# Reverse Engineering Algorithmic Mechanism Behind WeChat Red Envelope

Yimin Tang
School of Information Science and Technology
ShanghaiTech University

***Abstract*—This article is about WeChat Red Envelope and collects 33 examples from WeChat group as the base data. We start with the base data firstly and plot graphs like PDF, CDF, received money with time.Then we analyze these graphs using many methods of linear regression, calculating the var of examples. And we think in WeChat programmer's way to find which program will have the same results as our base data. Then we decide the final PDF of WeChat Red Envelope and write some code to test our assumption. Finally, we dope out the WeChat program about Red Envelope and the distribution of the Red Envelope.**

## I. Introduction

WeChat is widely used and its Red Envelope part is indispensable in many special days. We want to find out the distribution of the Red Envelope. The problem is that it is like a black box and we can only reverse it by Red Envelope generated data. We decide to get the answer starting with the program and get the distribution in the end.

The main results and contributions of this report are summarized as follows:

- **Base Data Visualization**
- **Modeling.**
- **Simulation Verification.**
- **Theoretical Analysis.**

## II. Base Data Visualization

We get 33 examples from WeChat and 31 examples have 5 yuan and are divided into 20 parts.

- Fig.1 : Every Example for Data visualization, x-axis is the parts of a red Envelope order by time (After normalization)
- Fig.2 : Every Example for Data 3D visualization(After normalization)
- Fig.3 : Average of all Data, x-axis is the parts of a red Envelope order by time (After normalization)
- Fig.4 : PDF of average Data, x-axis is the parts' size, the curve is calculated by linear regression and the max exponent is 7 (After normalization)
- Fig.5 : CDF of average Data, x-axis is the parts' size, the curve is calculated by linear regression and the max exponent is 7(After normalization)
- Fig.6 : Accumulation by time (After normalization)
- Fig.7 : Average accumulation by time(After normalization)
- Fig.8 : Every example's variance, x-axis is the number of examples (After normalization)
- Fig.9 : Every part's variance, x-axis is the parts of a red Envelope order by time (After normalization)

## III. Model and Algorithms

We have some assumptions based on information from the WeChat engineer and real situation in the real world. First Group Assumptions:

- Simple is good, the part of Red Envelope is not complexity.
- Money involves the part of Red Envelope, so no mistake is the most important.
- The programmer won't show his/her virtuosity in a job that has some relationship with finance.

First Group Assumptions will lead to one result — the program part is easy to reproduce by ourselves.

Second Group Assumptions:

- All money that each people get is not generated one time.
- Generating all money that each people get will cost more store resource.

Second Group Assumptions will lead to one result — each money generation is decided by last time's surplus of money and people.

But the base data shows that when you open the Red Envelope in every time, the number is close to the expectation in theory and the Var of each example is very small($10^{-3}$) but the final time's number of money will higher than other times and this position's var is not the smallest.

The PDF of each example is like $\mathcal{N}(E(X), *)$ and for the $P(X < 0.1) = 0, P(X > maxX) = 0$ , so it's may be a Truncated normal distribution.

$$f(x; \mu, \sigma, a, b) = \frac{\frac{1}{\sigma}\phi(\frac{x-\mu}{\sigma})}{\Phi(\frac{b-\mu}{\sigma}) - \Phi(\frac{a-\mu}{\sigma})}$$

$\phi(\cdot)$ is the standard normal distribution's PDF.
$\Phi(\cdot)$ is the standard normal distribution's CDF.
$P(X > b) = 0, P(X < a) = 0$

## IV. Simulation Results

The simulator is written by Python3. According to the Python3 documentation, it's random() function generates a number between $[0, 1)$ by $Unif(0, 1)$ and 1 is excluded. Our first Red Envelope generator is based on the second group assumptions and has no limit in each time you want to get a

random money number except the lowest bound and the max bound. Through this type program, the result was very terrible.

