unit_3_classwork

October 18, 2023

1 Unit 3 Classwork

The goals of this assignment are to help you (1) calculate expectations of random variables using R, (2) simulate probabilistic processes, and (3) estimate probabilities and other quantities through numerical (computer) simulation. Such simulations can be useful. We can use simulations to help confirm that we've calculated a probability "by hand" correctly, or to estimate other quantities, like areas/integrals!

1.1 Problem #1

1.(a) Write a function to calculate the expected value of a binomial random variable, using the definition of expectation. In other words, if $X \sim$ Bin(n, p), your function should take n and p as inputs, and return $E[X]$.

[ ]:
```
calculate_E <- function(n, p) {
  E <- n * p
  return(E)
}
```

1.(b) Store $m = 10,000$ binomial random numbers in a vector `x`. Each entry of `x` should represent the number of tails in 50 flips of a biased coin (where $P(\text{Heads}) = 0.75$).

[ ]:
```
m <- 10000
n <- 50
p <- 0.25

x <- rbinom(m, n, p)
```

1.(c) Calculate the (sample) mean of `x`. Then, use the formula for the mean of a binomial, and the function above, to calculuate the mean of the random variable X, where X counts the number of tails in 50 flips of a biased coin (again, $P(\text{Heads}) = 0.75$). What do you notice? Do these answers match?

[ ]:
```
Sample_Mean = mean(x)
Theory_Mean = calculate_E(n,p)

Sample_Mean
Theory_Mean
```

12.5066

12.5

The sample mean and the theoretical mean are very close to each other. However, they are usually not be exactly the same due to random sampling variability. The larger your sample size, the closer the values should be.

1.2 Problem #2

2.(a) First, write a function to calculate the expected value of an exponential random variable for an arbitrary λ. Then, use it to compute the expected value $E[X]$, where $X \sim$ Exp$(1/4)$. Hint: first make a function `integrand()` which takes a vector `x` and `lambda` as arguments, and then pass it to the `integrate()` function in `R`.

[ ]:
```
calculate_E_Exponential <- function(lambda) {
  expected_value <- 1 / lambda
  return(expected_value)
}

integrand <- function(x, lambda) {
  return(x * lambda * exp(-lambda * x))
}

lambda <- 1/4

E <- calculate_E_Exponential(lambda)
E_integrate <- integrate(integrand, lower = 0, upper = Inf, lambda = lambda)

E
E_integrate
```

4

```
4 with absolute error < 1.2e-05
```

In class, we also worked with the PDF $f(x) = \frac{3}{2}(1 - x^2)$ on $0 \leq x \leq 1$ (zero elsewhere).

2.(b) Find $E[X^3]$.

[ ]:
```
integrand <- function(x) {
  return(x * (3/2) * (1 - x^2))
}

expected_x3 <- integrate(integrand, lower = 0, upper = 1)

expected_x3
```

```
0.375 with absolute error < 4.2e-15
```

2.(c) Estimate $E[X^3]$ using Monte Carlo integration. How could you get a better estimate?

[ ]:
```
n <- 200000

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

mean((x * (3/2) * (1 - x^2)))
```

