Mini-Project 1

Section 1:

Problem 1

a. Lifetime of Block A: X_A. And $E(X_A) = 10$
Lifetime of Block B: X_B. And $E(X_B) = 10$
Lifetime of Satellite $T = max(X_A, X_B)$
$E(T) = 15$

$$f_t(T) = \left\{ \begin{array}{ll} 0.2\,e^{-0.1t} - 0.2\,e^{-0.2t}, & 0 \leq t \leq \infty \\ 0, & \text{otherwise} \end{array} \right\}$$

$$P(T > 15) = \int_{15}^{\infty} 0.2\,e^{-0.1t} - 0.2\,e^{-0.2t}\,\mathrm{d}t \tag{1}$$

$$= 0.2\left[\frac{-e^{-0.1t}}{0.1} + \frac{e^{-0.2t}}{0.2}\right]_{15}^{\infty} \tag{2}$$

$$= \left[-2\,e^{-0.1t} + e^{-0.2t}\right]_{15}^{\infty} \tag{3}$$

$$= \left[-2\,e^{-\infty} + e^{-\infty}\right] - \left[-2\,e^{-1.5} + e^{-3}\right] \tag{4}$$

$$P(T > 15) = \frac{2}{e^{1.5}} - \frac{1}{e^3} = 0.39647 \tag{5}$$

Thus, the probability of the satellite lasting longer than 15 years is 39.65%

b. Covered in code section.

Problem 2

Estimating Pi - Discussed in Section 2

Section 2:

Problem 1

a. Covered in problem section.

b. i. Simulating 1 draw of the lifetime of a satellite using 1 draw of components A & B in one line of code.

Snippet 1: R Code

```
# max of one draw of A and B each
# one draw of either is an exponential dist. w/ freq. 0.1
oneSatelliteLifeDraw = max(rexp(1, 1/10), rexp(1, 1/10))
```

ii. Repeating the process 10,000 times without a for loop and in one line of code.

Snippet 2: R Code

```r
# the exponential dist. generator can produce an array of values
# using pairwise maximum, we can get an array of T, size 10k
nSatelliteLifeDraws = pmax(rexp(10000, 1/10), rexp(10000, 1/10))
```

iii. Plotting the histogram of draws and superimposing with the curve of the pdf, additionally, the line in red is the density of the histogram plotted (see Figure 1).

Snippet 3: R Code

```r
nSatelliteLifeDraws = pmax(rexp(10000, 1/10), rexp(10000, 1/10))
# defining the pdf function
pdf = function(z) 0.2 * (exp(-0.1*z) - exp(-0.2*z))
# plotting the histogram
hist(nSatelliteLifeDraws, freq = FALSE)
# plotting the function overlay
curve(pdf, from=0, to=80, add=TRUE)
# plotting the density curve of the histogram in red
lines(density(nSatelliteLifeDraws), col='red')
# adding a legend to the plot
legend(x='topright', legend = c("PDF", "Hist␣Density"), lty=c(1,1),
    ↪ col=c(1,2))
```

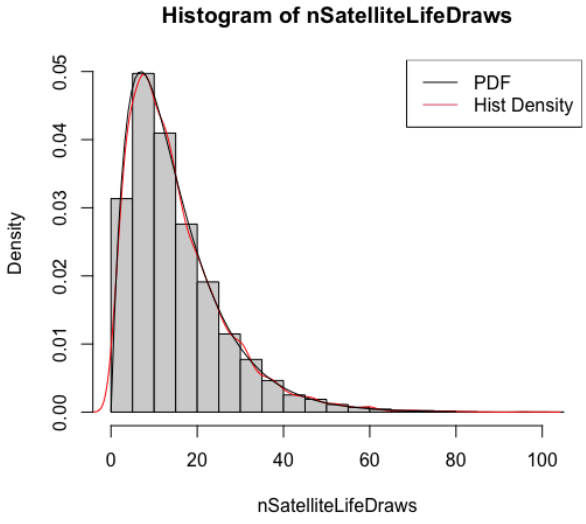

Figure 1: Histogram of Satellite Life in years

iv. $E(T)$ is simply the mean of all the draws of T

Snippet 4: R Code

```
> meanSatelliteLife = mean(nSatelliteLifeDraws)
> meanSatelliteLife # the exact value of E(T) is 15
[1] 14.98626
```

We can see the estimated expectation of T is within 0.0091% of the calculated actual $E(T)$.

v. Calculated $P(T > 15) = 0.3964$

Snippet 5: R Code

```
# number of satellites lasting > 15 divided by the total number of
   ↪ satellites
> sum( nSatelliteLifeDraws > 15) / length(nSatelliteLifeDraws)
[1] 0.3899
```

We can see that the estimated $P(T)$ is within 1.64% of the calculated value.

vi. Repeating the steps in step v to see consistency of results.