Let the first time's number be X. So the X's domain is $[0.01, max - 0.01 * people + 0.01]$, it means $P(X < 0.5 * max) < 1/2$.

It will be an Exponential function of the order number. The order of open Red Envelope will mainly decide how much you will get from the Red Envelope. But the fact is that every people will get some money near the expectations. So the failure proves that no limit in each generation is false.

According to the following two assumptions, one is that the code is simple, the other is the fact we just mentioned above here, we created the second generator. We give each generation a limit that its expectation is the expectation of last people and money.

Then we found the Var of this generator's data have the same order of magnitudes as the base data but its last opening's average money is not larger than others. So we get the third version of the generator and add a little correction.And its generated data perfectly like our base data.Maybe we get the same code just like the code in WeChat.

- Fig.10 : Average of all Data by Generator 1, x-axis is the parts of a red Envelope order by time (After normalization)
- Fig.11 : Average of all Data by Generator 2, x-axis is the parts of a red Envelope order by time (After normalization)
- Fig.12 : Every example's variance by Generator 2, x-axis is the number of examples (After normalization)
- Fig.13 : Every part's variance by Generator 2, x-axis is the parts of a red Envelope order by time (After normalization)
- Fig.14 : Average of all Data by Generator 3, x-axis is the parts of a red Envelope order by time (After normalization)
- Fig.15 : Every example's variance by Generator 3, x-axis is the number of examples (After normalization)
- Fig.16 : Every part's variance by Generator 3, x-axis is the parts of a red Envelope order by time (After normalization)
- Fig.17 : CDF of average Data, x-axis is the parts' size, the curve is calculated by linear regression and the max exponent is 7 (After normalization)
- Fig.18 : PDF of average Data, x-axis is the parts' size, the curve is calculated by linear regression and the max exponent is 7 (After normalization)

## V. Result Analysis

Give a Red Envelope with two parameters $a, n$, a is the total money and n is the total number of people. Let $X_1, X_2, \ldots, X_n$ is the money that n people get.

$E(X_1) = \frac{a}{n}$

$E(X_2) = E(\frac{a-E(X_1)}{n-1}) = \frac{a}{n}$

If $X_1, X_2, \ldots, X_{n-1}$ are all satisfied $E(X_i) = \frac{a}{n}$

$E(X_n) = E(\frac{a-E(X_1+X_2+\ldots+X_{n-1})}{n-(n-1)})$
$= a - E(X_1 + X_2 + \ldots + X_{n-1})$
$= \frac{a}{n}$

So $E(X_i) = \frac{a}{n}$ for all positions. The theory result is satisfied the real data. The simple idea make sure the expectation of all people who received the money from Red Envelope is equal.

And the CDF or PDF of the real base data is very like Truncated normal distribution's CDF or PDF.(Fig.4, Fig.5) So we have reasons to believe WeChat Red Envelope's distribution is Truncated normal distribution.

## VI. Conclusions

The code mat be very simple and the generate function is that its expectation is the expectation of last people and money. This simple idea create the result of Truncated normal distribution.


## Acknowledgment

During this project, I did it by myself. I also adopted part of contents from Zhihu [1] to Section III and get imformation of Truncated normal distribution from [2]. All my image, data and code [3] are on my Github website


## References


[1] "Discussions in zhihu," https://www.zhihu.com/question/22625187.
[2] "Tnd," https://en.wikipedia.org/wiki/Truncated_normal_distribution.
[3] "code," https://github.com/TachikakaMin/WechatRedPackets.


