Reverse Engineering Algorithmic Mechanism Behind WeChat Red Envelope

Ziyang Wang
wangzy4@shanghaitech.edu.cn

School of Information Science and Technology
ShanghaiTech University

January 20, 2018

Introduction

At the beginning, I made a 6 people group with Xinchen 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 cosidering the order of taking them.

Introduction

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	WXC	WZY	GYD	WKF	ZZH	LHN
2	WZY	GYD	WKF	ZZH	LHN	WXC
3	GYD	WKF	ZZH	LHN	WXC	WZY
4	WKF	ZZH	LHN	WXC	WZY	GYD
5	ZZH	LHN	WXC	WZY	GYD	WKF
6	LHN	WXC	WZY	GYD	WKF	ZZH

Introduction

We used 60 yuan red envelope to test each plan 30 times. 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 expection of one red envelope. So I assume that the distribution of the red envelope has nothing to do with specific person.

Introduction

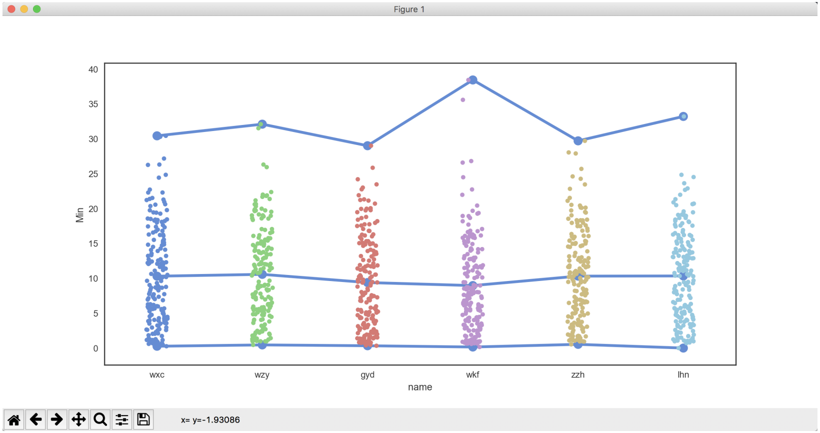


Figure: The result of different people

Introduction

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

Introduction

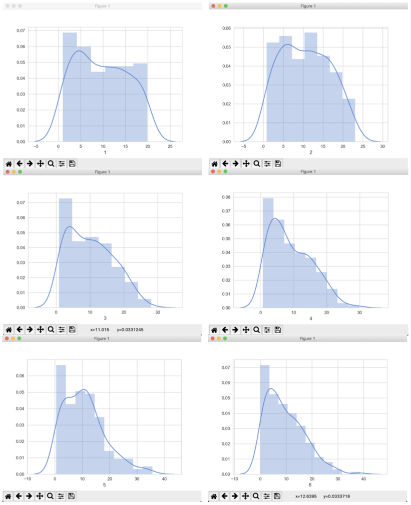


Figure: The fitting PDFs of six orders

Introduction

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 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.

Introduction

After plotting the fitting probability density function curve of every order, I started modeling to search for the algorithm.

Model and Algorithms

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

- **Normal distribution**
- **Truncated normal distribution**
- **Uniform distribution**.

Model and Algorithms

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.

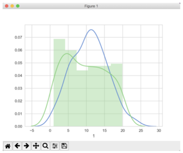
Model and Algorithms

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

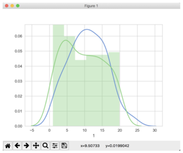
Model and Algorithms

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.

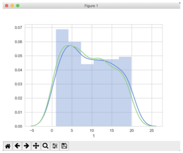
Simulation Results



Normal



Truncated Normal



Uniform

Figure: The result of three distributions

Simulation Results

After the test and plot, 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.

Simulation Results

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.

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	14.35	15.09	18.38	38.47
Min	1	0.8	0.83	0.67	0.5	0.03
RAvgMin	10	8.02	8.35	6.75	5.095	0.03

Simulation Results

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

Simulation Results

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

Simulation Results

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.

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.