Snippet 6: R Code

```
> # a function to simulate n satellite life draws
> nSatelliteDraws = function(n) pmax(rexp(n, 1/10), rexp(n, 1/10)
> # a function calculate the mean and the prob satellite lives more
   ↪ than 15 years as c(mean, probability)
> meanAndP15.plus = function(x) c(mean(x), sum(x>15)/length(x))
> # repeat the process 4 times for 10k draws
> meansAndProbs = replicate(4, meanAndP15.plus(nSatelliteDraws(10000))
   ↪ )
> meansAndProbs
        [,1] [,2] [,3] [,4]
[1,] 14.82187 14.88929 15.20864 15.02049
[2,] 0.38860 0.39530 0.40950 0.39880
> means = meansAndProbs[1,]
> probs = meansAndProbs[2,]
# the mean of all expectations of T, is close the the actual value of
   ↪ 15
> mean(means)
[1] 14.98507
# the probability of a satellite living more than 15 years is close to
   ↪  the calculated probability 0.3964
> mean(probs)
[1] 0.39805
```

c. Repeating vi with 1000 and 100,000 draws. Please check code comments for the explanation of observations made. The table generated is for 1000, 10k and 100k draws from left to right.
A pair of rows represent the result for the n-th iteration. Where the pair of rows are 2n and 2n-1 for n in [1, 5]
Odd numbered rows represent the expectation of satellite life
Even numbered rows represent the probability only after 15 years have passed.

Snippet 7: R Code

```
> # a function to simulate n satellite life draws
> nSatelliteDraws = function(n) pmax(rexp(n, 1/10), rexp(n, 1/10)
> # a function calculate the mean and the prob satellite lives more than
    ↪ 15 years as c(mean, probability)
> meanAndP15.plus = function(x) c(mean(x), sum(x>15)/length(x))
> multipleSimulationsWrapperFor.nDraws = function(n) replicate(5,
    ↪ meanAndP15.plus(nSatelliteDraws(n)))
> simInfo = mapply(multipleSimulationsWrapperFor.nDraws, c(1000, 10000,
    ↪ 100000))
> simInfo
          [,1] [,2] [,3]
 [1,] 14.71305 15.07337 15.02949
 [2,] 0.38600 0.39660 0.39767
 [3,] 15.38976 15.02609 14.98939
 [4,] 0.41600 0.39730 0.39626
 [5,] 14.55127 15.07423 14.97308
 [6,] 0.39600 0.39150 0.39528
 [7,] 15.18423 14.99074 15.01255
 [8,] 0.38800 0.39970 0.39654
 [9,] 14.17917 14.97434 14.97907
[10,] 0.36500 0.39700 0.39545


# separating the means and the probabilities
> extractMeans = function(x) {simInfo[seq(1,10,2), x]}
> extractMeans(1:3)
         [,1] [,2] [,3]
[1,] 15.33669 15.11678 14.98671
[2,] 14.41929 15.01776 15.00655
[3,] 15.09954 15.06258 15.00169
[4,] 15.41968 15.12550 14.97042
[5,] 14.72374 15.00424 14.97040
> extractProbs = function(x) {simInfo[seq(2,10,2), x]}
```

```
> extractProbs(1:3)
      [,1] [,2] [,3]
[1,] 0.410 0.4003 0.39518
[2,] 0.384 0.3974 0.39502
[3,] 0.391 0.3944 0.39565
[4,] 0.410 0.3975 0.39546
[5,] 0.397 0.3954 0.39520


# we can see that as the number of draws increase from left to right, the
    ↪ average of the expectation of the 5 simulations tends to get closer
    ↪  to the calculated expectation.
> colMeans(extractMeans(1:3))
[1] 14.80349 15.02775 14.99672
# also, the standard deviation of the estimated expectations across 5 sims
    ↪  drops with increase in number of draws, showing that more draws
    ↪ better model the continuous distribution
> apply(extractMeans(1:3), 2, sd)
[1] 0.48740409 0.04601106 0.02370948

# the same pattern is seen for the probability of a satellite living more
    ↪ than 15 years, and the mean probs across 5 sims gets closer to the
    ↪ calculated probability of 0.3964
> colMeans(extractProbs(1:3))
[1] 0.39020 0.39642 0.39624
# while the average of probs seems to break the pattern a bit, we can
    ↪ better observe with the sd values that with more draws, the sd of
    ↪ the probabilities decreases showing a tendency to predict more
    ↪ accurately and precisely.
> apply(extractProbs(1:3), 2, sd)
[1] 0.0184173831 0.0030044966 0.0009592966
```

Problem 2

Estimating Pi - Consider a square from -1 to 1 on both the x and y axes. Also, consider a circle of radius 1 centred at the origin. The ratio of the area of the circle, A_C, to the area of the square, A_S, is given by

$$\frac{A_C}{A_S} = \frac{\pi * r^2}{s^2} = \frac{N_C}{N_S} \tag{6}$$

Since $r = 1$ and $s = 2$, Thus Pi is equal to

$$\frac{N_C}{N_S} = \frac{\pi}{4} \tag{7}$$

For a random uniform distribution of both x and y from -1 to 1, the number of points within the circle N_C will represent the area of the circle while the number of points overall N_S will represent the area of the square. Thus, to calculate the value of pi, we will use the following equation

$$\pi \approx \frac{4 * N_C}{N_S} \tag{8}$$

Snippet 8: R Code

```
> # a function to calculate the euclidean distance of a point from origin
> dist = function(x, y) sqrt(x^2 + y^2)
> # total number of points in the square, i.e. the number of replications
> numberOfPointsInSquare = 10000
> # 10k random numbers between -1,1 for the x and y axes
> x = runif(numberOfPointsInSquare, -1, 1)
> y = runif(numberOfPointsInSquare, -1, 1)
> # distances of the 10k points from the origin
> dists = dist(x,y)
> # the number of points within the circle, i.e. within a distance of 1 from
  ↪ the center
> numberOfPointsInCircle = sum(dists <= 1)
> # using the formula from above, (4*N_C)/N_S
> pi.estimate = (4*numberOfPointsInCircle)/numberOfPointsInSquare
> pi.estimate
[1] 3.1428
> # pi fairly close to it's value of 3.14159
```