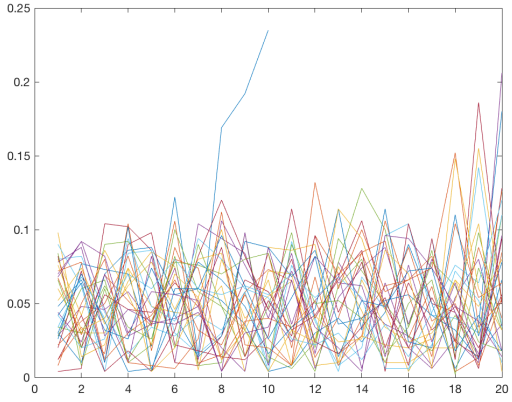


Fig. 1. Every Example for Data visualization

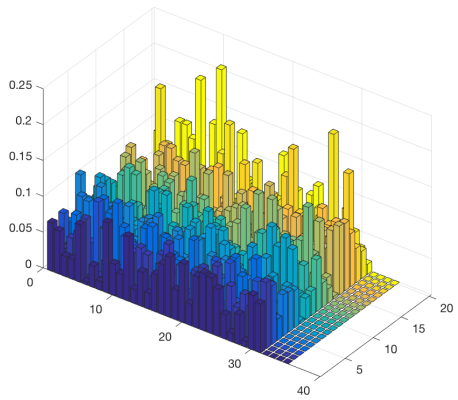


Fig. 2. Every Example for Data 3D visualization

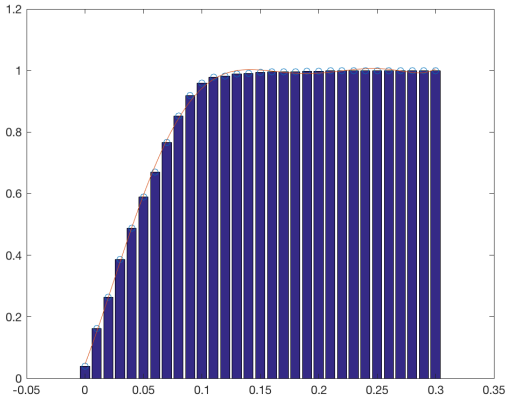


Fig. 5. CDF of average Data

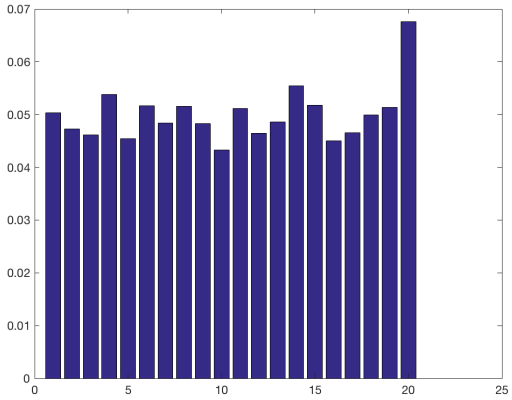


Fig. 3. Average of all Data

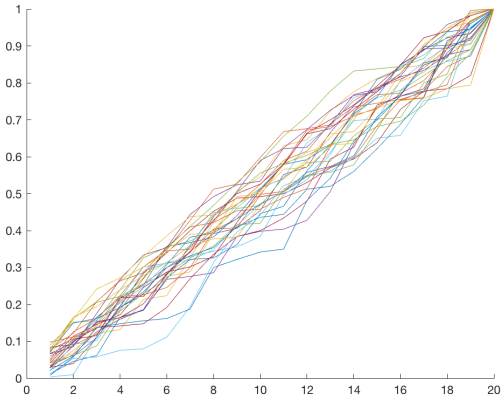


Fig. 6. Accumulation by time

