

Reverse Engineering Algorithmic Mechanism Behind WeChat Red Envelope

Ziyang Wang

wangzy4@shanghaitech.edu.cn

School of Information Science and Technology

ShanghaiTech University

Abstract—In this project, I would like to find the algorithmic mechanism behind WeChat red envelope. At the beginning of the project, we made 180 same random red envelope in a 6 people group to collect the original data. After collecting the data, I compared them with some distribution that it possibly use, and finally find that the algorithmic mechanism of WeChat random red envelope follows uniform distribution, but the parameter of the distribution could be changed by the money which had been withdrawn.

I. INTRODUCTION

At the beginning, I made a 6 people group with Xincheng Wang, Kefei Wu, Yiduo Gu, Haonan Liu and Zhenghang Zhi. To make the result more accurate, we chose to test as more times as possible and get more details of the data instead of making a large group but just made red envelope while not considering the order of taking them.

Our strategy is to make everyone take the same order red envelopes same times, it would be like this:

Plan/Order	1	2	3	4	5	6
1	Xincheng	Ziyang	Yiduo	Kefei	Zhenghang	Haonan
2	Ziyang	Yiduo	Kefei	Zhenghang	Haonan	Xincheng
3	Yiduo	Kefei	Zhenghang	Haonan	Xincheng	Ziyang
4	Kefei	Zhenghang	Haonan	Xincheng	Ziyang	Yiduo
5	Zhenghang	Haonan	Xincheng	Ziyang	Yiduo	Kefei
6	Haonan	Xincheng	Ziyang	Yiduo	Kefei	Zhenghang

We used 60 yuan red envelope to test each plan 30 times, and the result of each people was in Fig1. Not only the probability density, but also the maximum value/minimum value/expectation/variance do not have obvious different, and the expectation is close to the expectation of one red envelope. So I assume that the distribution of the red envelope has nothing to do with specific person.

Therefore, I began to study the relationship between the value of the red envelope and the order which was taken.

Order	1	2	3	4	5	6
Average	9.65	10.58	10.20	9.20	10.78	9.64
Variance	34.22	37.72	44.41	40.51	57.08	55.35
Max	19.92	23.04	28.08	29.74	35.62	38.47
Min	1	0.8	0.83	0.67	0.5	0.03

It's obvious that the average number of each red envelope is close to the expectation 10, but the variance is increasing with the order. Another interesting result is that I found that the minimum value was never lower than $\frac{1}{10}$ * the average number of the remaining money. Also, the number of 1 in the first