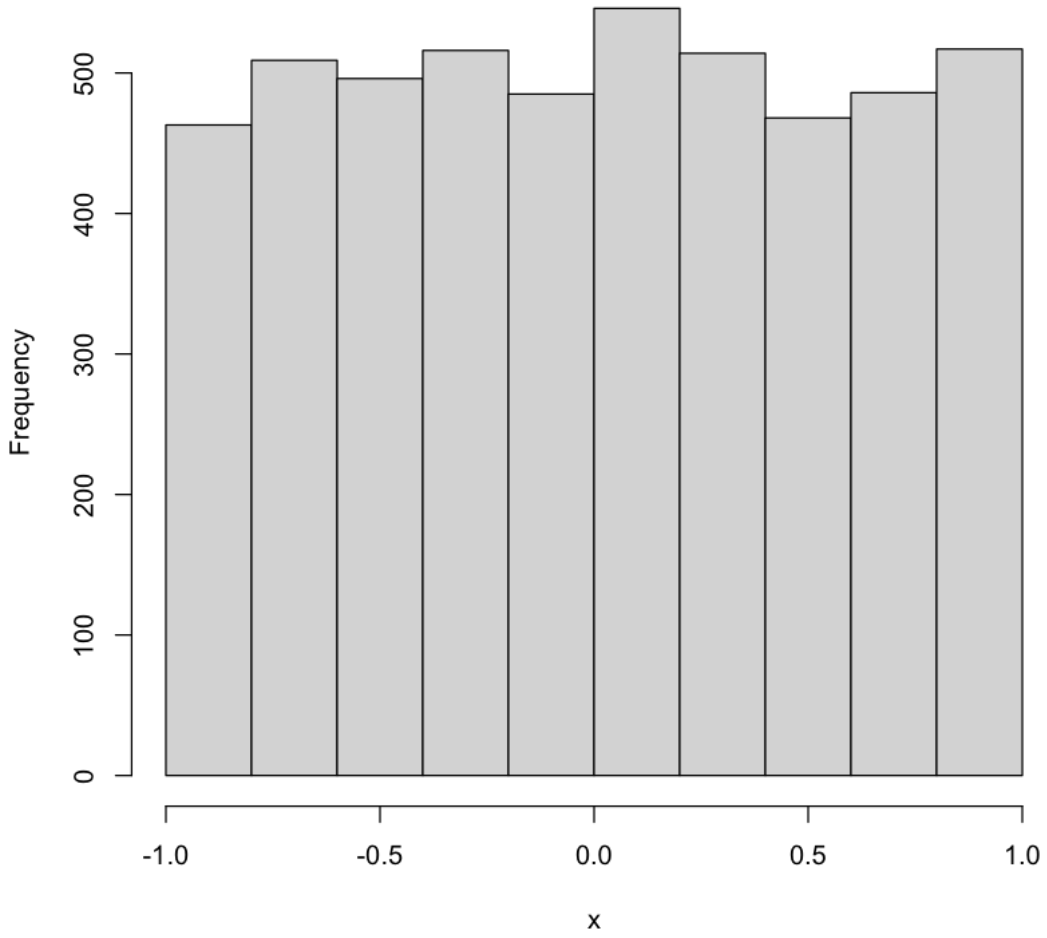
0.374750342925439

1.3 Problem #3

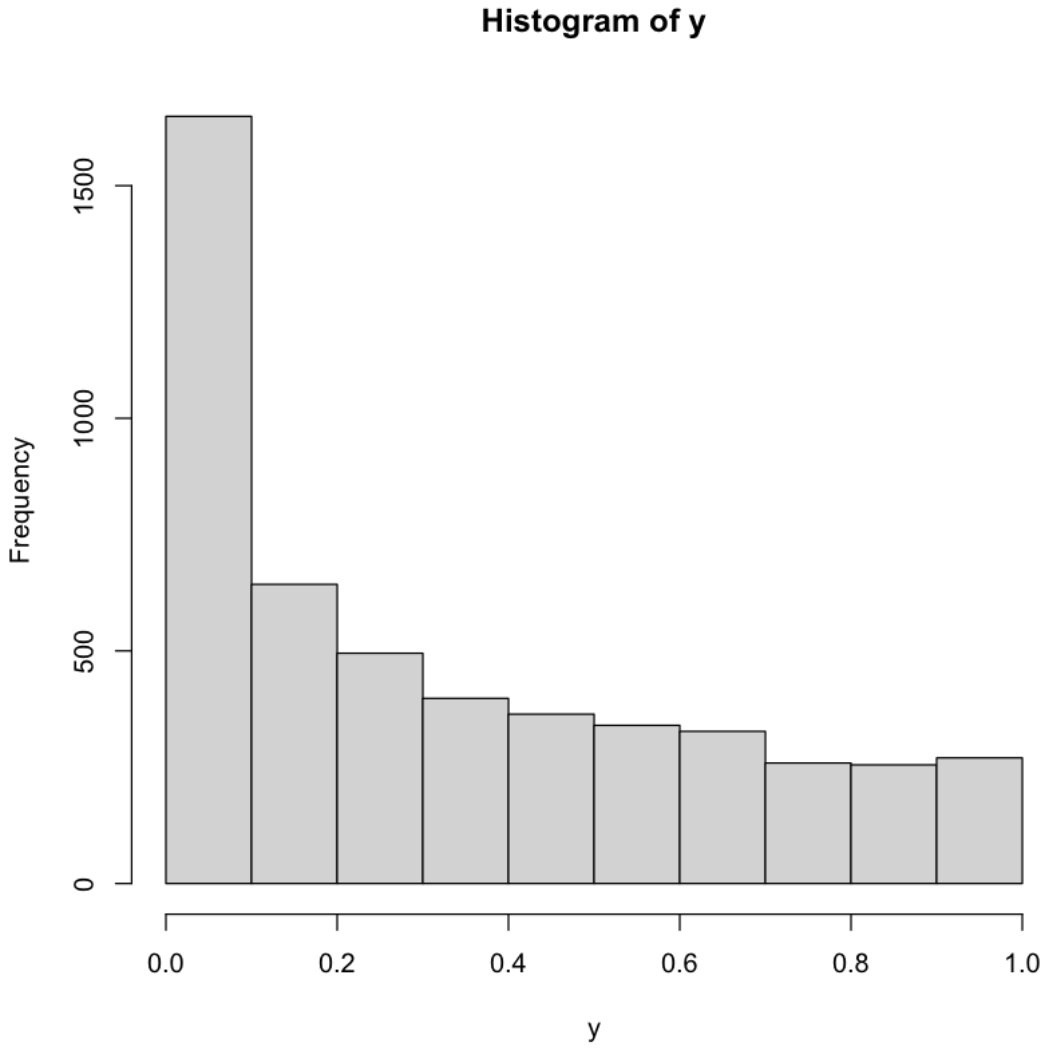
In class, we emphasized that variables can be uncorrelated but dependent. Let's look at an example of this.