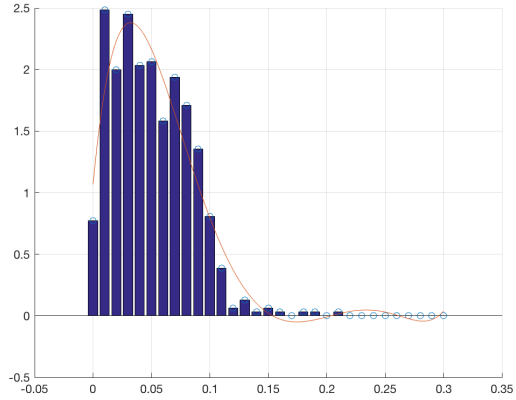


Fig. 4. PDF of average Data

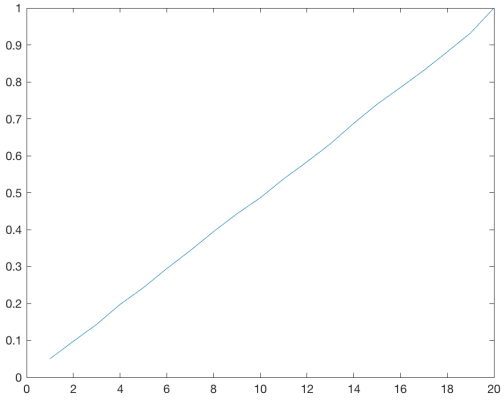


Fig. 7. Average accumulation by time

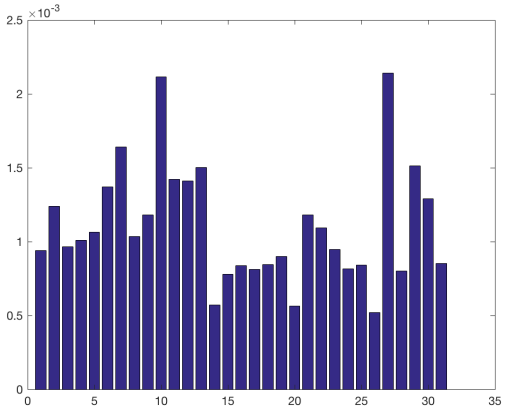


Fig. 8. Every example's variance

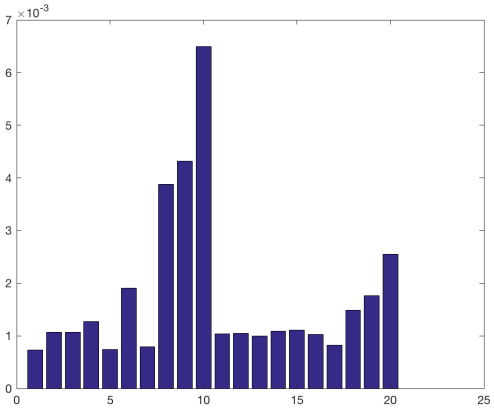


Fig. 9. Every part's variance

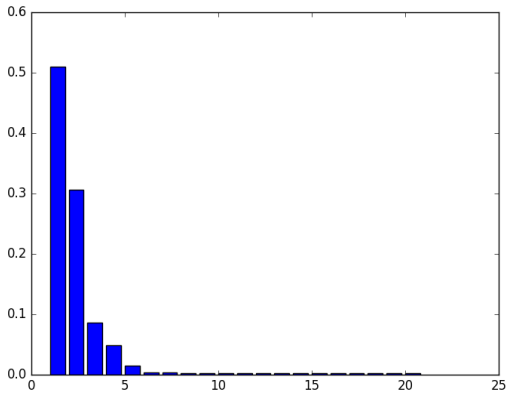


Fig. 10. Average of all Data by Generator 1

