STAT 33B Homework 4

Ming Fong (3035619833)

Oct 22, 2020

This homework is due **Oct 22, 2020** by 11:59pm PT.

Homeworks are graded for correctness.

As you work, write your answers in this notebook. Answer questions with complete sentences, and put code in code chunks. You can make as many new code chunks as you like.

Please do not delete the exercises already in this notebook, because it may interfere with our grading tools.

You need to submit your work in two places:

- Submit this Rmd file with your edits on bCourses.
- Knit and submit the generated PDF file on Gradescope.

If you have any last-minute trouble knitting, **DON'T PANIC**. Submit your Rmd file on time and follow up in office hours or on Piazza to sort out the PDF.

Rejection Sampling

In Homework 1, you saw that R provides functions to sample from a variety of well-known distributions. For instance, `rbinom()` samples from a binomial distribution, and `runif()` samples from a (continuous) uniform distribution.

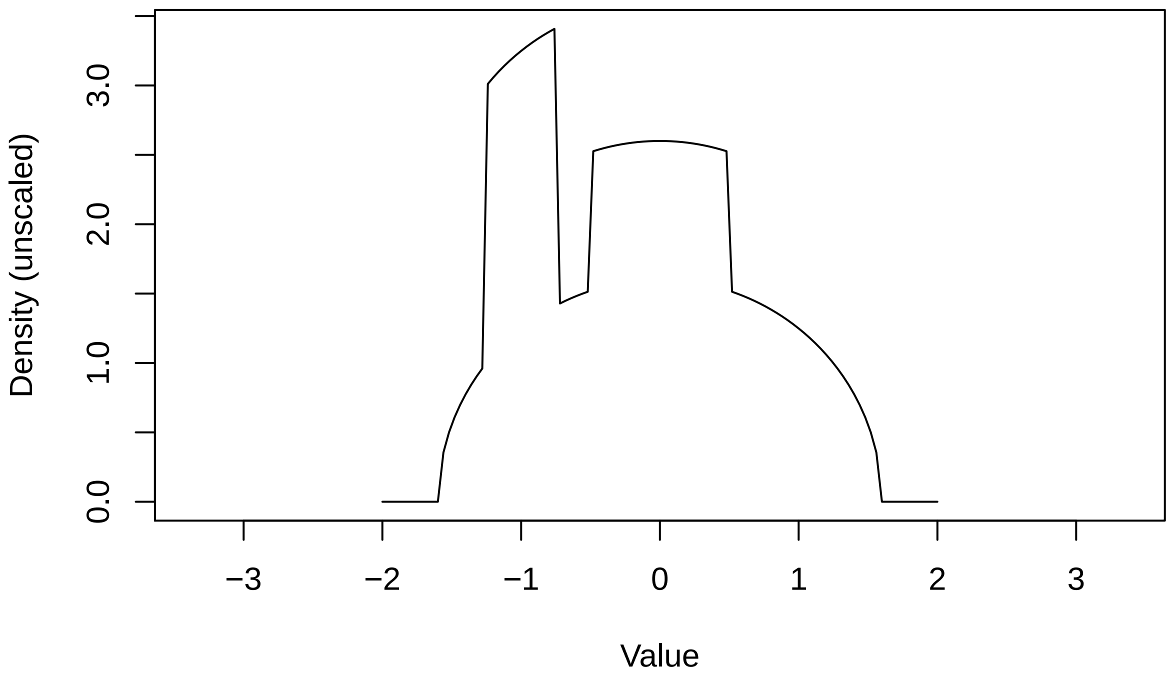
What if you want to sample from a distribution that's not well-known?

For example, suppose you want to sample from this distribution on -1.6 to 1.6:

```r
dhand = function(x) {
  y = numeric(length(x))
  i = -1.6 < x & x < 1.6
  y[i] = sqrt(2.56 - x[i]^2) + dunif(x[i], -1.25, -0.75) +
    dunif(x[i], -0.5, 0.5)

  y
}

curve(dhand, -2, 2, xlab = "Value", ylab = "Density (unscaled)", asp = 1)
```



Let's call this distribution the "hand" distribution, since it resembles the silhoutte of a person (or robot) raising their hand.

One way to sample from distributions that are not well-known by using a statistical technique called *rejection sampling.*

The idea is to choose a rectangle that completely encloses the target density curve, and then uniformly sample points within the rectangle. If a point falls below the density curve, then the point is accepted and its x-coordinate is a new sample value. If a point falls above the density curve, then it is rejected (and discarded). This produces the correct distribution because relatively more points will be accepted in places where the density curve is taller.

