CS4830 System Simulation Final Project

Nick Werle

December 16, 2017

Contents

1 Introduction

The year is 2075, and robots mining on mars is now a thing. MineCo, the biggest player in the Mars based robot mining game, has decided that to answer business questions, they should be using simulations rather than their current methodology of just guessing. This document will enumerate the simulation methodology, some of the models used in the simulation, and then 3 questions and the answers that the simulation produced.

2 Discussion

2.1 Simulation Background

This simulation has a few basic components. First, there is one mining base. The mining base has chargers that can be used by any robot, but no more than any one robot. Next, there are mining sites, which can only be mined by one robot at a time. Finally, there are robots, that do the mining. Robots have a speed that is used to determine how long it takes to get to the mining site or back to the base. Robots have an unloading time, and they can have different digging implements that mine at different speeds. Additionally, robot batteries have variable qualities, which affect how well they charge, and the power consumed is a function of amount of ore being carried.

2.2 Simulation Methodology

The simulation follows a simple structure. There is a SimulationDriver that is responsible for running the simulation. After each event, it runs a callback to determine if the simulation should continue running. In the case of all three scenarios presented today, that condition will be a check to see if at least one year has elapsed. The simulation is an event driven simulation, and the global clock is present in the State object. Prior to the execution of each event, the time jumps to the time of the next event of interest. Each event has code that is executed that manipulates the state in some way - for example, the charging complete event marks a robot's battery as being fully recharged. The documentation on what each event type can be found in the source code section of this document, above the run method for each event.

One thing that I opted to do, for simplicity, is to assume that the robots are always able to complete a round trip to and from a mining site - after all, why would you include one in the pool if you could not[1]?

Additionally, a limitation of this simulation is that it actually requires a mining site per miner, as I did not implement queue logic to wait for a mining site to become available. Furthermore, another limitation is that ideally the robots would be able to head towards a mining site that is very close to becoming available. This isn't possible in the simulation, and even in reality, the best you could do is a guess because you can't predict exactly when a miner will finally collect a full load of ore, so it didn't strike me as being particularly worthwhile[2].

2.3 Simulation Models

This simulation employs many models, both black and white box. In this section I will describe the models, state whether they are black or white box, where appropriate identifying them as coming from a PMF or a PDF, and provide justification for my statement. Finally, for ease of grading, I have denoted what grading criteria I believe each model should count towards using **boldface** text where I make the claim.

2.3.1 Robot Speed

Each robot has a value for their speed, which is assigned at the start of the simulation. To avoid being needlessly trivial, and to reflect variation in manufacturing quality, the speed is generated from a Gaussian distribution. I believe **this is a black box model** because it does not actually attempt to explain why the

[1]You might want to if you were interested in modeling a scenario where there are robots with mixed capabilities/max travel ranges, but that seemed unnecessary for this assignment.

[2]Perhaps the robots get to and from the mining sites via rail, and that's why they aren't available until the current robot is done working...

speed is what it is - in fact, it isn't even really stated what the mechanism for locomotion is - the thing could be hovering for all this simulation cares.

Please see the verification section for code validating that my standard Gaussian distribution implementation faithfully generates normal pseudo-random values.

2.3.2 Load Factor

The load factor, as previously discussed, is the value which determines how much extra power is consumed when traveling with a particular amount of ore. Rather than impacting travel speed, I instead chose to make it impact battery consumption, and thus charge time. **This model is a black box**, because it is just the implementation of a regression equation developed by the engineers over at MineCo. While it is a function of ore, and in that sense *could* be argued to be a white box, it fails to truly explain the underlying mechanics of why more power is consumed, such as if it causes the motors driving the wheels to work harder, or maybe the force field holding all the collected ore needs more juice to hold the extra weight. You can see the load factor implemented in the Robot class' `GetLoadFactor()` method, and you can see it being called to modify power consumed in the AtBaseEvent class. I'm not sure how I would validate this - the model is a very simple equation, and is pretty much verifiable with a cursory review of the code.

2.3.3 Battery Quality

The quality of the battery affects how quickly the battery charges. An ideal battery will have a quality of 100 or greater, as such a battery would have no impact on charge time, or even better, could slightly reduce charge time.