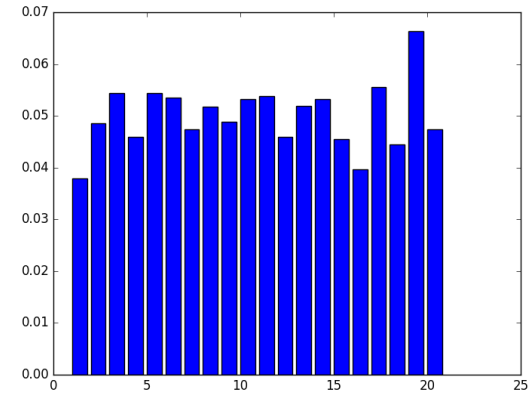


Fig. 11. Average of all Data by Generator 2

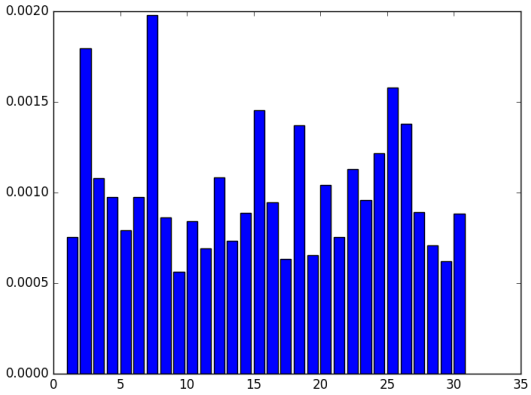


Fig. 12. Every example's variance by Generator 2

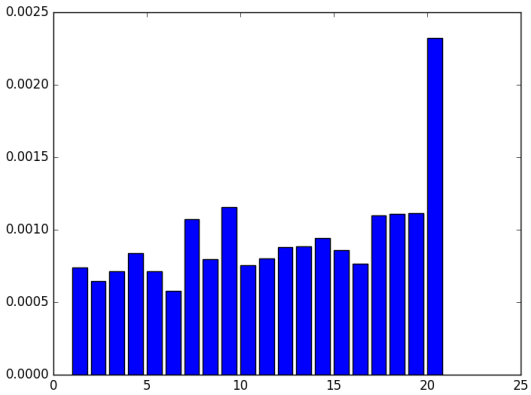


Fig. 13. Every part's variance by Generator 2

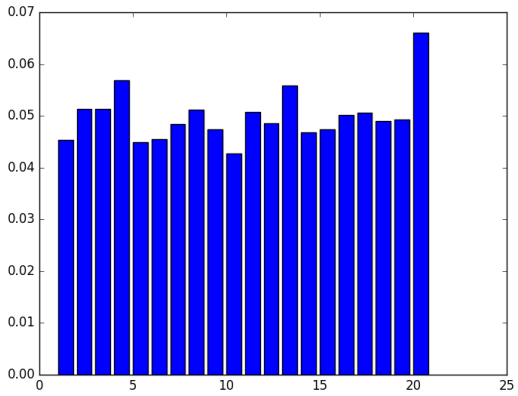


Fig. 14. Average of all Data by Generator 3

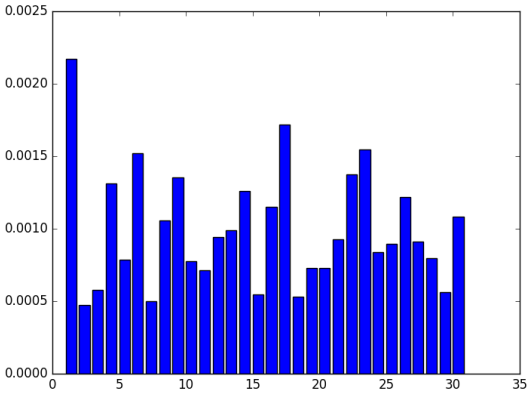


