

>

**Results Table for 1b:**

|  |  |  |
| --- | --- | --- |
| **Number of draws – 10000** | **E(T)** | **P(T>15)** |
| First Run | 15.04825 | 0.3690609 |
| Test 1 | 14.95094 | 0.3666743 |
| Test 2 | 14.90976 | 0.3656596 |
| Test 3 | 15.1616 | 0.3718214 |
| Test 4 | 14.88354 | 0.3650121 |

Upon repetition, the expected value and probability values are varying in the same manner, either increasing or decreasing, but remains in the proximity of the actual or calculated values in 1a. This justifies the central limit theorem.

**c)**

**> ##########################**

**> # R code for Section 1c #**

**> ##########################**

>

**> # Test 1 - 1000 draws - oneK => 1000**

> xOneK <- replicate(1000, max(rexp(1, 0.1), rexp(1, 0.1)))

> hist(xOneK, prob = T, ylim = c(0, 0.05), xlab = "Time T", ylab = "Probability", main = "histogram-1000 draws")

> mean(xOneK)

[1] 14.99978

> 1-pexp(15, rate = 1 / mean(xOneK))

[1] 0.3678741

>

**> # Test 2 - 1000 draws - oneK => 1000**

> xOneK <- replicate(1000, max(rexp(1, 0.1), rexp(1, 0.1)))

> hist(xOneK, prob = T, ylim = c(0, 0.05), xlab = "Time T", ylab = "Probability", main = "histogram-1000 draws")

> mean(xOneK)

[1] 14.46897

> 1-pexp(15, rate = 1 / mean(xOneK))

[1] 0.3546224

>

**> # Test 3 - 1000 draws - oneK => 1000**

> xOneK <- replicate(1000, max(rexp(1, 0.1), rexp(1, 0.1)))

> hist(xOneK, prob = T, ylim = c(0, 0.05), xlab = "Time T", ylab = "Probability", main = "histogram-1000 draws")

> mean(xOneK)

[1] 14.59851

> 1-pexp(15, rate = 1 / mean(xOneK))

[1] 0.3578997

>

**> # Test 4 - 1000 draws - oneK => 1000**

> xOneK <- replicate(1000, max(rexp(1, 0.1), rexp(1, 0.1)))

> hist(xOneK, prob = T, ylim = c(0, 0.05), xlab = "Time T", ylab = "Probability", main = "histogram-1000 draws")

> mean(xOneK)

[1] 15.2096

> 1-pexp(15, rate = 1 / mean(xOneK))

[1] 0.3729841

>

**> # Test 5 - 1000 draws - oneK => 1000**

> xOneK <- replicate(1000, max(rexp(1, 0.1), rexp(1, 0.1)))

> hist(xOneK, prob = T, ylim = c(0, 0.05), xlab = "Time T", ylab = "Probability", main = "histogram- 1000 draws")

> mean(xOneK)

[1] 15.15149

> 1-pexp(15, rate = 1 / mean(xOneK))

[1] 0.371576

>

**> # Test 1 - 100000 draws - hunK => 100000**

> xhunK <- replicate(100000, max(rexp(1, 0.1), rexp(1, 0.1)))

> hist(xhunK, prob = T, ylim = c(0, 0.05), xlab = "Time T", ylab = "Probability", main = "histogram-hunK draws")

> mean(xhunK)

[1] 14.95058

> 1-pexp(15, rate = 1 / mean(xhunK))

[1] 0.3666655

>

**> # Test 2 - 100000 draws - hunK => 100000**

> xhunK <- replicate(100000, max(rexp(1, 0.1), rexp(1, 0.1)))

> hist(xhunK, prob = T, ylim = c(0, 0.05), xlab = "Time T", ylab = "Probability", main = "histogram-hunK draws")

> mean(xhunK)

[1] 15.02762

> 1-pexp(15, rate = 1 / mean(xhunK))