The bottom side of the enclosing rectangle should always be on the line `y = 0`.

The exact steps in rejection sampling are:

1. Uniformly sample (x, y) coordinates for a candidate point.
2. Test whether the y coordinate is below the target density curve. If it is, then the x coordinate is a new sample value. If it isn't, then the x coordinate is discarded.
3. Repeat steps 1-2 until the desired number of sample values is reached.

Exercise 1

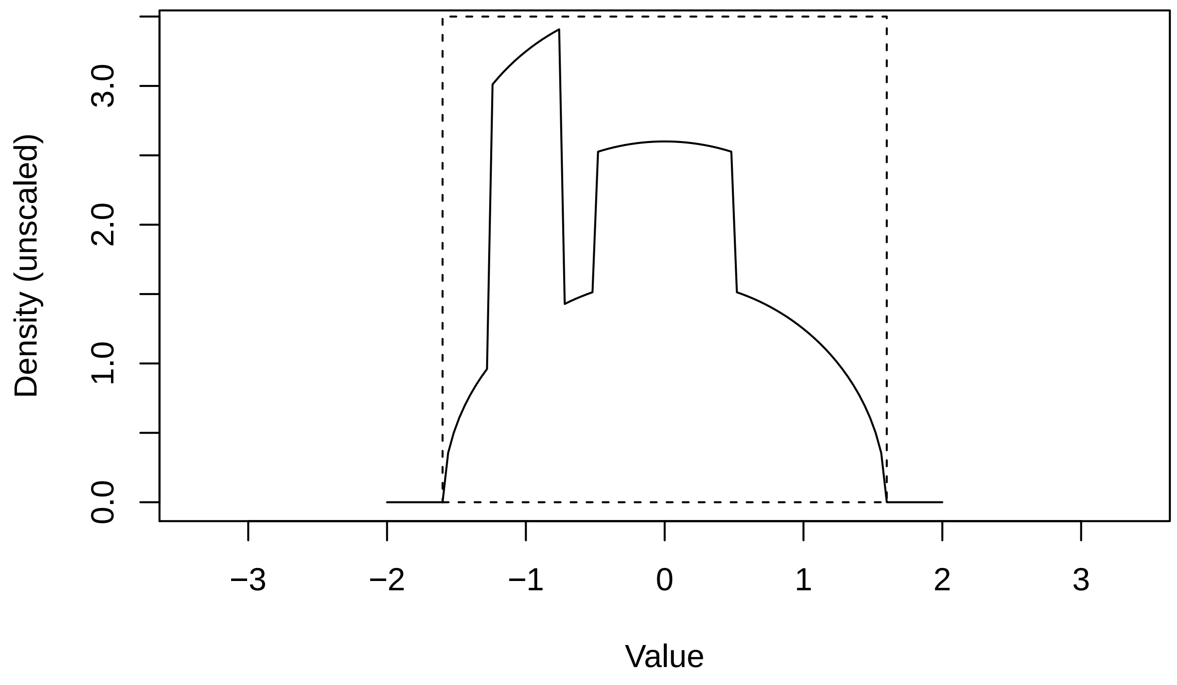
What's an appropriate enclosing rectangle for the hand distribution's density curve?

Use the call to `curve` above, followed by a call to `rect` (a base R plotting function), to plot the density curve with the enclosing rectangle superimposed. Use the argument `lty = "dashed"` in the call to `rect` so that the rectangle is visually distinct from the curve.

There are many possible answers to this question, but smaller rectangles are more efficient for rejection sampling.

YOUR ANSWER GOES HERE:

```r
curve(dhand, -2, 2, xlab = "Value", ylab = "Density (unscaled)", asp = 1)
rect(-1.6, 0, 1.6, 3.5, lty = "dashed")
```



Exercise 2

Write a function `rhand` that uses rejection sampling to return a vector of `n` samples from the hand distribution. Your function should have a parameter `n` with default argument `100`.

Use the `dhand()` function provided above as the target density curve.

Test that your function runs without error for `n` equal to 10, 100, and 1000.