3.(a) Generate $n = 5000$ random variables from a $U(-1, 1)$ distribution. Store these values in `x`. Then, create a variable `y` such that $y = x^2$. Plot `x` and `y` and compute the (sample) correlation coefficient using `cor()`. What do you notice?

[ ]:
```
n <- 5000
x <- runif(n, min = -1, max = 1)
y <- x^2

hist(x, main = "Histogram of x", xlab = "x")
hist(y, main = "Histogram of y", xlab = "y")

cor <- cor(x, y)

cor
```



0.0245658805992994



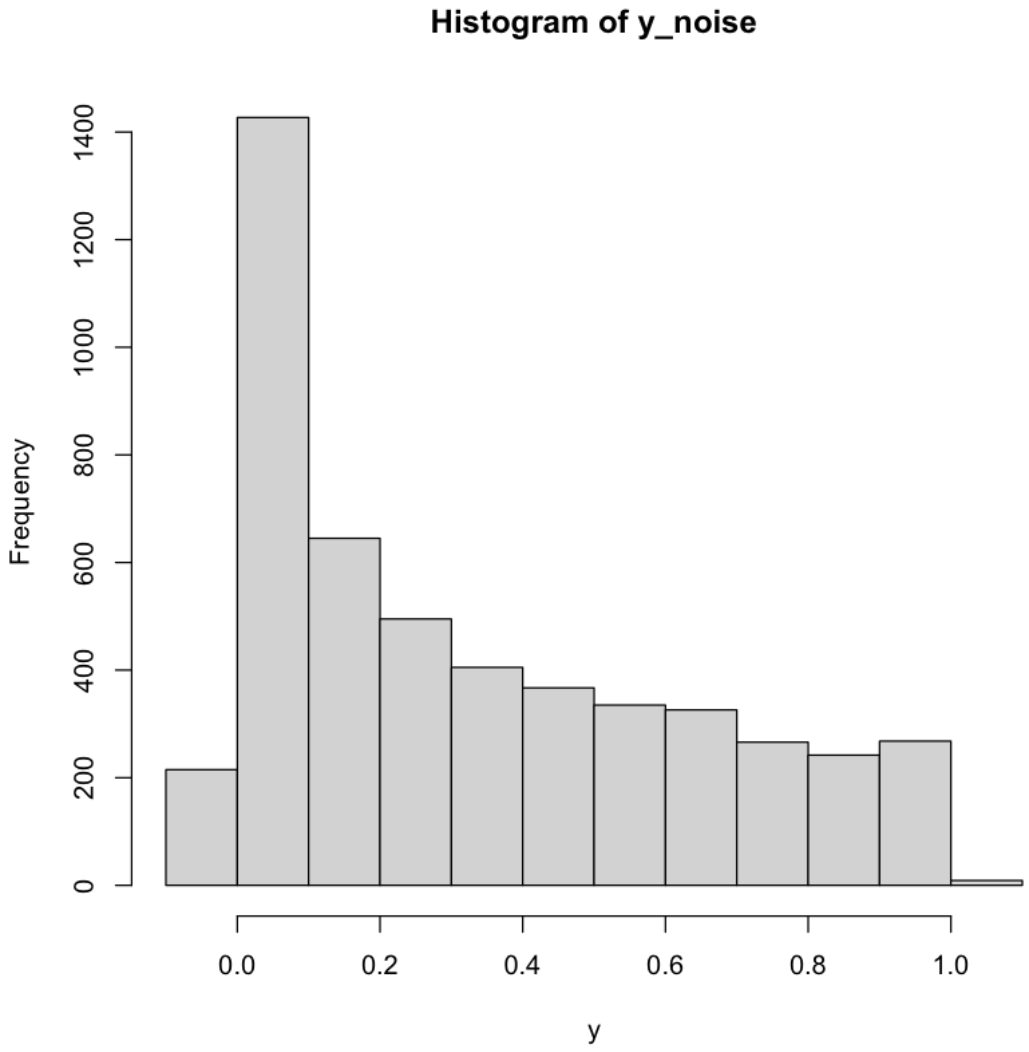
I noticed that the correlation coefficient between x and y is close to zero. This is because cor() function is trying to find linear correlations but the relationship is quadratic.