To implement battery quality, the friendly engineers at MineCo did not have a regression function for me to use this time, but they do have data from past tests of battery quality from similar robots - you can see a histogram in Figure 1. I opted to perform **rejection sampling against a PMF** created from the data they provided. To do that, I precomputed frequency of the different quality values, as in Figure 2, and then I load those values in at runtime, determine the maximum frequency value, then produce a PMF by dividing each frequency by the number of observations.

From there, generating pseudo-random values is just a matter of following standard rejection sampling procedures, generating uniform numbers and checking if they fit inside the area under the distribution's curve. All of this code can be seen in

2.3.4 Unload Time

When a robot arrives at the base station, it first must unload the ore collected. Rather than just using a constant, I opted to use a use a **PDF to generate times** - in this case, the PDF for a normal curve. The result is a black box model, because it doesn't really explain why it takes an average of 10 seconds to unload, it simply provides a time that the system accepts. To generate the numbers, I used a Box-Muller Transform, the code for which is in my ThreadSafeRandom class, and you can find the verification of that class (and by extension this model) in the verification section.

2.3.5 Travel Time

Travel time is one of my simulation's **white box models**. It is clearly a white box model because it uses the underlying variables to determine how long it will take to arrive at a particular destination - speed and distance. You can see the implementation of this model in the Robot class. I don't believe any verification is required, because of the simple nature of the formula driving the model.

2.3.6 Mining Time

Mining time is my final **white box model**. Mining time, which you could think of as the time to collect one unit of ore, is determined using a normal distribution. Ordinarily this would fall under a black box model, but I believe it is white box in the context of the simulation as a whole, because the mining time can be tweaked to examine how different digging implements would affect time. The code for this can be found in

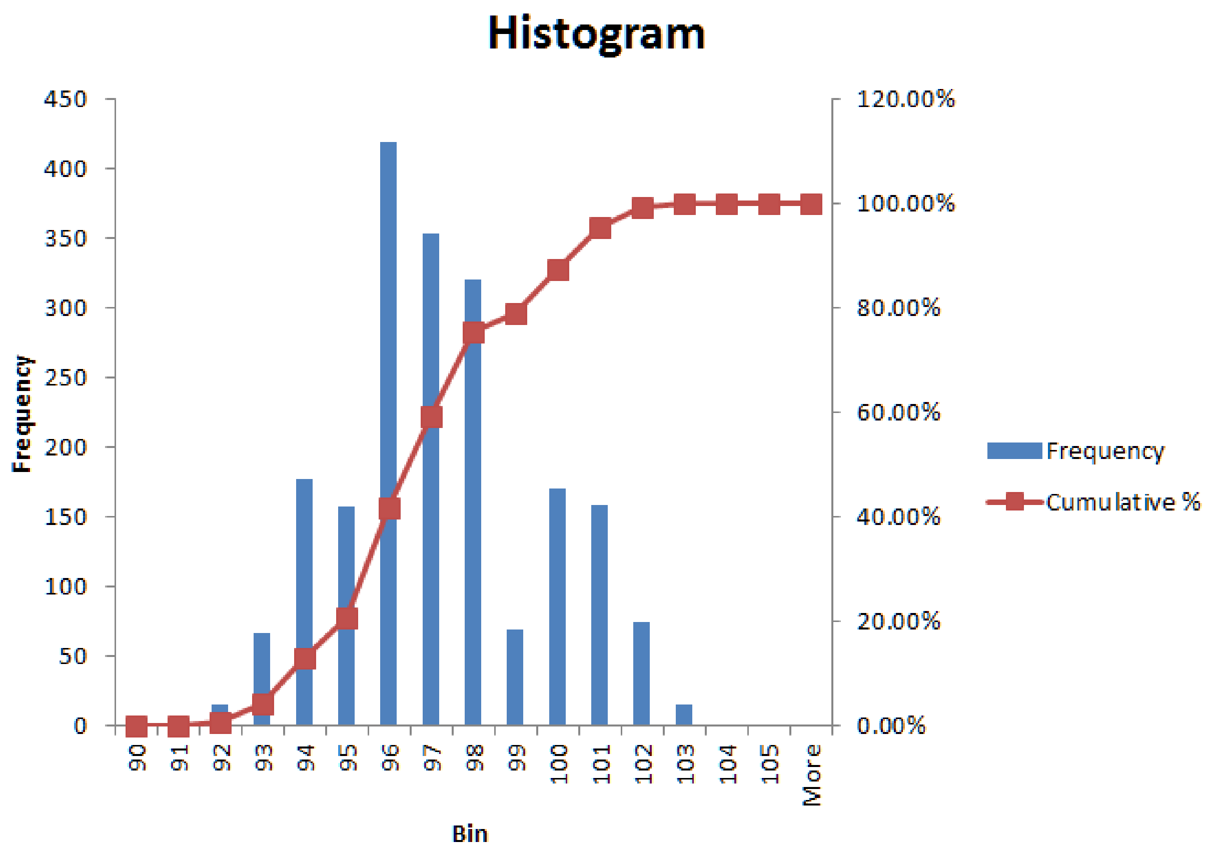
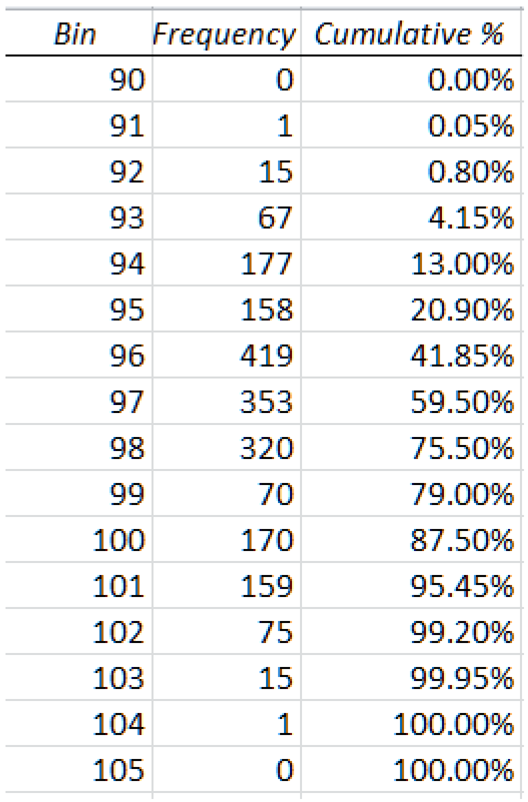


