

# Chapter 1 - Random Number Generation

## Conditional Distributions

---

Prof. Alex Alvarez, Ali Raisolsadat

School of Mathematical and Computational Sciences  
University of Prince Edward Island

**Conditional Distribution:** Suppose that we start with a random variable  $X$  with a distribution  $P_X$  that we know how to sample from. Let  $A$  be an event and consider the conditional distribution of  $X$  given  $A$   $P_{X|X \in A}$ .

We will be able to sample from the conditional distribution  $P_{X|X \in A}$  by using a version of the rejection sampling algorithm as follows:

## Algorithm

1. Generate  $X \sim P_X$  (proposal)
2. If  $X \in A$  then  $Y = X$  (proposal is accepted)
3. If  $X \notin A$  then return to step 1
4. Return  $Y$

# Conditional Distributions Example

**Example** Generate a random sample of size  $n = 1000$  from the conditional distribution of  $X \sim N(0, 1)$  conditioned on  $X \geq 0$

## Code

### R Code

```
1  n <- 1000
2  counter <- 1
3
4  target_sample <- vector();
5  while (counter < n+1) {
6    proposal <- rnorm(1)
7
8    if (proposal > 0){
9      target_sample[counter] <-
        proposal
10     counter <- counter+1
11   }
12 }
13 hist(target_sample)
```

### Python Code:

```
1  import numpy as np
2  import matplotlib.pyplot as plt
3
4  n = 1000
5  counter = 0
6  target_sample = []
7  mu, sigma = 0, 1
8
9  while counter < n:
10     proposal = np.random.normal(loc=
        mu, scale=sigma)
11     if proposal > 0:
12         target_sample.append(
            proposal)
13         counter += 1
14
15  plt.hist(target_sample, bins=30)
16  plt.show()
```

# Conditional Distributions Example (Vectorized)

**Example:** Generate a random sample of size  $n = 1000$  from the conditional distribution of  $X \sim N(0, 1)$  conditioned on  $X \geq 0$

## Code

### R Code

```
1  n <- 1000
2  mu <- 0
3  sigma <- 1
4  samples <-
5      rnorm(n * 2, mean = mu, sd =
6          sigma)
7  target_sample <-
8      samples[samples > 0]
9  target_sample <-
10     target_sample[1:n]
11 hist(target_sample, breaks = 30)
```

### Python Code

```
1  import numpy as np
2  import matplotlib.pyplot as plt
3
4  n = 1000
5  mu, sigma = 0, 1
6  samples =
7      np.random.normal(loc=mu, scale=
8          sigma, size=n*2)
9  target_sample =
10     samples[samples > 0]
11 target_sample = target_sample[:n]
12 plt.hist(target_sample, bins=30)
13 plt.show()
```

## Remarks

- The described method to generate samples from conditional distributions is very straightforward as long as we know how to sample from the unconditional distribution of  $X$ .
- The proposals will be accepted with probability  $P(A)$  so the efficiency of the method may not be good if  $A$  is an event with very low probability.
- In cases where  $P(A)$  is small, this method might not be advisable. Example 1.28 from the textbook covers one of such examples and provides an alternative solution.

## Rejection Sampling in Arbitrary Spaces

One of the advantages of rejection sampling is that it can be used to generate random objects in more general spaces, in particular random vectors.

A standard use of the rejection sampling algorithm is related to the generation of uniformly distributed random vectors over some arbitrary subsets of  $\mathbb{R}^d$ .

For simplicity we will focus on the case  $d = 2$  (which can be visualized easily) but these ideas also apply in higher dimensions.

