Lab 05: The Count of Monte Carlo – SOLUTIONS

PSTAT 100, Summer Session A 2025 with Ethan P. Marzban

MEMBER 1 (NetID 1) MEMBER 2 (NetID 2)
MEMBER 3 (NetID 3)

July 10, 2025

Required Packages

```
library(ottr)        # for checking test cases (i.e. autograding)
library(pander)      # for nicer-looking formatting of dataframe outputs
library(tidyverse)   # for graphs, data wrangling, etc.
```

Logistical Details

Logistical Details

- This lab is due by **11:59pm on Friday, July 11, 2025**.
- Collaboration is allowed, and encouraged!
  - If you work in groups, list ALL of your group members' names and NetIDs (not Perm Numbers) in the appropriate spaces in the YAML header above.
  - Please delete any "MEMBER X" lines in the YAML header that are not needed.
  - No more than 3 people in a group, please.
- Ensure your Lab properly renders to a `.pdf`; non-`.pdf` submissions will not be graded and will receive a score of 0.
- Ensure all test cases pass (test cases that have passed will display a message stating `"All tests passed!"`)

Lab Overview and Objectives

Welcome to another PSTAT 100 Lab! In this lab, we will cover the following:

- Inversion sampling
- Basic Monte Carlo methods

Part I: Sampling from Distributions

By now, you've hopefully realized that simulating samples from a given distribution is an incredibly useful tool. Up until now, we've been accomplishing this type of sampling using built-in `R` functions; for example, to sample from the Exponential distribution, we used `rexp()`. Sometimes it becomes useful or necessary to sample from distributions that *don't* have built-in `R` functions.

In general, generating such "nonstandard" samples is quite involved, and encompasses an active area of research. There is, however, one particularly useful result that we will make use of in this lab, called the **Probability Integral Transform**:

> **Probability Integral Transform**
>
> Let X be a random variable that follows a distribution with CDF $F_X(\cdot)$. Then the random variable $U := F_X(X)$ follows a Uniform distribution on the interval $[0, 1]$.

The proof of this, though quite straightforward, is out-of-scope for PSTAT 100. For now, we'll simply leverage this result to note that we can generate samples from an arbitrary distribution with CDF $F_X(x)$ by:

1) Taking a sample from the Uniform$[0, 1]$ distribution, using `runif()`,
2) Applying the inverse CDF $F_X^{-1}()$ to our sampled values.

Such a sampling scheme is often called **inversion sampling**, for its use of the *inverse* CDF (which we often call the **quantile function** of a distribution).

As a simple example of inversion sampling, let's generate a sample from the Exponential distribution. We know, from PSTAT 120A, that if $X \sim \text{Exp}(\lambda)$ (where this notation means $\mathbb{E}[X] = (1/\lambda)$), the CDF of X is given by

$$F_X(x) = \begin{cases} 1 - e^{-\lambda x} & \text{if } x \geq 0 \\ 0 & \text{otherwise} \end{cases}$$