Figure 1: Histogram of provided data

the Robot class, and because it uses the same Box-Muller transform as before, I encourage you to see the verification section to verify this model.



Bin	*Frequency*	*Cumulative %*
90	0	0.00%
91	1	0.05%
92	15	0.80%
93	67	4.15%
94	177	13.00%
95	158	20.90%
96	419	41.85%
97	353	59.50%
98	320	75.50%
99	70	79.00%
100	170	87.50%
101	159	95.45%
102	75	99.20%
103	15	99.95%
104	1	100.00%
105	0	100.00%

Figure 2: Frequency histogram of battery qualities

3 Questions

I asked 3 questions with my simulation:

1. What is the average amount of ore that can be mined in one year?
2. What if instead of using the standard mining set from question 1, we used slow moving, quick digging robots?
3. Would halving the number of available charging stations adversely effect production?

To answer these questions, I first defined a baseline scenario:

- 4 robots that can carry 400 units of ore and move an average of 15 ft/second
- 4 mining sites, located 100, 110, 80, and 100 ft away from the camp
- 4 charging stations available

3.1 Question 1

I thought that this would be an important question to ask out the gate, because it gives a good idea of the scale of the operation being run, and it provides a baseline to compare the other questions to. It would also be good for MineCo because it would allow them to get at least a rough idea of what sort of income they might expect to generate.

To answer this, I first set up my simulation according to the baseline, and then told it to run a 1 year simulation 10,000 times. Doing so produced the datafile "scen1.txt" located in the data folder. The average value was 17,132,063 units of ore, and using excel I produce a 95% confidence interval of ± 3,242 units. I would show these results, but all I did was open the text file in excel and then use the AVERAGE and CONFIDENCE functions, so it didn't make for a particularly interesting screenshot/data table.