## Uniform sampling from a rectangle

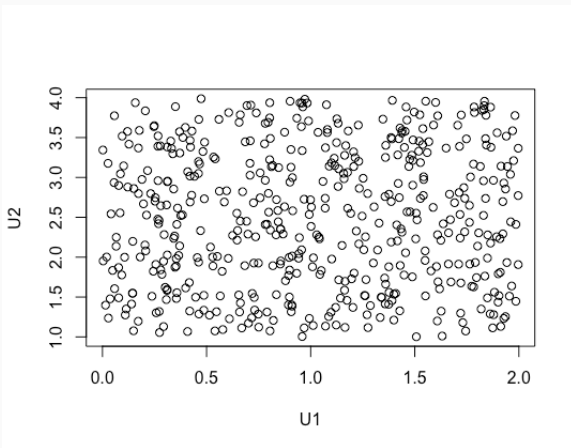
If we want to generate a random vector with uniform distribution over the rectangle  $[a, b] \times [c, d]$  we can do so directly by generating its two components independently one from the other (each of them uniformly distributed) as follows:

### Algorithm:

1. Generate  $X \sim U[a, b]$
2. Generate  $Y \sim U[c, d]$  (independently of  $X$ )
3. Return vector  $(X, Y)$

# Uniform sampling from a rectangle

Plot of 500 uniformly distributed random points on  $[0, 2] \times [1, 4]$ .





## Main result

**Lemma 1.31 from textbook:** Let  $X$  be uniformly distributed on a set  $A$ , and let  $B$  be a set such that  $|A \cap B| > 0$ . Then the conditional distribution  $P_{X|X \in B}$  of  $X$  conditioned on the event  $X \in B$  coincides with the uniform distribution on  $A \cap B$ .

**Remark:** The symbol  $|Y|$  refers to the “volume” (or measure) of set  $Y$ . For instance, In the case of two dimensions we need that the area of  $A \cap B$  is strictly positive.

- The previous Lemma indicates a possible approach (using rejection sampling) to generate random vectors uniformly distributed in some “irregular” subset  $B$  of  $\mathbb{R}^2$  (and  $\mathbb{R}^d$  in general).
- First we would need to start with a set  $A \supset B$  so that we can sample uniformly distributed random vectors from  $A$ . Then we will reject all the vectors that do not belong to  $B$  and keep the generated random vectors in  $B$ .
- According to the previous Lemma, this sample will be uniformly distributed on  $A \cap B = B$ .
- The easiest approach (but not strictly necessary or possible) consists on selecting  $A$  with a rectangular shape, as we already know how to generate random vectors from a rectangle.

**Example:** Generate a sample of uniformly distributed random vectors on the semicircle defined by  $x^2 + y^2 \leq 1$  and  $y \geq 0$ .

**Example:** Generate a sample of uniformly distributed random vectors on the semicircle defined by  $x^2 + y^2 \leq 1$  and  $y \geq 0$ .

**Solution:** We will use the algorithm described earlier with  $B$  being the described semicircle and  $A$  being the rectangle  $[-1, 1] \times [0, 1]$  which includes  $B$ .

Essentially we will start generating uniformly distributed random numbers in the rectangle  $A$  and out of those, we will reject the ones that are not in  $B$ . Notice that the theoretical probability that a proposal point is accepted is  $\pi/4 \approx 0.785$ .

# Algorithm: Rejection Sampling Inside Unit Semicircle

---

**Algorithm 1** Generate points  $(X_1, X_2)$  uniformly inside the semicircle

---

```
1: Input: Integer  $n$ 
2: Initialize empty vectors  $X_1$  and  $X_2$ 
3: Generate uniform random vectors:
    $U_1 \sim \text{Uniform}(-1, 1)^n, U_2 \sim \text{Uniform}(0, 1)^n$ 
4: Initialize counter:  $\text{counter} \leftarrow 1$ 
5: for  $i = 1$  to  $n$  do
6:   if  $U_1[i]^2 + U_2[i]^2 < 1$  then
7:      $X_1[\text{counter}] \leftarrow U_1[i]$ 
8:      $X_2[\text{counter}] \leftarrow U_2[i]$ 
9:      $\text{counter} \leftarrow \text{counter} + 1$ 
10:  end if
11: end for
12: Output:  $X_1, X_2$  (accepted points inside the semicircle)
```