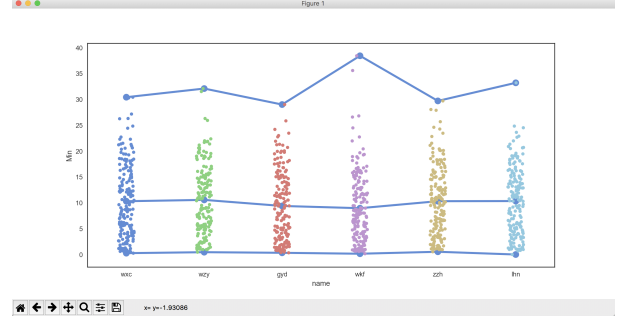


Fig. 1. The result of different people

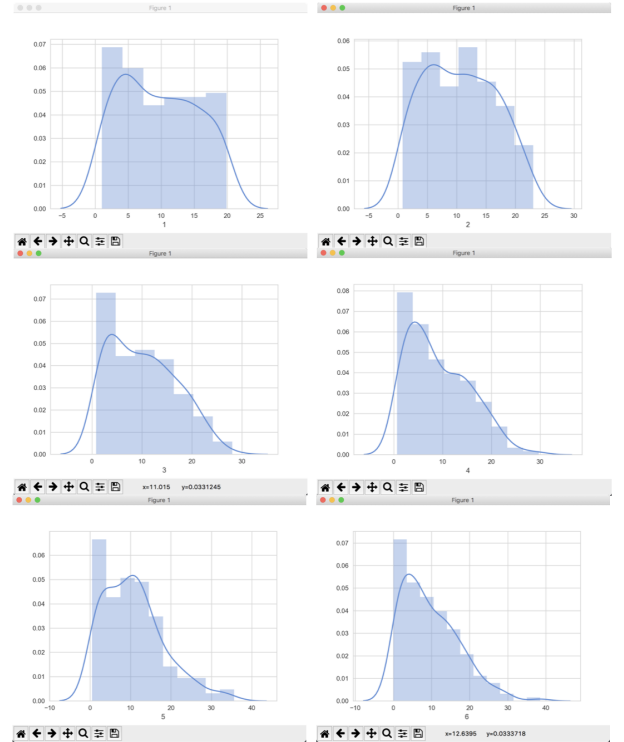


Fig. 2. The fitting PDFs of six orders

envelope is obviously higher than others. That's why I assume that if the random number less than $\frac{1}{10}$ * the average number of the remaining money, it will be increased to $\frac{1}{10}$ * the average number of the remaining money to reduce the gap.

After plotting the fitting probability density function curve

of every order as Fig.2, I started modeling to search for the algorithm.

II. MODEL AND ALGORITHMS

After searching for some articles online, I found three opinions of the algorithms.

- Normal distribution
- Truncated normal distribution
- Uniform distribution.

Because the value of the first envelope would never be affected by other values, I try to use the data of the first envelope to find the fitting distribution, and use the other 5 groups of data to test and verify it.

For Normal Distribution, I use the expectation 10 and variance 34.22 at the beginning.

For Truncated Normal Distribution, I use the same μ and σ with the normal distribution, but I set the lower x and upper x -5 and 25, by observing the PDF of the real data.

Because the maximum value of the first envelope was never greater than 20, I assume the value of the first envelope follows Unif(0.01,20).

And like I guess before, when the random value less than $\frac{1}{10}$ *the average number of the remaining money, it will be increased to $\frac{1}{10}$ *the average number of the remaining money.

III. SIMULATION RESULTS

After the test and plot (Fig.3), although it might be some mistake on parameters of Normal/Truncated Normal Distributions, I tend to believe that the algorithm follows Uniform Distribution instead of others.

Further more, even though Normal/Truncated Normal Distributions could get a similar result to the real data by adjusting the parameters again and again, it's unknown for what it would become in the second envelope. Therefore, even if it succeed, it might just be a coincidence, and it's complicated to find what the algorithm will do next. I believe "less is better", and I believe that it couldn't be a complicated algorithm when dividing red envelopes, because it's meaningless for a company to spend a lot of time on a simple function in their app. Finally, I decided to search more for the uniform distribution they use.

IV. RESULT ANALYSIS

To find more about what would the algorithm do after the first envelope was withdrawn, I need to observe our original data again. RAvGMax and RAvGMin means the average number of remaining money when the envelope got it's maximum/minimum number.

Order	1	2	3	4	5	6
Max	19.92	23.04	28.08	29.74	35.62	38.47
RAvgMax	10	11.8	14.355	15.09	18.38	38.47
Min	1	0.8	0.83	0.67	0.5	0.03
RAvgMin	10	8.016	8.35	6.75	5.095	0.03

First, my guess about the lower limit was never less than $\frac{1}{10}$ *the average number of the remaining money was verified. At the same time, I found that the maximum value was never