3.2 Question 2

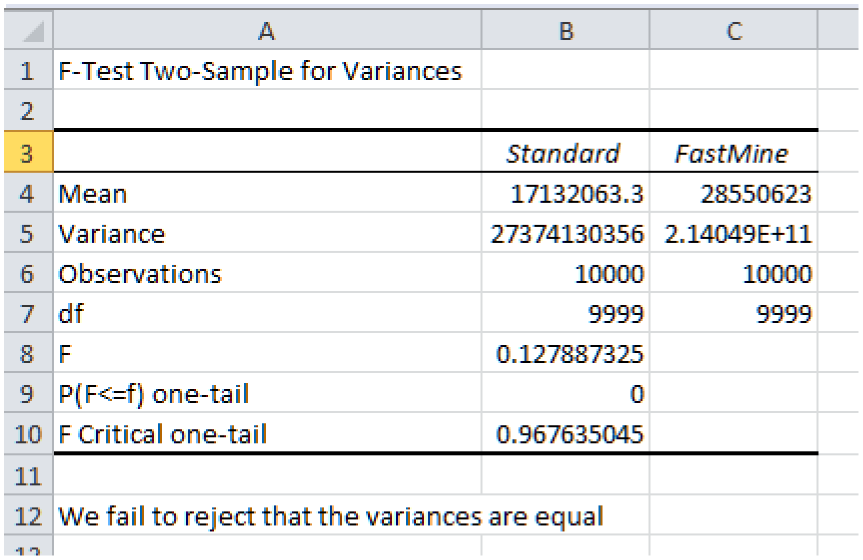
I thought this was a particularly interesting question to ask too. In this scenario, MineCo is considering using a different type of robot - one that moves much, much more slowly, but can extract ore very quickly.

To answer this, I set up my simulation similarly to the baseline, but I reduced the average speed from 15ft/s down to 5ft/s, and then decreased average mining time and standard deviation from 7 and 3 down to 4 and 1, respectively, and again ran 10,000 one year simulations. I wasn't sure if they would be faster or slower - so I just made my null hypothesis that they were equal, and unsurprisingly they aren't - I was however, really surprised at the magnitude of the increase. In hindsight the substantial jump makes sense - most of the time is spent mining, so it's unsurprising that this would have a massive impact on ore mined.

3.3 Question 3

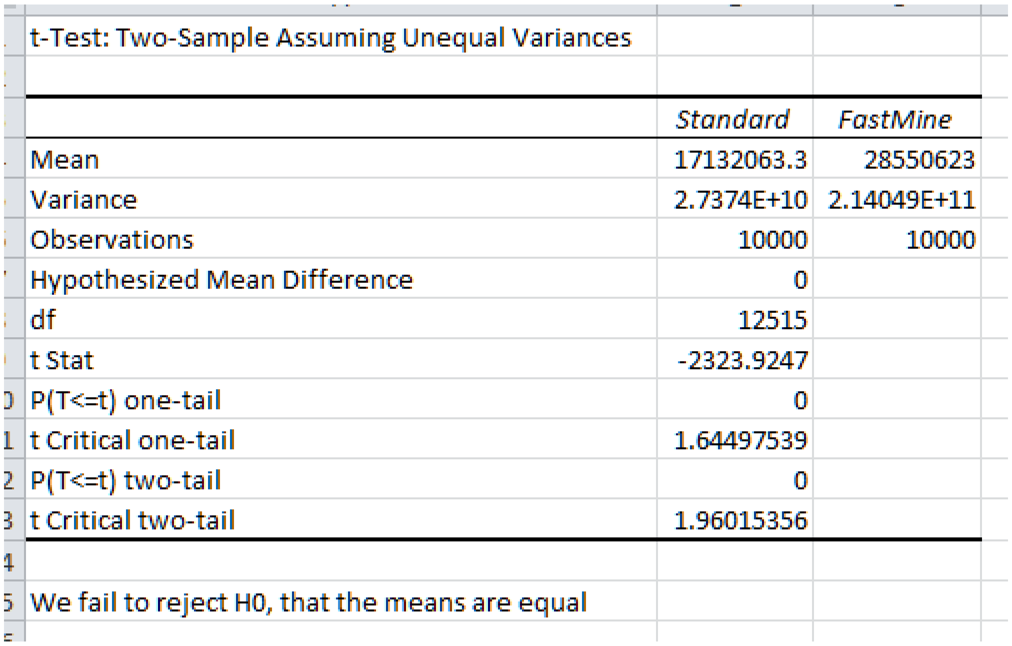
This final question was another one I thought of out of curiosity. In my initial setup, I gave the base station four charger stations because there were four robots. I was curious if having two charging stations would affect things much, so I decided to test it. In the case of MineCo, this would be good to know, because I'm sure it's quite costly to deliver all the infrastructure necessary to support up to four simultaneous chargings at once.