Fig. 15. Every example's variance by Generator 3

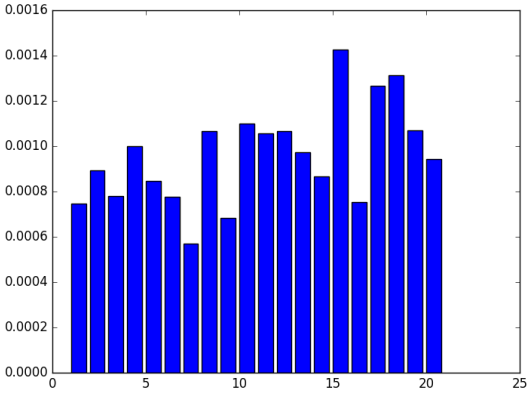


Fig. 16. Every part's variance by Generator 3

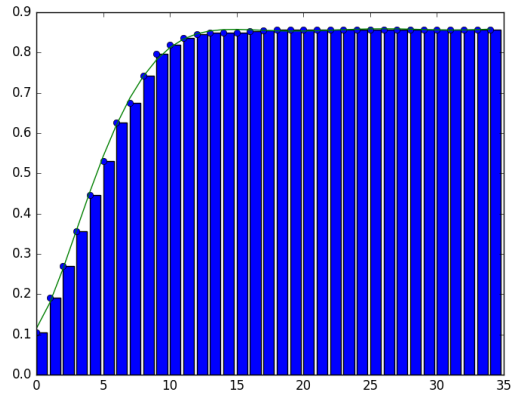


Fig. 17. CDF by Generator 3

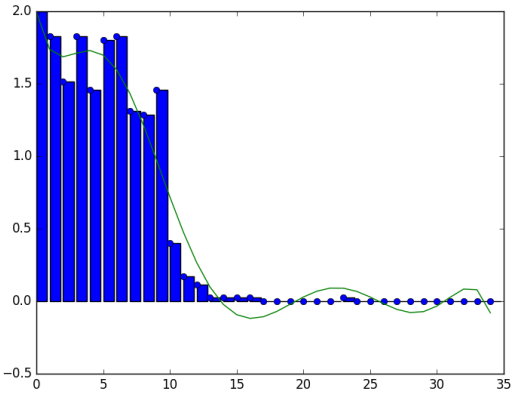


Fig. 18. PDF by Generator 3

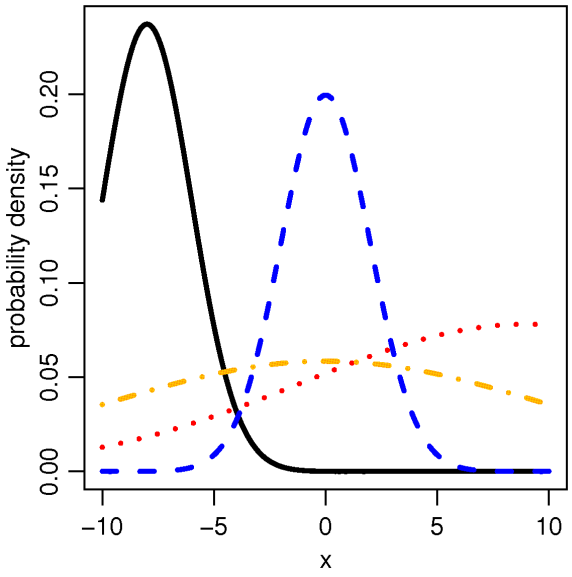


Fig. 19. CDF of average Data

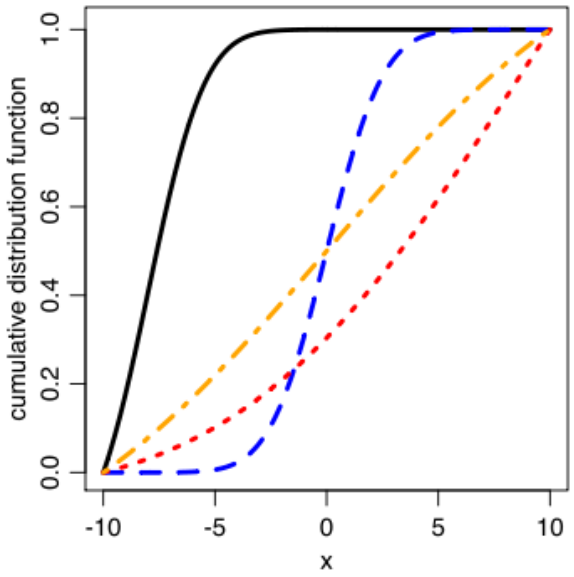


Fig. 20. CDF of average Data