Comparing with the real data

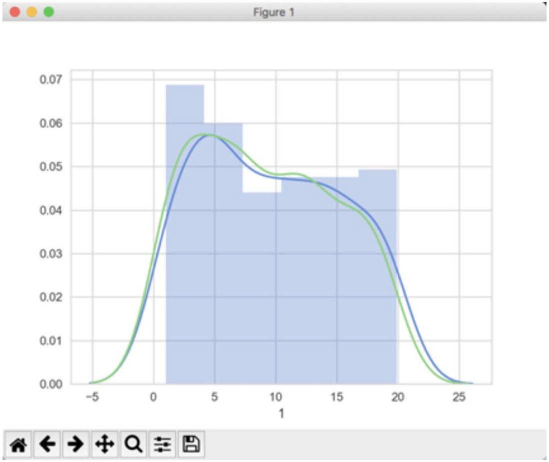


Figure: The result of the first envelope

Comparing with the real data

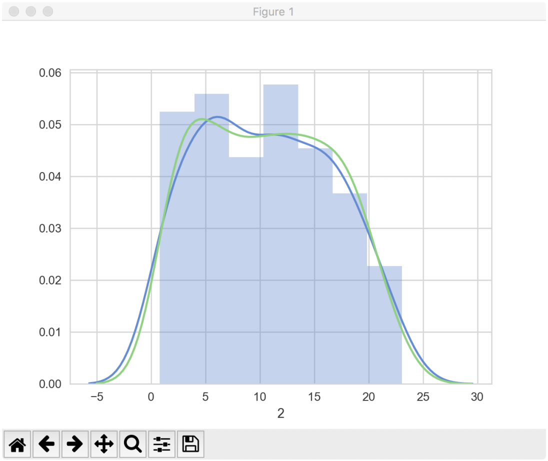


Figure: The result of the second envelope

Comparing with the real data

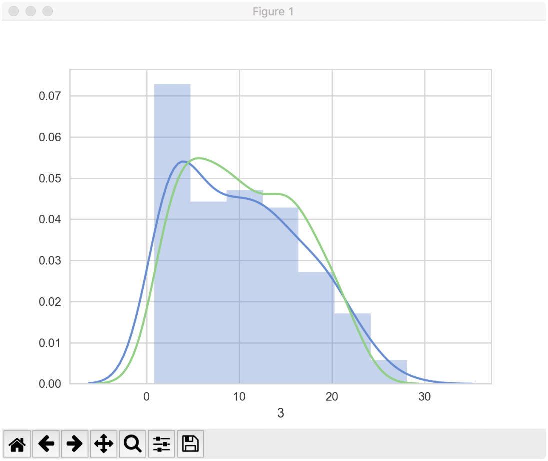


Figure: The result of the third envelope

Comparing with the real data

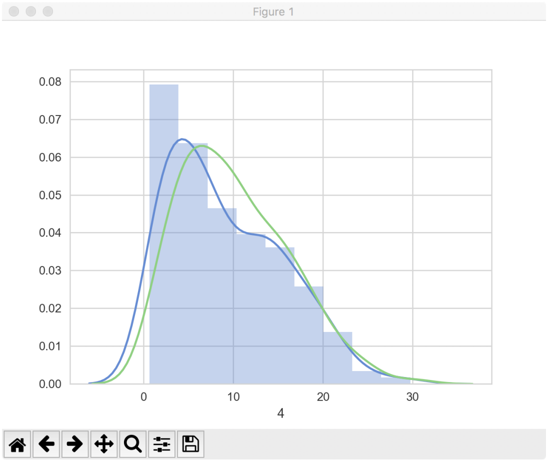


Figure: The result of the forth envelope

Comparing with the real data

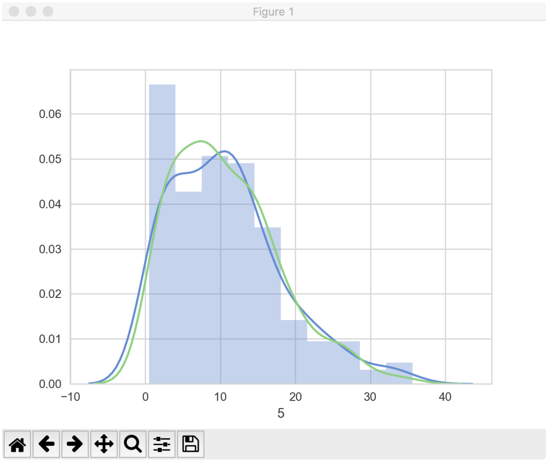


Figure: The result of the fifth envelope

Comparing with the real data

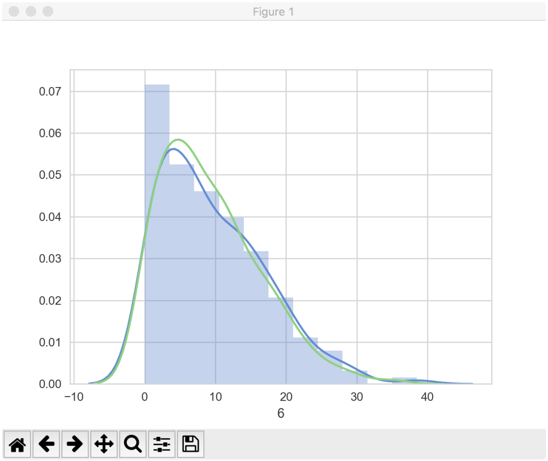


Figure: The result of the sixth envelope

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

Further simulations

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

Further simulations

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.

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 varience increases it's interest and excitement.

Acknowledgment

During this project, I made the red envelope experiment with Xinchen 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 $\frac{RAvg}{10}$.

Reference

- Discussions in Zhihu: https://www.zhihu.com/question/22625187.
- Bidao's WeChat article