To answer this, I took my baseline and tweak it so it only had two charging stations, and again ran 10,000 one year simulations. Next, I ran the appropriate F and T tests, which you can see the results and conclusions of in Figures 5 and 6.



	A	B	C
1	F-Test Two-Sample for Variances		
2			
3		*Standard*	*FastMine*
4	Mean	17132063.3	28550623
5	Variance	27374130356	2.14049E+11
6	Observations	10000	10000
7	df	9999	9999
8	F	0.127887325	
9	P(F<=f) one-tail	0	
10	F Critical one-tail	0.967635045	
11			
12	We fail to reject that the variances are equal		

Figure 3: F-Test and conclusion for question 2



t-Test: Two-Sample Assuming Unequal Variances		
	Standard	*FastMine*
Mean	17132063.3	28550623
Variance	2.7374E+10	2.14049E+11
Observations	10000	10000
Hypothesized Mean Difference	0	
df	12515	
t Stat	-2323.9247	
P(T<=t) one-tail	0	
t Critical one-tail	1.64497539	
P(T<=t) two-tail	0	
t Critical two-tail	1.96015356	
We fail to reject H0, that the means are equal		

Figure 4: T-Test and conclusion for question 2

4 RNG Verification

4.1 PMF Histogram

Histogram of data generated by the PMF. Visual inspection suggests that it is similar to the raw data provided by MineCo, seen in Figure 1.

4.2 Normal Histogram

The following is 20,000 data points generated by my Guassian pRNG implementation that is used in nearly every model. As you can see, it is very normal looking.

