Modelling Rubik's Cube Times with Probability Density Functions

Mathematics: Analysis and Approaches HL Exploration

I confirm that this work is my own work, and it is a final version. I have acknowledged each use of the words or ideas of another person, whether written, oral or visual.

Number of pages : 1 title page, 19 exploration pages, 1 bibliography page

Total : 21 pages

Introduction

When I was younger, Rubik's cubes were the first thing I was ever truly good at, so naturally I dreamed of being a professional speedcuber, breaking records, winning competitions, and getting endorsements from Rubik's cube companies. From the moment I first solved one I was instantly hooked. Seeing my times improve and passing benchmarks such as sub-1-minute, sub-30 seconds, sub-20 seconds, and finally sub-15 seconds gave me the experience of an emotional high as I got faster and faster. Alas, I eventually hit a barrier and it got harder and harder to improve and shave more seconds off of my times. The amount of work it would take for me to get to the next benchmark, averaging sub-10, was astronomical, and so I gave up on my dream to become a professional speedcuber. I still pick up the hobby and have fun solving, but I have since found other things that I enjoy and am good at such as math and physics. The first time I solved a physics problem and saw how the real physical world was described by mathematics left me in awe of the elegance and beauty of math, and now I dream of being a physicist because of my interest in using math to model the real world. For this math IA I wanted to find a way to combine my interest in Rubik's cubes with my interest for using math to model real world phenomena.

In 2018, Yusheng Du broke the previous world of 4.22 seconds by a whole 0.75 seconds with a time of 3.47, which almost seemed too good to be true. This record was even more incredible when considering that Yusheng Du averages in between 8 and 9 seconds, while most of the top elite speedcubers average between 5 and 7 seconds. Additionally, Yusheng Du's world record solve only took 27 moves! For reference, the average move count of elite speedcubers is around 55-60 moves. To some, all of this was evidence that something was suspicious about this world record, but at the very least Yusheng Du got very lucky with his world record. This inspired me to investigate the role luck plays in Rubik's Cube solving with mathematics. I will do this by applying single variable statistical analysis techniques to create probability density functions to model Rubik's Cube solve times. By modelling these Rubik's cube solve times with probability density functions, I can get an approximate idea of the underlying theoretical probability and make use it to find an estimation of what Yusheng Du's theoretical probability of getting such a good solve might have been, rather than simply looking at the experimental probability based on the data. Modelling raw data with a probability density function will give me a more general understanding of how the underlying probability in the data is behaving.

While Yusheng Du averaging in between 8 and 9 seconds and breaking the world record with 3.47 was enough to cause suspicion, I personally average between 12 and 13 seconds. That means it would be practically impossible for me to break the world record. Even when I do get exceptionally lucky my personal best time of 7.37 seconds is more than double the world record. Nevertheless, I thought it would be fun to sit down and do a ton of solves, record the data, and see what the statistical models I build predict my odds of breaking the world record are.

In summary, the main aim of the investigation is to model Rubik's Cube times using probability density functions.

Continuous Probability Distributions

Theoretically, a Rubik's Cube solve time can take on any value in a continuous spectrum, although in practice they are only measured to one one-hundredth of a second of precision. Nevertheless, this suggests we should use a continuous probability distribution to model the data. The most common continuous probability distribution is the normal distribution. It is described by the function $f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2}$, where μ is the sample mean and σ is the sample standard deviation. While many probabilistic phenomena are modelled with normal distributions, I do not expect Rubik's Cube data to be normally distributed. That is because it is very difficult to get times significantly better than your mean, so I expect less data to the left of the mean, and thus the data would be right skewed instead of symmetric around the mean like a normal distribution is.

Two candidates for continuous probability distributions which would model this predicted skewed behavior are the log-normal distribution and Weibull distribution. Along with the skewed behaviors of both of these distributions, there are two other reasons why these distributions can be justified as good candidates to model the data:

1. They are both versatile distributions since they both have multiple degrees of freedom in the form of parameters which can be adjusted to fit a set of data, and these degrees of freedom are able to describe skew.
2. They are both restricted so that $f(x) = 0, x < 0$, which makes sense since the probability of solving a Rubik's cube in less than 0 seconds should be 0. This cannot be guaranteed with a normal distribution, where the restriction has to be artificially imposed. This may suggest that the log-normal and Weibull distributions could be more natural fits to model the data.

Log-normal distribution

A log-normal distribution is simply defined to be the distribution for which taking the natural logarithm of the random variable produces a normal distribution. "A random variable X is log-normally distributed if (the random variable) Y=ln(X) is normally distributed" *. The probability density function of a log-normal distribution is $f(x) = \frac{1}{x\sigma\sqrt{2\pi}} e^{-\frac{(lnx-\mu)^2}{2\sigma^2}}$. While μ and σ were the mean and standard deviation of the dataset for the normal distribution, for modelling data with a log-normal distribution, μ and σ are actually free parameters which we can vary to fit the data best. This is because μ and σ in this case are the mean and standard deviation of Y=ln(X), not X itself, and thus we have the freedom to choose any normal distribution characterized by some μ and σ such that the log-normal distribution produced by it best resembles the data.

