# MAST20005/MAST90058: Week 10 Lab Solutions

1. The theoretical pdf for $X_{(1)}$ in this case is

$$g_1(x) = 4e^{-4x}, \quad x > 0.$$

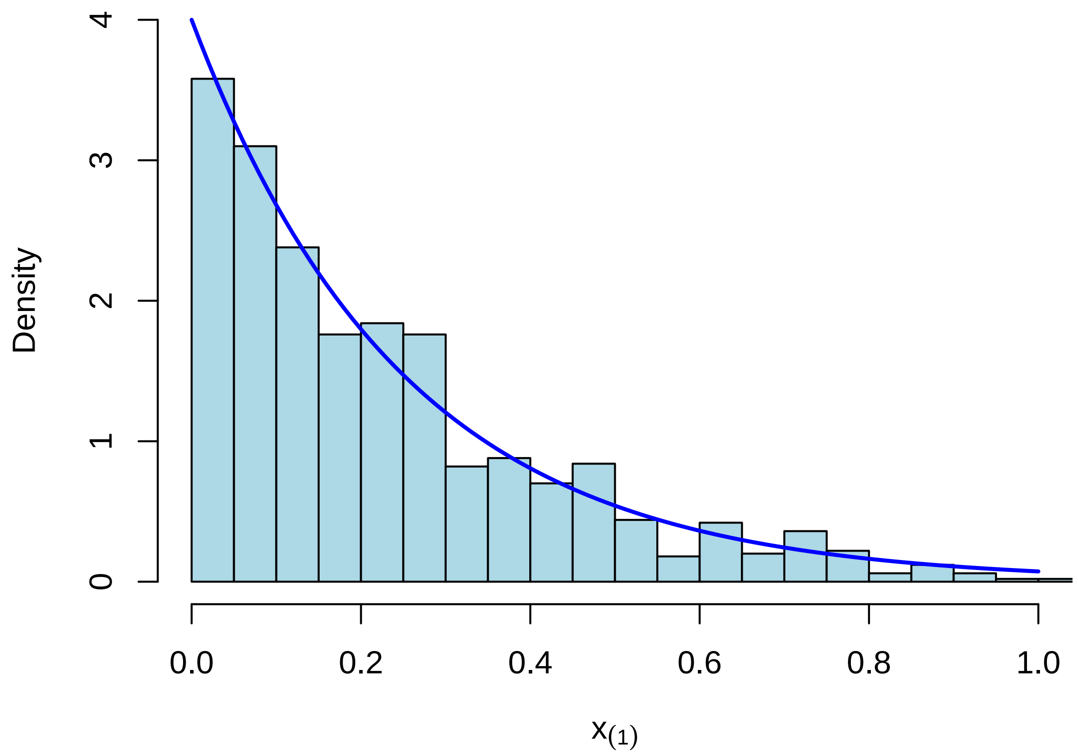
   Doing the simulations:

```
x1.simulated <- numeric(1000)  # initialise an empty vector
for (i in 1:1000) {
    x <- rexp(4)
    x1.simulated[i] <- min(x)
}
```

   Then creating the plots:

```
g <- function(x)
    4 * exp(-4 * x)

hist(x1.simulated, breaks = 50, freq = FALSE, col = "lightblue",
     xlim = c(0, 1), ylim = c(0, g(0)),
     main = NULL, xlab = expression(x[(1)]))
curve(g, from = 0, to = 1, add = TRUE, col = "blue", lwd = 2)
```



2. Note that we can simulate from this distribution by simulating from a standard (unshifted) exponential distribution and adding $\theta$ to each realisation.

```
t.simulated <- matrix(nrow = 1000, ncol = 2)  # initialise an empty matrix
for (i in 1:1000) {
    x <- 3 + rexp(10)  # theta = 3, n = 10
    x <- sort(x)
    t1 <- mean(x) - 1
    t2 <- x[1] - 0.1
    t.simulated[i, ] <- c(t1, t2)
}
```

Now, let's compare the estimators.

```
apply(t.simulated, 2, mean)

## [1] 2.976587 2.999145

apply(t.simulated, 2, var)

## [1] 0.094744084 0.009470275
```

We see that both estimators seem to be unbiased (means are close to $\theta = 3$) and that the variance of $T_2$ is about 10 times smaller than that of $T_1$.