YOUR ANSWER GOES HERE:

```r
rhand = function(n = 100) {
   samples = numeric(n)
   count = 0
   while (count < n) {
      x = runif(1, -1.6, 1.6)
      y = runif(1, 0, 3.5)
      if (y < dhand(x)) {
         samples[count + 1] = x
         count = count + 1
      }
   }

   samples
}
x = rhand(10)
x = rhand()
x = rhand(1000)
```

Exercise 3

1. Use your `rhand` function to sample 100 points from the hand distribution and plot the estimated density. Make sure to call `set.seed` first so that your result is reproducible.

   Hint: You can use base R graphics to plot an estimated density curve for a sample `x` with `plot(density(x), asp = 1, xlim = c(-3, 3))`.

2. How does the shape of your estimated density curve compare to the shape of the actual density curve for the hand distribution (see above)?

3. Repeat Parts 1-2 with a sample of 1,000,000 points. Comment on how the new estimated density curve compares to the one from Part 1.

   Note 1: Sampling this many points may take 10-60 seconds. Anything substantially longer means your function is doing something inefficient.

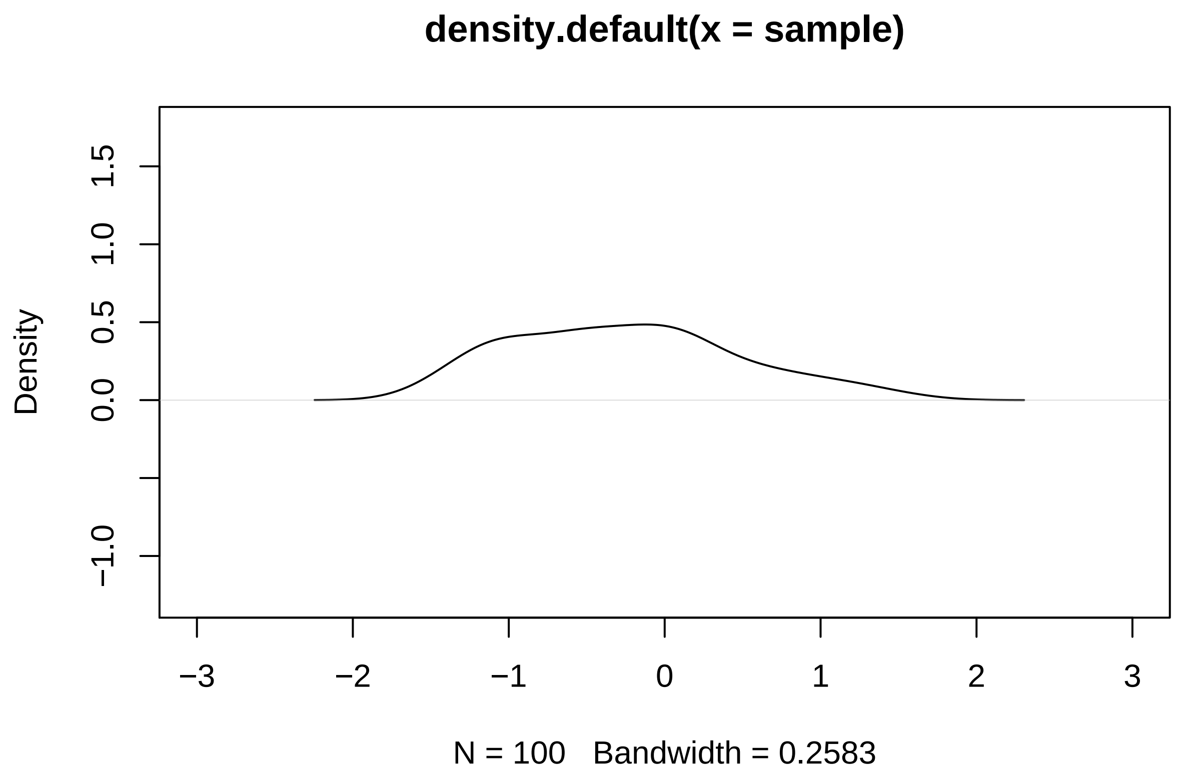
   Note 2: Your estimated density curve in this part should look flatter than the actual density curve, but have the same general shape. If it has a different shape, there may be a bug in your `rhand` function.