3.(b) Now add various degrees of "noise" to y, by adding random normal numbers to the y equation. Use `rnorm(n,0,b)`, where `b` is set to values like `0.01`, `0.1`, and `1`. What do you notice about the plot and corresponding correlation coefficient?

[ ]:
```
noise <- function(y, b) {
  n = length(y)
  noise <- rnorm(n, 0, b)
  y_with_noise <- y + noise
  return(y_with_noise)
}
```

```
b = 0.01
y_noise <- noise(y, b)

hist(y_noise, xlab = "y")

cor <- cor(x, y_noise)
cor
```

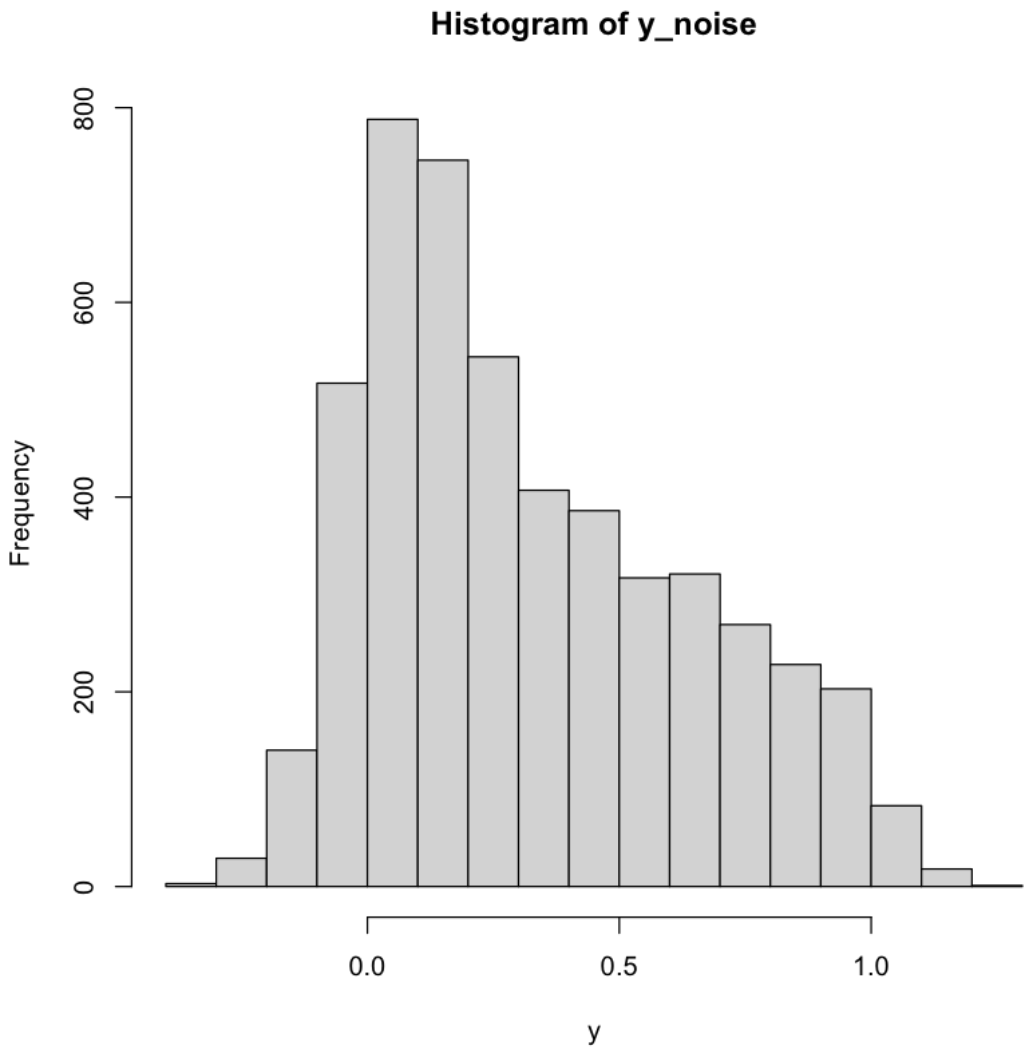
0.0256035609538797



[ ]:
```
b = 0.1
y_noise <- noise(y, b)
```

```
hist(y_noise, xlab = "y")

cor <- cor(x, y_noise)

cor
```