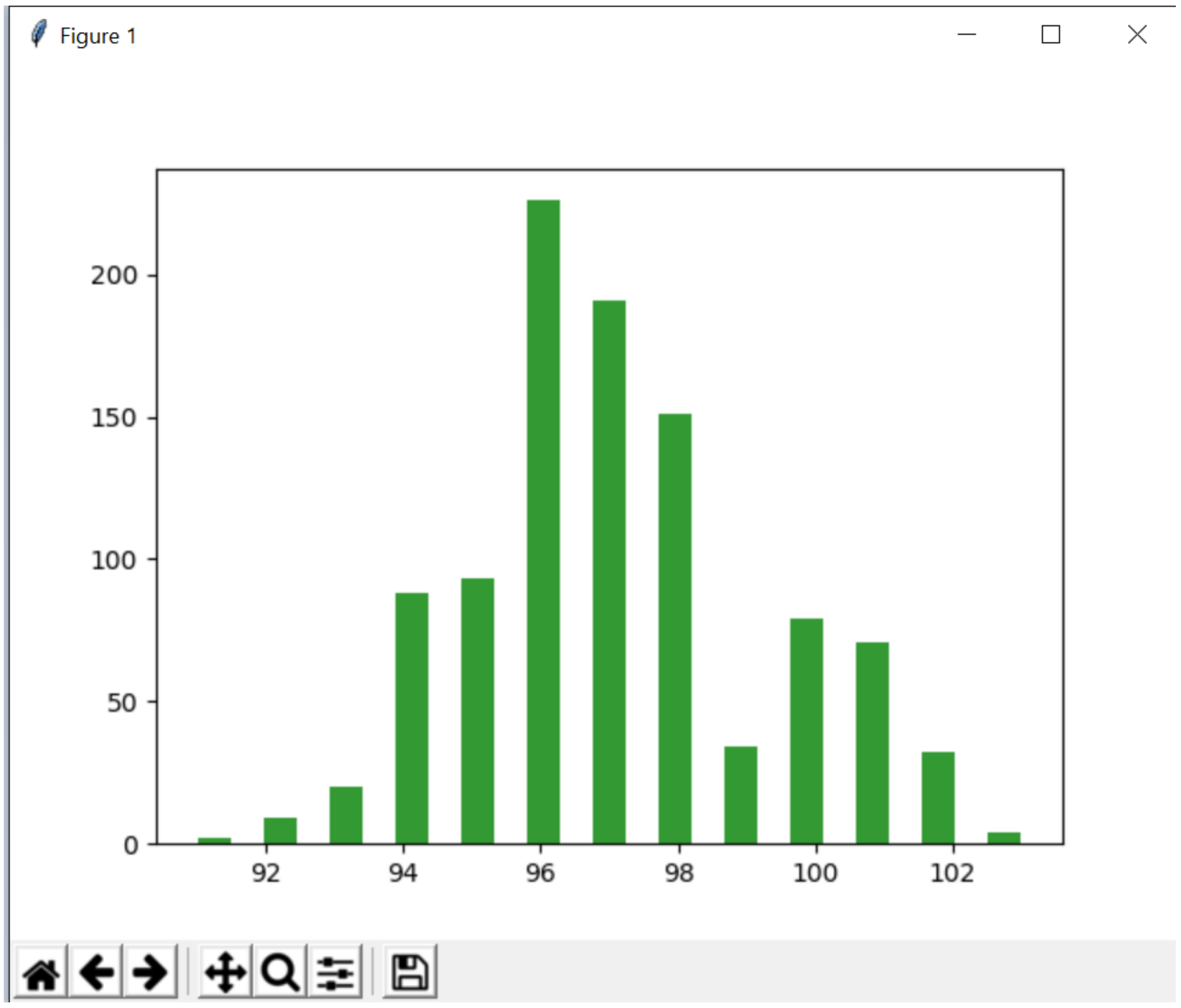


Figure 5: Histogram of data generated by the PMF.

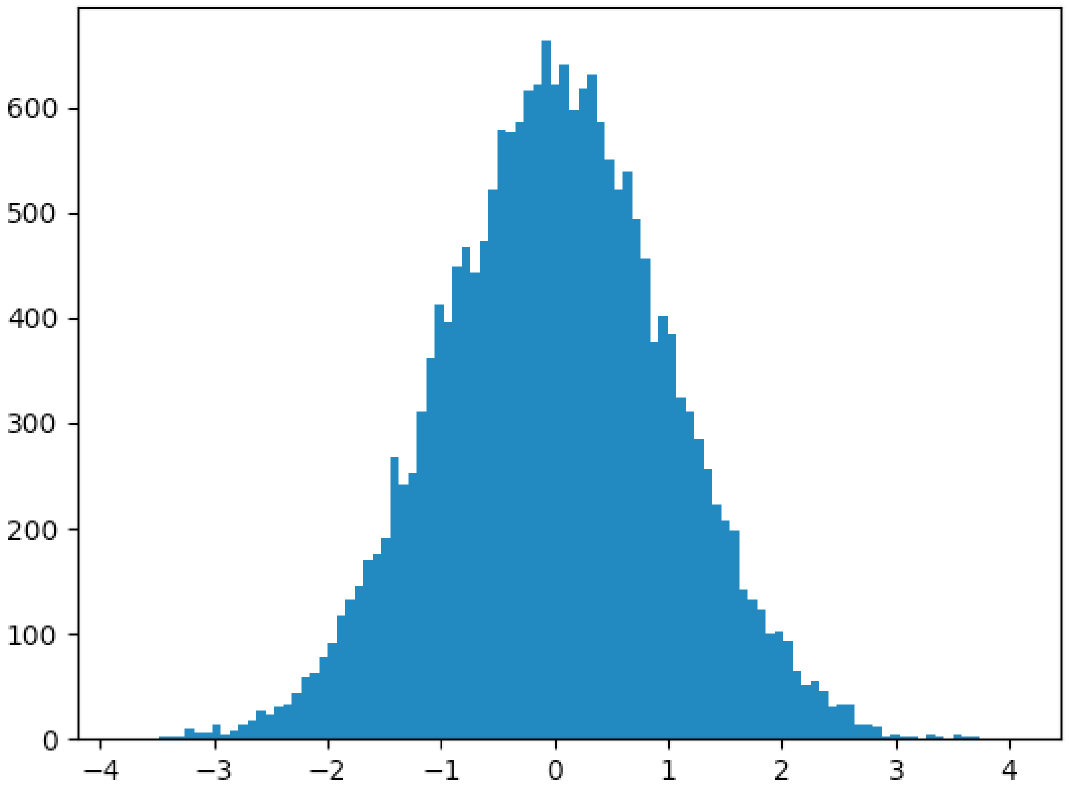


Figure 6: Histogram of data generated by the PMF.