*Quote taken from: https://www.itl.nist.gov/div898/handbook/eda/section3/eda3669.htm

Weibull distribution

The Weibull distribution is widely used to model many different situations due to how versatile it is since it has 3 free parameters that allow it to describe a large variety of behaviors. The probability density function for a 3-parameter Weibull Distribution is:

$$f(x) = \frac{\gamma}{\alpha}\left(\frac{x-\mu}{\alpha}\right)^{\gamma-1} e^{-\left(\frac{x-\mu}{\alpha}\right)^{\gamma}}$$

Where μ, γ, and α are the free parameters. γ is called the shape parameter since it determines the shape of the distribution. α is known as the scale parameter and is analogous to the role standard deviation plays in a normal distribution. μ is known as the location parameter and plays a similar role as the mean in a normal distribution, since it horizontally shifts the distribution. A lot of the time, μ is set to 0, which leads to the more common 2-parameter Weibull distribution:

$$f(x) = \frac{\gamma}{\alpha}\left(\frac{x}{\alpha}\right)^{\gamma-1} e^{-\left(\frac{x}{\alpha}\right)^{\gamma}}$$

The Weibull distribution is used to model scenarios with a feature known as "time to failure". Usually time to failure refers to something such as the time for a product to become defective, but in the situation this math IA deals with, time to failure refers to the time for the Rubik's Cube to be solved. The third parameter μ is used if there is some failure-free time period, and in this context a failure-free time period would correspond to the fact that it is physically impossible for a human to solve a Rubik's Cube under 1 second.

Chi-Square Goodness of Fit Test

The Chi-Square Goodness of Fit Test is a way to quantify how well a set of data grouped into bins follows a probability distribution. The Chi Squared statistic is defined as:

$$\chi^2 = \sum_{i=1}^{k} (O_i - E_i)^2 / E_i$$

This Chi-Squared statistic measures the difference between the observed value for a bin compared to the expected value of the bin based on a probability distribution's cumulative distribution function. As in many other statistical measures such as variance, the difference is squared so that all the quantities are positive.

In the definition for the Chi-squared statistic, k is the number of bins, O_i is the observed frequency of a bin, and E_i is the expected frequency of a bin. The expected frequency is found through the following equation:

$$E_i = n\big(F(Y_u) - F(Y_l)\big)$$

Where n is the population size of the dataset, F is the cumulative distribution function of the probability distribution we are testing if it fits, and Y_u and Y_l are the upper and lower limits of the i-th bin respectively. If $\chi^2 > \lambda$ where λ is some critical value found through a Chi-square

probability table, then the observed data is too far from the theoretical probability distribution to be a good fit.

The critical value is found through a Chi-square probability table, which is based on the probabilities of a different probability distribution called the Chi-Square distribution.

df	0.995	0.99	0.975	0.95	0.90	0.10	0.05	0.025	0.01	0.005
1	---	---	0.001	0.004	0.016	2.706	3.841	5.024	6.635	7.879
2	0.010	0.020	0.051	0.103	0.211	4.605	5.991	7.378	9.210	10.597
3	0.072	0.115	0.216	0.352	0.584	6.251	7.815	9.348	11.345	12.838
4	0.207	0.297	0.484	0.711	1.064	7.779	9.488	11.143	13.277	14.860
5	0.412	0.554	0.831	1.145	1.610	9.236	11.070	12.833	15.086	16.750
6	0.676	0.872	1.237	1.635	2.204	10.645	12.592	14.449	16.812	18.548
7	0.989	1.239	1.690	2.167	2.833	12.017	14.067	16.013	18.475	20.278
8	1.344	1.646	2.180	2.733	3.490	13.362	15.507	17.535	20.090	21.955
9	1.735	2.088	2.700	3.325	4.168	14.684	16.919	19.023	21.666	23.589
10	2.156	2.558	3.247	3.940	4.865	15.987	18.307	20.483	23.209	25.188

Fig. 1 First 10 entries of a Chi-Square probability table from https://people.richland.edu/james/lecture/m170/tbl-chi.html

The vertical label is the degrees of freedom of the Chi-Square distribution, which is found by $k - c$, where k is the number of non-empty bins and c is the number of parameters in the distribution + 1.

The horizontal label is found based on some desired statistical significance level α. The most common choice for significance level in hypothesis testing is 0.05 so I will follow the same convention.

Therefore, the entry in the $k - c$-th row and column 0.05 gives the critical value λ where if χ^2 exceeds λ then the observed data does not follow the theoretical distribution.