3. For compactness, we present the solutions in a different order to the question parts. Firstly, note that since the population distribution is uniform on a unit interval centered on $\theta$, $X_{(3)}$ will have a beta distribution that has been shifted accordingly. On the standard unit interval it would have an expectation of $3/4$; after shifting this gets mapped to $\theta + 1/4$. Therefore, we need $a = -0.25$ to make $W_5$ unbiased.

Doing all of the simulations:

```
theta <- 0.5
w.simulated <- matrix(nrow = 1000, ncol = 5)  # initialise an empty matrix
for (i in 1:1000) {
    x <- runif(3, theta - 0.5, theta + 0.5)
    x <- sort(x)
    w1 <- mean(x)
    w2 <- x[2]
    w3 <- (x[1] + x[3]) / 2
    w4 <- x[3] - 0.5
    w5 <- x[3] - 0.25
    w.simulated[i, ] <- c(w1, w2, w3, w4, w5)
}
apply(w.simulated, 2, mean) - theta  # bias

## [1]  0.006302829  0.006275741  0.006316373 -0.241077574  0.008922426

apply(w.simulated, 2, var)

## [1] 0.02739359 0.04906474 0.02472136 0.03646129 0.03646129
```

(a) The simulations estimate the bias to be $-0.241$ (we know the true bias is $-0.25$).
(b) $a = 0.25$
(c) The estimated variance of $W_5$ is lower than that of $W_2$ but higher than the other estimators.

4. Set up and run the simulations:

```
f <- function() {
    X <- runif(11)
    Y <- sort(X)
    c(Y[2], Y[9])
}
nsimulations <- 100
c1 <- t(replicate(nsimulations, f()))
inside.interval <- (c1[, 1] < 0.5) & (0.5 < c1[, 2])
```

Note that we end up with a simulation of a binary (Bernoulli) variable and wish to estimate the proportion. We can use the standard method of calculating a confidence interval for a proportion:

```
prop.test(sum(inside.interval), nsimulations)$conf.int

## [1] 0.9375880 0.9994778
## attr(,"conf.level")
## [1] 0.95
```

Repeat for 1000 simulations:

```
nsimulations <- 1000
c1 <- t(replicate(nsimulations, f()))
inside.interval <- (c1[, 1] < 0.5) & (0.5 < c1[, 2])
prop.test(sum(inside.interval), nsimulations)$conf.int

## [1] 0.9557545 0.9784915
## attr(,"conf.level")
## [1] 0.95
```

This interval is much narrower.

5. 
```
x <- c(0.67, 2.46, 1.00, 8.89, 8.85, 28.45, 2.95,
       2.36, 0.37, 5.66, 6.26, 1.80,  1.88, 4.66)
n <- length(x)
```

The ‘automatic’ way of doing this is to simply rely on `qbinom()`:

```
qbinom(c(0.025, 0.975), n, 0.5)

## [1]  3 11
```

```r
sort(x)[c(3, 11)]

## [1] 1.00 6.26

pbinom(10, n, 0.5) - pbinom(2, n, 0.5)  # exact coverage

## [1] 0.9648438
```

Note that this gives a slightly different answer to the more considered one that is given in the solution for tutorial problem.

6. Calculate $T$:

```r
x <- c(0.252, 0.287, 0.537, 0.511, 0.054,
       0.022, 0.142, 0.021, 0.155, 0.241)
t <- 1 / mean(x)
t

## [1] 4.50045
```

Generate the bootstrapped statistics:

```r
B <- 1000
t.boot <- numeric(B)
for (i in 1:B) {
    x.ast <- sample(x, size = 25, replace = TRUE)
    t.boot[i] <- 1 / mean(x.ast)
}
```

Calculate a 95% confidence interval:

```r
quantile(t.boot, c(0.025, 0.975))

##     2.5%    97.5%
## 3.370262 6.269043
```