and the associated quantile function is given by

$$F_X^{-1}(p) = -\frac{1}{\lambda}\ln(1-p)$$

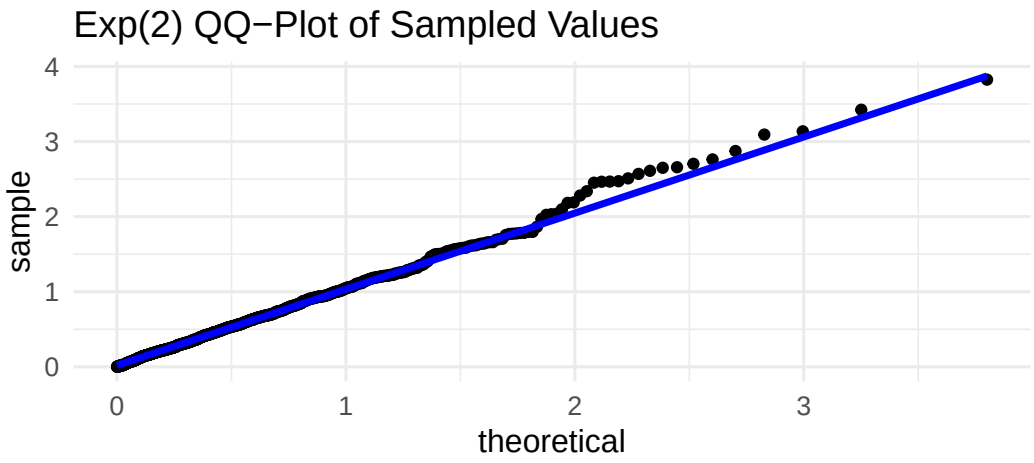
Therefore, a sample from the Exp(2) distribtuion can be obtained using:

```
set.seed(100)
U <- runif(1000)
X <- (-1/2) * log(1 - U)
```

To confirm that `X` follows an Exponential distribution with rate 2, we can use a **QQ-Plot**:

```
data.frame(X) %>% ggplot(aes(sample = X)) +
  geom_qq(distribution = stats::qexp, dparams = list(rate = 2)) +
  geom_qq_line(distribution = stats::qexp, dparams = list(rate = 2),
               color = "blue", linewidth = 1.25) +
  theme_minimal(base_size = 12) +
```

```
    xlab("theoretical") + ylab("sample") +
    ggtitle("Exp(2) QQ-Plot of Sampled Values")
```



Over the next few questions, we'll develop a function to generate samples from the distribution with probability density function given by

$$f_X(x) = \begin{cases} (x/2) & \text{if } 0 \leq x \leq 2 \\ 0 & \text{otherwise} \end{cases} \tag{1}$$

Keep in mind that you may need to do some scratch work (Calculus) on the side, before coding!

! Question 1

Write a function `pdist1()` that takes in a single argument `x` and returns the CDF of the distribution with density given by (1) at `x`. Make sure your function is vectorized: as a hint, consider using the `Vectorize()` function in `R`.

Solution:

```
## replace this line with your code
pdist1 <- Vectorize(function(x) {
  if(x <= 0) {return(0)}
  else if((0 <= x) & (x <= 2)) {return(x^2 / 4)}
  else {return(1)}
})
```

Answer Check:

```
# DO NOT EDIT THIS LINE
invisible({check("tests/q1.R")})
```

```
All tests passed!
```

! Question 2

Write a function `qdist1()` that takes in a single argument `p` and returns the quantile function (i.e. the inverse CDF) of the distribution with density given by (1) at `x`. Make sure your function is vectorized: as a hint, consider using the `Vectorize()` function in `R`.

IMPORTANT: Due to a quirk in `R`, you need to make sure the argument for your function is called `p`, and not a different letter so that we can use this function in Question 3 below.

Solution:

```
## replace this line with your code
qdist1 <- Vectorize(function(p) {
  if(p <= 0) {return(0)}
  else if((0 <= p) & (p <= 1)) {return(2 * sqrt(p))}
  else {return(2)}
})
```

Answer Check:

```
# DO NOT EDIT THIS LINE
invisible({check("tests/q2.R")})
```

```
All tests passed!
```