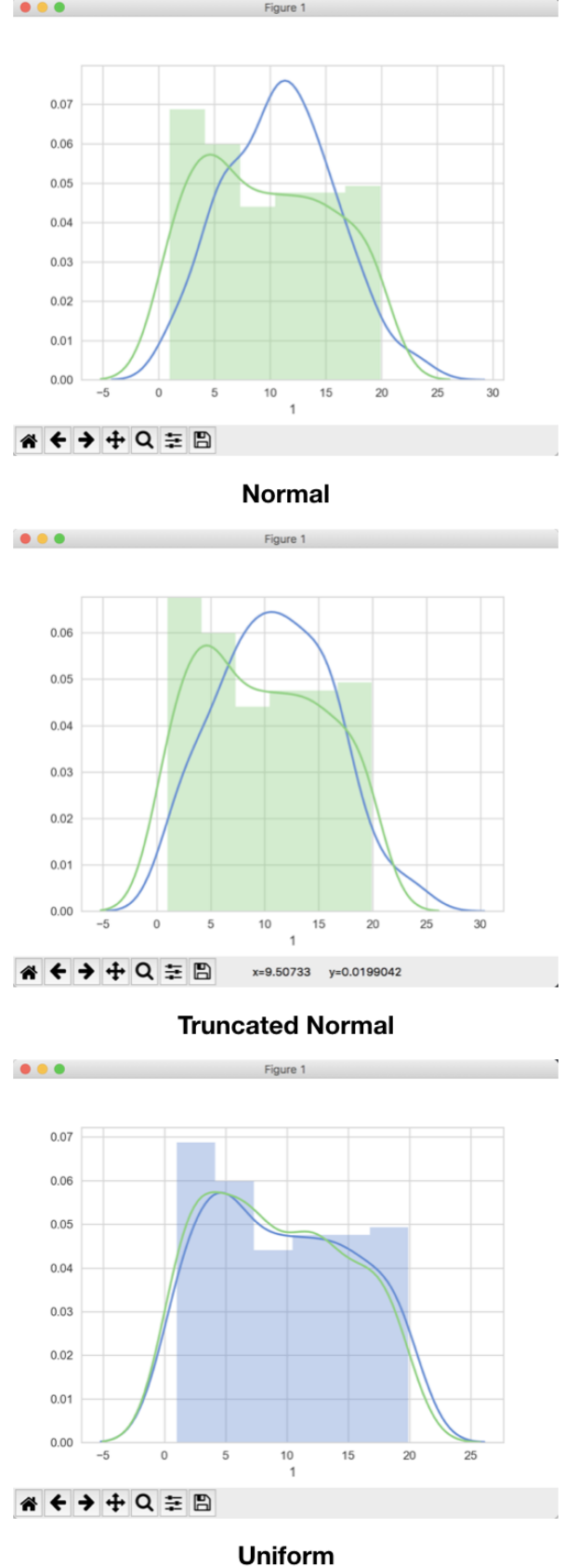


Fig. 3. The result of three distributions

larger than $2 \times$ the average number of the remaining money. As a result, I assume the value of the envelope follows $\text{Unif}(0.01, 2 \times \text{RAvg})$, and when the random value is less than $\frac{1}{10} \times \text{RAvg}$, it becomes $\frac{1}{10} \times \text{RAvg}$.

Also, to make sure that everyone can get money, if the penultimate envelope has all of the remaining money, I let it random again to assure there's money left in the last envelope.

This algorithm is recursive, which means it can be used in whatever the total number and value the red envelope is. So it can fit any conditions, and the value would never be negative, which is better than Normal distribution.

V. COMPARING WITH THE REAL DATA

I write a program following my algorithm in python, and made 180 red envelopes as our real experiment. The result simulated by the algorithm was almost perfectly matched the result of our real experiment as Fig.4,5,6,7,8,9 shows.