4. Based on the sample from Part 3, what are the approximate mean and standard deviation of the hand distribution?

YOUR ANSWER GOES HERE:

Part 1

```
set.seed(42)
sample = rhand()
plot(density(sample), asp = 1, xlim = c(-3, 3))
```



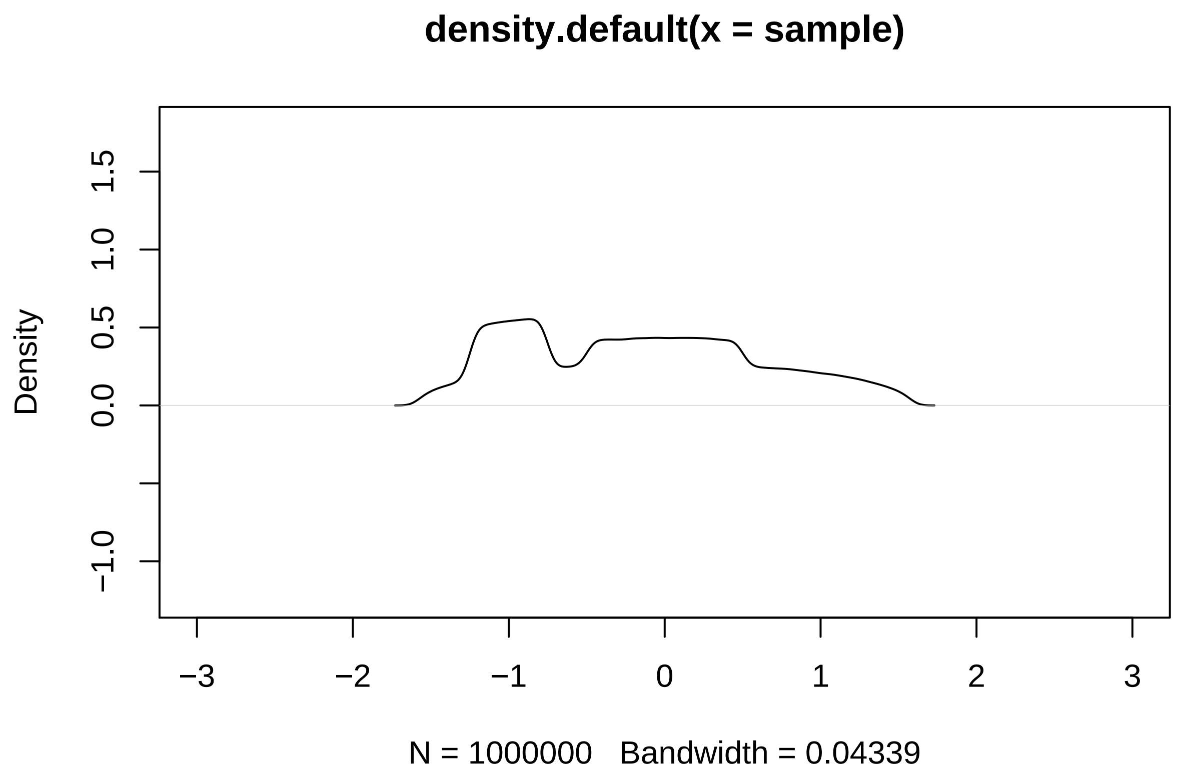
Part 2

My estimation looks like a blob while the original curve had clearly defined curves and edges.

Part 3

The new curve with 1,000,000 points has much more distinguishable features than the previous sample.

```r
sample = rhand(1000000)
plot(density(sample), asp = 1, xlim = c(-3, 3))
```



Part 4

```
mean(sample)
```

```
## [1] -0.1655812
```

```
sd(sample)
```

```
## [1] 0.7641279
```

Exercise 4

Create a new version of your `rhand` function called `rhand_rejected`. Instead of returning the sampled vaues, the `rhand_rejected` function should keep track of and return the number of points that were rejected.

What percentage of points is typically rejected for your rejection sampler? Use an average across multiple runs to get a more accurate estimate.

Hint: The `replicate` function may be helpful for calling `rhand_rejected` repeatedly.

YOUR ANSWER GOES HERE:

About 53% of the points are typically rejected.

```
rhand_rejected = function(n = 100) {
   count = 0
   iterations = 0
   while (count < n) {
```

```r
      iterations = iterations + 1
      x = runif(1, -1.6, 1.6)
      y = runif(1, 0, 3.5)
      if (y > dhand(x)) {
         count = count + 1
      }
   }

   iterations - n
}
n = 1000
mean(replicate(100, 1 - n / (n + rhand_rejected(n))))
```

```
## [1] 0.5373334
```