! Question 3

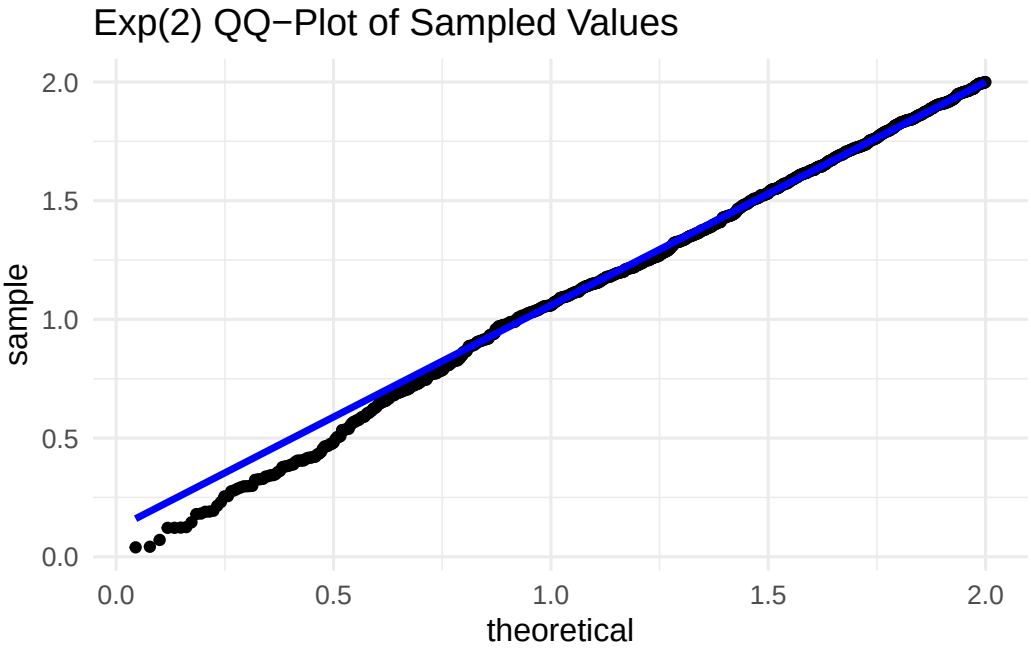
Write a function `rdist1()` that takes in a single argument `n` and returns a sample of size n from the distribution with density given by (1) at `x`.

Then, use your `rdist1()` function to generate a sample of size 1000 from the distribution with density given by (1), and plot a QQ-plot comparing the quantiles of this sample to those given by the density. As a hint: copy-paste the code used to generate a QQ-plot above, change the `distribution` argument, and remove the parameter specifications.

Solution:

```r
## replace this line with your code
rdist1 <- function(n) {
  return(qdist1(runif(n)))
}

set.seed(100)
X <- rdist1(1000)
data.frame(X) %>% ggplot(aes(sample = X)) +
  geom_qq(distribution = qdist1) +
  geom_qq_line(distribution = qdist1,
               color = "blue", linewidth = 1.25) +
  theme_minimal(base_size = 12) +
    xlab("theoretical") + ylab("sample") +
    ggtitle("Exp(2) QQ-Plot of Sampled Values")
```



Answer Check:
There is no autograder for this question; your TA will manually check that your answers are correct.

Part II: Monte Carlo Methods

In lecture, we were introduced to the notion of **Monte Carlo Methods**. Generally speaking, Monte Carlo methods are those that utilize *stochastic* (random/probabilistic) methods to estimate *deterministic* quantities.

Here is the general idea. Suppose we wish to evaluate

$$I := \int_a^b g(x)\ \mathrm{d}x$$

for some function $g(x)$. Further suppose that $g(\cdot)$ is such that computing I analytically is intractable. The key is to note that

$$I = \int_a^b g(x)(b-a) \cdot \frac{1}{b-a} \, \mathrm{d}x$$

which can be written as $\mathbb{E}[g(X)(b-a)]$ for $X \sim \text{Unif}[0,1]$. So, to estimate I, we can:

1) Take a large sample X from the Uniform$[a,b]$ distribution
2) Apply $g(\cdot)$ to each element and multiply by $(b-a)$, then take the mean of the resulting values

For example, suppose we wish to evaluate

$$I_1 = \int_0^1 e^{-x^2/2} \, \mathrm{d}x$$

Note that I_1 doesn't even admit a closed-form solution, because $g(x) = e^{-x^2/2}$ does not have an elementary antiderivative. However, to obtain an approximation to I_1, we can:

1) Generate a large sample from the Unif[0, 1] distribution
2) Compute $e^{-x^2/2}$ for each x from our sample in step (1), and take the sample average of the resulting values

```
set.seed(100)       ## for reproducibility purposes
U <- runif(5000)    ## take a sample from the Unif[0, 1] distribution
mean(exp(-U^2 / 2))     ## take the mean of e^(-U^2 / 2)
```

```
[1] 0.8553212
```

We can compare this to the "analytic" answer obtained by `R`:

```
integrate(\(x){exp(-x^2 / 2)}, 0, 1)
```

```
0.8556244 with absolute error < 9.5e-15
```

! Question 4

Consider the integral

$$I_2 := \int_0^2 x^2 \, \mathrm{d}x$$

Though this is a relatively simple integral to compute analytically, let's obtain an approximation using Monte Carlo methods.

1) Generate a sample of size 5000 from the Unif[0, 2] distribution;
2) Use this sample to compute a Monte Carlo estimate for I_2.

As a hint:

$$I_2 = \int_0^2 x^2 \, \mathrm{d}x = \int_0^2 (2x^2) \cdot \frac{1}{2-0} \, \mathrm{d}x$$

Solution:

```
## replace this line with your code
X <- runif(5000, 0, 2)
mean(2 * X^2)
```

```
[1] 2.655008
```

If we like, we can compare this against the true answer:

```
integrate(\(x){x^2}, 0, 2)
```

```
2.666667 with absolute error < 3e-14
```

Answer Check:
There is no autograder for this question; your TA will manually check that your answers are correct.

We do not have to limit ourselves to using the Uniform distribution in Monte Carlo simulations!

! Question 5

Consider the integral

$$I_3 := \int_0^2 x\cos(x)\ \mathrm{d}x$$

Though this is a relatively simple integral to compute analytically, let's obtain an approximation using Monte Carlo methods.

1) Use your `rdist1()` function from Question 3 above to generate a sample of size 5000 from the distribution with density given by (1)

2) Use this sample to compute a Monte Carlo estimate for I_3.

As a hint:

$$I_3 = \int_0^2 x\cos(x)\ \mathrm{d}x = \int_0^2 [2\cos(x)]\cdot\frac{x}{2}\ \mathrm{d}x$$

Solution:

```
## replace this line with your code
set.seed(100)
X <- rdist1(5000)
mean(2 * cos(X))
```

```
[1] 0.3992176
```

If we like, we can compare this against the true answer:

```
integrate(\(x){x * cos(x)}, 0, 2)
```

```
0.402448 with absolute error < 8.2e-15
```

Answer Check:
There is no autograder for this question; your TA will manually check that your answers are correct.

Finally, allow me to point out that Monte Carlo simulations work for integrals other than expectations as well. For example, let us return to I_1 above. Note that, if $X \sim \mathcal{N}(0,1)$, then

$$I_1 := \int_0^1 e^{-x^2/2}\,\mathrm{d}x = \sqrt{2\pi} \cdot \int_0^1 \frac{1}{\sqrt{2\pi}} e^{-x^2/2}\,\mathrm{d}x = \sqrt{2\pi} \cdot \mathbb{P}(0 \leq X \leq 1)$$

This reveals an alternate way to estimate I_1:

1) Generate a large sample from the $\mathcal{N}(0,1)$ distribution
2) Compute the sample *proportion* of values that are between 0 and 1, and multiply the result by $\sqrt{2\pi}$.

```
set.seed(100)
X <- rnorm(10e5)
sqrt(2 * pi) * ifelse((X >= 0) & (X <= 1), 1, 0) %>% mean()
```

```
[1] 0.8567756
```

Caution

Note that, in this second method, we needed to take a much larger sample than using the first method.