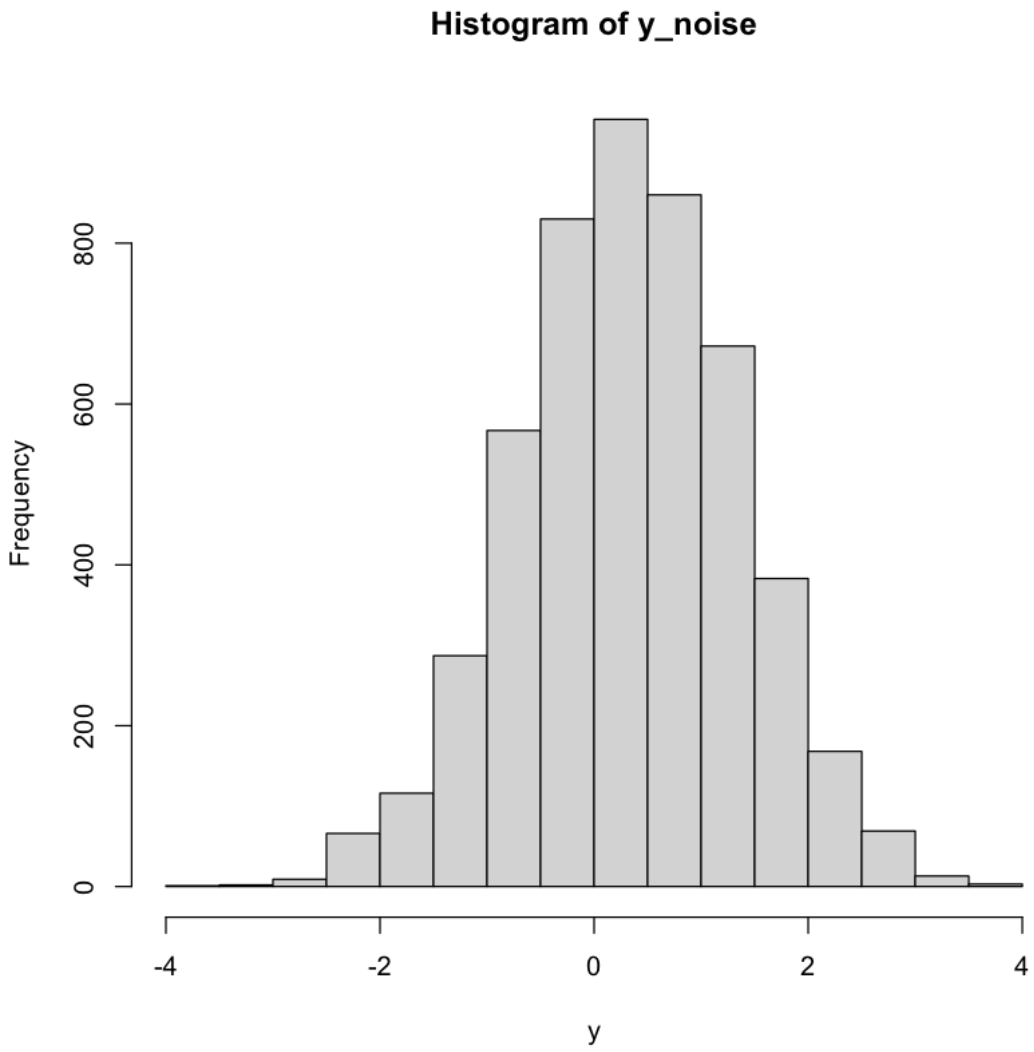
0.0196585797547282



[ ]:
```
b = 1
y_noise <- noise(y, b)

hist(y_noise, xlab = "y")
```

```
cor <- cor(x, y_noise)

cor
```

0.0083996811323634



As b (SD) increases, the noise makes the relationship between x and y less clear. This causes a decrease in the correlation coefficient