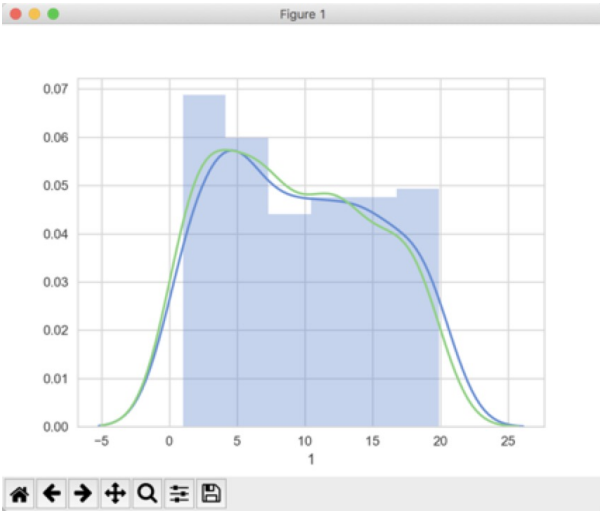


Fig. 4. The result of the first envelope

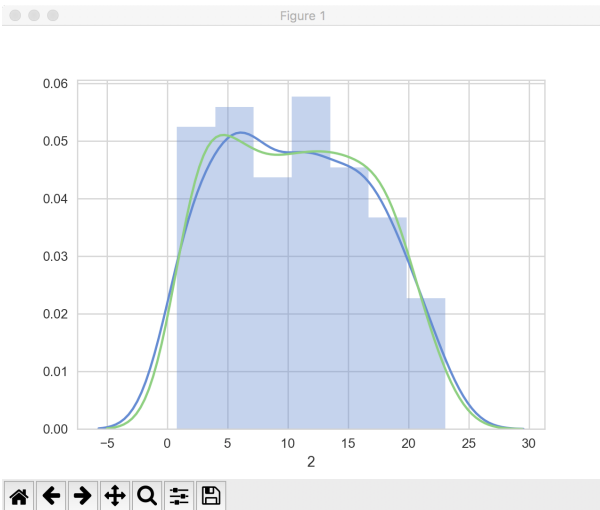


Fig. 5. The result of the second envelope

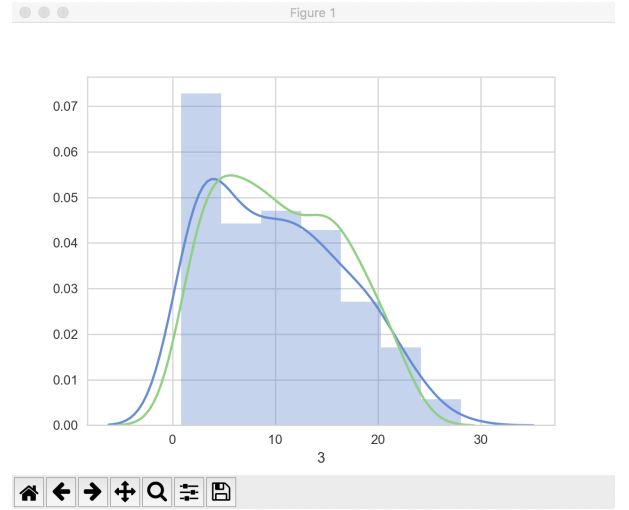


Fig. 6. The result of the third envelope

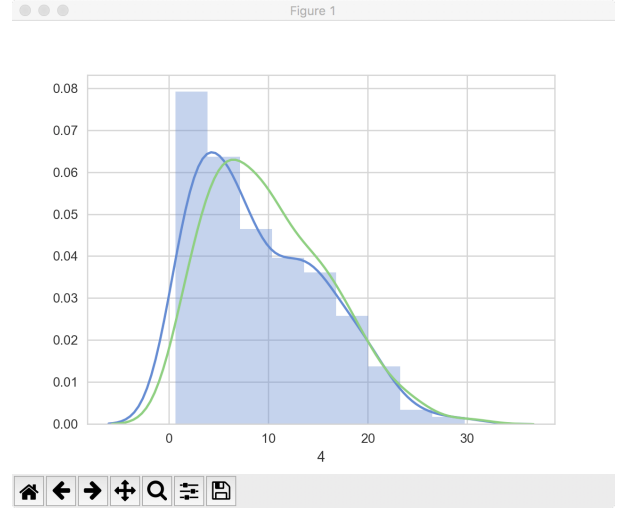


Fig. 7. The result of the fourth envelope

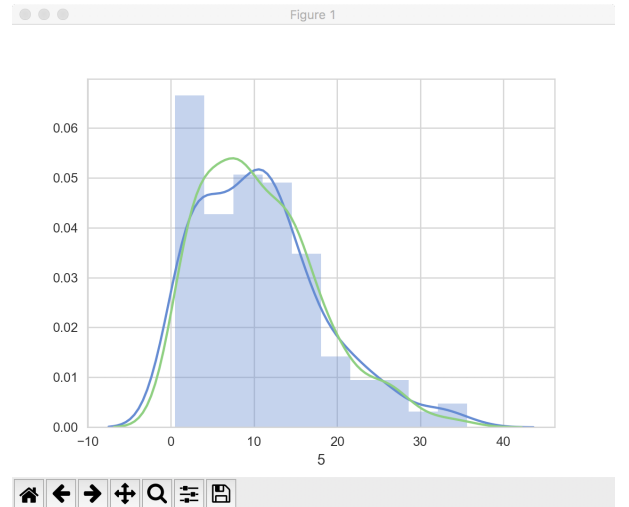


Fig. 8. The result of the fifth envelope

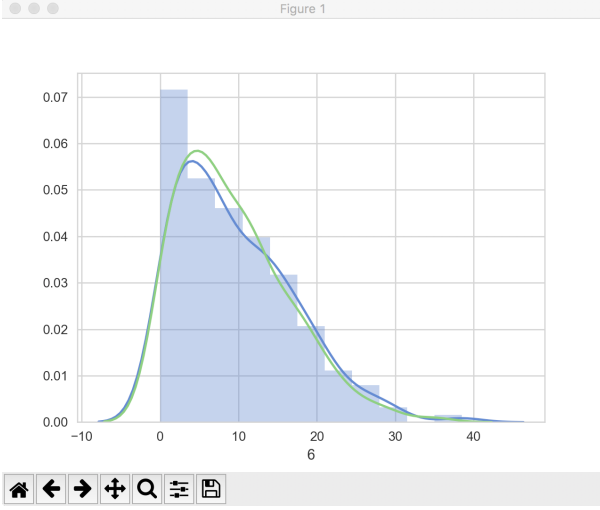


Fig. 9. The result of the sixth envelope

VI. FURTHER SIMULATIONS

After getting the algorithm, I made a test to make 1000000 red envelopes instead of 180. The result was as below:

Order	1	2	3	4	5	6
E	10.03	10.03	10.01	10.00	9.99	9.93
Max	19.99	23.60	28.90	37.24	53.24	51.14
Min	1	0.8	0.6	0.42	0.23	0.01
Var	32.84	34.56	37.26	42.31	53.62	53.70

Which means the expectations of 6 envelope are almost same, but the first one will be the largest, but the variance of the first envelope is the smallest, which means it will not get extremely small value, but extremely large value either.

On the contrast, the last envelope is with the highest risk. When the last envelope has the largest number, it is most likely a very large number, but at the same time, the last envelope has the highest probability to become the smallest one.

VII. CONCLUSIONS

From the research above, my conclusion is that the distribution of WeChat envelope follows the Uniform Distribution from 0.01 to the double of the average number of remaining money each time, except the last envelope.

This strategy make it relatively fair, because the expectation of every red envelope are almost same. At the same time, the different variance increases its interest and excitement.

ACKNOWLEDGMENT

During this project, I made the red envelope experiment with Xincheng Wang, Zhenghang Zhi, Kefei Wu, Haonan Liu and Yiduo Gu, which means I share the same dataset with them. Meanwhile, I discussed with my classmates Kefei Wu, and he told me the algorithm when the value is less than $\frac{RAvg}{10}$.

REFERENCES

- [1] Discussions in zhihu, <https://www.zhihu.com/question/22625187>.
- [2] Bidao's WeChat article, <http://mp.weixin.qq.com/s/JA0gVWtedLpUk4XgLnCg>.