Question 6

Consider the integral

$$I_4 := \int_{0.5}^{1.5} \frac{x}{2}\,\mathrm{d}x$$

Obtain a Monte Carlo estimate for I_4 by generating a large sample from the distribution with density given by (1), and computing the proportion of values between 0.5 and 1.5. Use your `rdist1()` function from Question 3!

Solution:

```
## replace this line with your code
set.seed(100)
X <- rdist1(5000)
ifelse((0.5 <= X) & (X <= 1.5), 1, 0) %>% mean()
```

```
[1] 0.5012
```

If we like, we can compare this against the true answer:

```
integrate(\(x){x/2}, 0.5, 1.5)
```

```
0.5 with absolute error < 5.6e-15
```

Answer Check:
There is no autograder for this question; your TA will manually check that your answers are correct.

Now, one of the hallmarks of Monte Carlo estimation is its use of *randomness*. This means that the estimates produced by Monte Carlo methods are themselves random, and should follow some distribution. Let's investigate this distribution!

! Question 7

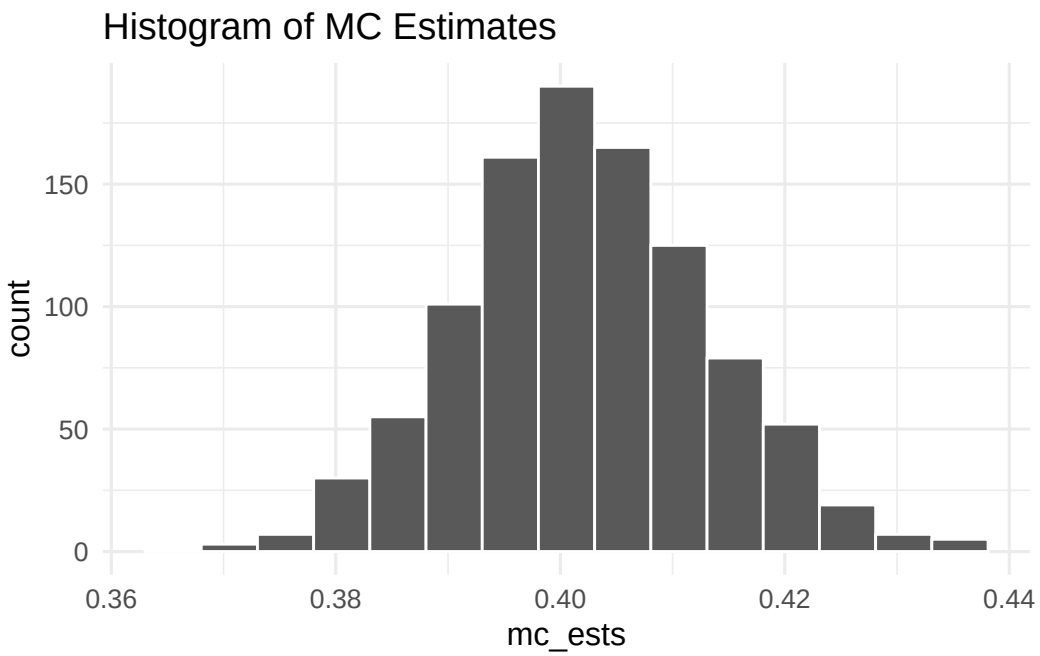
Return to I_3 above. Replicate your steps in Question 5 to obtain 1000 different Monte Carlo estimates of I_3 (each computed from a sample of size 5000 drawn from the distribution with density (1)). Plot the histogram of these estimates.

Solution:

```
## replace this line with your code
set.seed(100)
mc_ests <- c()
B <- 1000

for(b in 1:B) {
  mc_ests <- c(mc_ests,
               (rdist1(5000) %>% (\(x){2 * cos(x)})) %>% mean()
               )
}

data.frame(x = mc_ests) %>% ggplot(aes(x = mc_ests)) +
  geom_histogram(aes(x = mc_ests), bins = 15, col = "white") +
  theme_minimal(base_size = 12) +
  ggtitle("Histogram of MC Estimates")
```



Answer Check:
There is no autograder for this question; your TA will manually check that your answers are correct.