[1] 0.3685562

>

**> # Test 3 - 100000 draws - hunK => 100000**

> xhunK <- replicate(100000, max(rexp(1, 0.1), rexp(1, 0.1)))

> hist(xhunK, prob = T, ylim = c(0, 0.05), xlab = "Time T", ylab = "Probability", main = "histogram-hunK draws")

> mean(xhunK)

[1] 15.0187

> 1-pexp(15, rate = 1 / mean(xhunK))

[1] 0.3683378

>

**> # Test 4 - 100000 draws - hunK => 100000**

> xhunK <- replicate(100000, max(rexp(1, 0.1), rexp(1, 0.1)))

> hist(xhunK, prob = T, ylim = c(0, 0.05), xlab = "Time T", ylab = "Probability", main = "histogram-hunK draws")

> mean(xhunK)

[1] 15.01801

> 1-pexp(15, rate = 1 / mean(xhunK))

[1] 0.3683208

>

**> # Test 5 - 100000 draws - hunK => 100000**

> xhunK <- replicate(100000, max(rexp(1, 0.1), rexp(1, 0.1)))

> hist(xhunK, prob = T, ylim = c(0, 0.05), xlab = "Time T", ylab = "Probability", main = "histogram-hunK draws")

> mean(xhunK)

[1] 14.9987

> 1-pexp(15, rate = 1 / mean(xhunK))

[1] 0.3678476

**Results Table for 1c:**

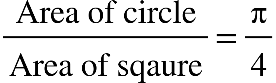
|  |  |  |
| --- | --- | --- |
| **Number of draws – 1000** | **E(T)** | **P(T>15)** |
| Test 1 | 14.99978 | 0.3678741 |
| Test 2 | 14.46897 | 0.3546224 |
| Test 3 | 14.59851 | 0.3578997 |
| Test 4 | 15.2096 | 0.3729841 |
| Test 5 | 15.15149 | 0.371576 |
| **Number of draws – 100000** | **E(T)** | **P(T>15)** |
| Test 1 | 14.95058 | 0.3666655 |
| Test 2 | 15.02762 | 0.3685562 |
| Test 3 | 15.0187 | 0.3683378 |
| Test 4 | 15.01801 | 0.3683208 |
| Test 5 | 14.9987 | 0.3678476 |

It can be inferred from the above results of mean and probability estimation using 100000 draws that there is not much deviation from the corresponding manually calculated values in section 1a. However, as the number of draws reduces, say 1000, the deviation of mean E[T] and P(T > 15) from the manually computed values increases.

**Section 2**

From the given hint, consider that a circle is inscribed in a square of unit area. Square of unit area can be simulated by taking independent draws from a Uniform (0, 1) distribution for each of x and y- coordinates. Intersection point of diagonals of the unit square and center of circle coincides at (1/2, 1/2).

The probability of a point ‘p’ falling in the circle inscribed in a unit square is as follows:

P(p) = 

This implies that straight pi is four times P(p).

For a point to be within the circle, it must be within the radius of the circle. Radius of the circle is 0.5. Hence to get the probability of points within the circle

P(p) = P open parentheses square root of open parentheses x minus 0.5 close parentheses squared plus open parentheses y minus 0.5 close parentheses squared less or equal than 0.5 end root close parentheses

straight pi = P(p) \* 4

Below is the RCode implemented based on above Monte Carlo approach.

**> ##########################**

**> # R code for Section 2 #**

**> ##########################**

> numberOfIterations <- 10000

> x <- runif(numberOfIterations, min = 0, max = 1)

> y <- runif(numberOfIterations, min = 0, max = 1)

> inside.Circle <- (sqrt((x-0.5)^2 + (y-0.5)^2) <= 0.5)

> MonteCarlo.pi <- (sum(inside.Circle)/numberOfIterations)\*4

> MonteCarlo.pi

[1] **3.1448**

Table

Description automatically generated with low confidence



The value of straight pi simulated using R is 3.1448 which is very much closer to the actual value of straight pi 3.14159.