---

# Algorithm: Vectorized Rejection Sampling Inside Unit Semicircle

---

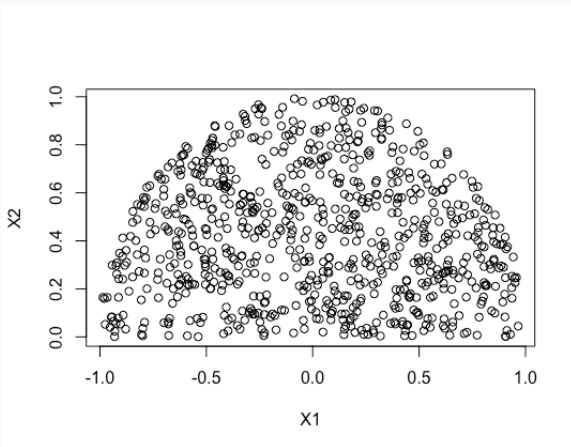
**Algorithm 2** Generate  $n$  points  $(X_1, X_2)$  uniformly inside the semicircle (vectorized)

---

- 1: **Input:** Integer  $n$
  - 2: Generate candidate points:  
     $U_1 \leftarrow \text{runif}(n \times 2, -1, 1)$   
     $U_2 \leftarrow \text{runif}(n \times 2, 0, 1)$
  - 3: Compute mask for points inside the unit circle:  
     $\text{inside} \leftarrow (U_1^2 + U_2^2 < 1)$
  - 4: Keep only the accepted points:  
     $X_1 \leftarrow U_1[\text{inside}]; X_1 \leftarrow X_1[1 : n]$   
     $X_2 \leftarrow U_2[\text{inside}]; X_2 \leftarrow X_2[1 : n]$
  - 5: **Output:**  $X_1, X_2$  (vectors of  $n$  points inside the semicircle)
- 

**Note:** In vectorized rejection sampling, we often generate more candidate points than needed (e.g.,  $n \times 2$ ) to ensure that enough points satisfy the acceptance condition. Since only a fraction of the candidates fall inside the desired region, oversampling increases the likelihood that we can select exactly  $n$  accepted points without looping. The factor 2 is a simple heuristic; larger factors may be needed if the acceptance rate is low.

The code gave us 780 generated uniformly distributed random points on the semicircle  $x^2 + y^2 \leq 1, y \geq 0$ .



- As the previous example shows, this method can be very useful to generate samples that follow the uniform distribution over some irregular sets of  $\mathbb{R}^d$ .
- If the set  $B$  is small compared to the set  $A$  then the method may be inefficient.
- It is better to use this method if we have a relatively easy way to check whether a given point belongs to set  $B$ . That is easy for a semicircle(previous example) but not so easy if  $B$  is a pentagram(a five-pointed star).
- Hard (but not impossible) to use this method if the set  $B$  is unbounded, as we won't be able to enclose it in a rectangle  $A$ .



1. Write code for the uniform sampling from a rectangle algorithm, with  $X \sim U[0, 2]$  and  $Y \sim U[1, 4]$ .
2. Implement code for Algorithms 1 and 2 using the rectangle  $[-1, 1] \times [0, 1]$ . If computationally feasible, run both algorithms for  $num\_sims = [50, 100, 500, 1000, 2000]$  and plot the difference in their running times.
  - **Hint:** Use built-in timers to measure running time:
    - In Python: `import time; start = time.time(); ...; end = time.time()`
    - In R: `system.time({ ... })`
3. Write a computer program to generate a random sample of size 1000 from the conditional distribution of  $X \sim Binomial(n, p)$  (with  $n = 10$  and probability of success  $p = 0.6$ ) conditioned on  $X \geq 5$ .
4. Write a computer program to generate (and plot) a sample of 500 uniformly distributed random points on the set of plane given by  $y \geq 0$ ,  $-\pi/2 \leq x \leq \pi/2$ , and  $y \leq \cos x$ .