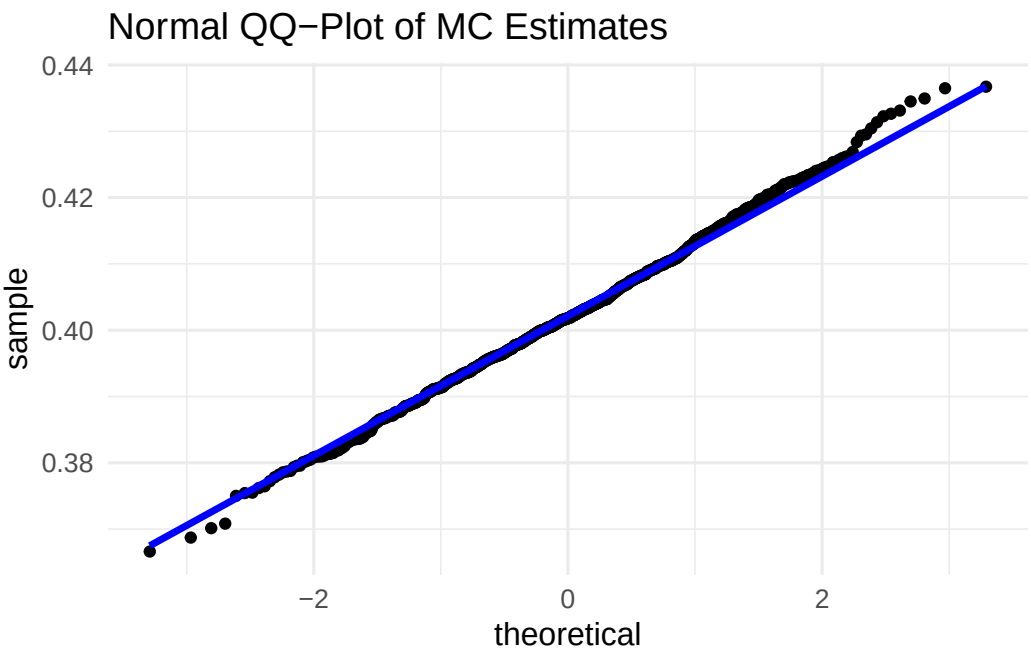
! Question 8

Based on your histogram from Question 7 above, what distribution do you think the Monte Carlo estimates follow? Confirm your answer using a QQ-plot.

Solution:
Replace this line with your answer

The histogram seems to indicate normality

```
## replace this line with your code
data.frame(x = mc_ests) %>% ggplot(aes(sample = mc_ests)) +
  geom_qq(distribution = stats::qnorm) +
  geom_qq_line(distribution = stats::qnorm, col = "blue", linewidth = 1.25) +
  xlab("theoretical") + ylab("sample") +
  theme_minimal(base_size = 12) +
  ggtitle("Normal QQ-Plot of MC Estimates")
```



The normality is, perhaps, unsurprising - Monte Carlo estimates are, in essence, sample means and the Central Limit Theorem ensures that sample means are approximately normally distributed.

Answer Check:
There is no autograder for this question; your TA will manually check that your answers are correct.

Submission Details

Congrats on finishing this PSTAT 100 lab! Please carry out the following steps:

Submission Details

1) Check that all of your tables, plots, and code outputs are rendering correctly in your final `.pdf`.
2) Check that you passed all of the test cases (on questions that have autograders). You'll know that you passed all tests for a particular problem when you get the message "All tests passed!".
3) Submit **ONLY** your `.pdf` to Gradescope. Make sure to **match ALL pages to the ONE question on Gradescope**; failure to do so will incur a penalty of 0.1 points.