Data Collection and Results

I will collect two datasets of solves to use to analyze and model. The first will be the dataset of Yusheng Du's official solve times in competitions taken from worldcubeassociation.org. This dataset will be the main focus of the investigation to look at the legitimacy of the world record. The data selected in this dataset may be influenced by selection bias because the solves happened at a competition, since at a competition, nerves tend to play a big factor. However, adrenaline can also increase performance at a competition on some solves, so it could go both ways. This potential bias cannot be avoided since Yusheng Du does not have a large database of his solves at home recorded, so this is the only available data of Yusheng Du's solves. The dataset consists of 781 solves.

The second dataset will be data of my own solves which I will collect through a Giiker Supercube, which is a Rubik's cube which is connected to my phone via Bluetooth so I can easily record my times as well as track some advanced data such as how many moves I take to solve the cube. I wanted to analyze my own data for a personal component so that I can compare

my own solves with those of Yusheng Du, using the mathematical models I build. The dataset of my own solves consists of 404 solves.

Fig. 2 Giiker Supercube used for recording my solve data

Below is Yusheng Du's data grouped into bins of 2 second intervals, and the frequency of solves within each 2 second interval is displayed. I had to group the bins into 2 second intervals to fit in an outlier time which was in the high 30's unfortunately. I would have preferred to group the bins in 1 second intervals as it would have been a more natural choice to show the number of solves that took a specific number of seconds rather than a range of seconds. This data alone is enough to create a histogram, however the histogram would not be comparable with a probability density function. This is because probability density functions are defined to be normalized so that the total probability is equal to 1, that is for some given probability function $p(x)$:

$$\int_{-\infty}^{\infty} p(x) = 1$$

Geometrically this integral corresponds to the area under the curve of the probability density function. I can then determine the area of the histogram in order to determine the scale factor in order to normalize the histogram so that the area of the histogram is also equal to 1. Since each bin of the histogram is rectangular, I can find each individual bin's area as length times width, and then sum together the area of each bin:

$$\sum_{i=1}^{k} f_i \times w_b \longrightarrow 2\sum_{\mathrm{i=1}}^{\mathrm{k}} f_i \longrightarrow 2\mathrm{n} = 1562$$

In the above equation, k is the number of bins, f_i is the frequency of the i-th bin, $w_b$is the width of the bins, and n is the total population size of the data. Therefore, in order to normalize the histogram and be made comparable to probability density functions, each frequency must be divided by 1562. The density column in the table below displays this normalized data.

Solve Time (s)	Frequency	Density	Solve Time	Frequency	Density
2-4	1	0.000640	16-18	3	0.00192
4-6	3	0.00192	18-20	0	0
6-8	265	0.170	20-22	0	0
8-10	371	0.238	22-24	0	0
10-12	100	0.0640	24-26	0	0
12-14	28	0.0179	…	0	0
14-16	9	0.00576	34-36	1	0.000640

Table 1 Yusheng Du's Solve Times grouped into bins by frequency and density (density rounded to 3 significant figures)

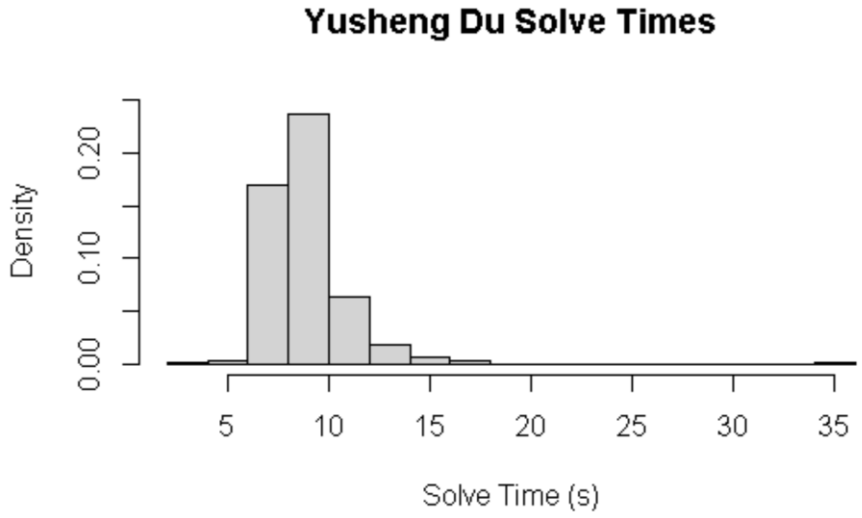

Fig. 3 Yusheng Du's Solve Times plotted as a histogram by density

I repeated the same process with my own data, with the only difference being since I do not have any large outliers, I was able to group my bins into 1 second intervals. This difference in bin size is worrying because it will slightly affect my results later on when doing Chi-Square goodness of fit tests since the value of χ^2 is dependent on the choice of bin size, however this will be accounted for by the fact that the critical value λ will be adjusted by the bin size. For example, if the bin size is larger and causes larger χ^2 values, it will also raise the λ value accordingly.

Solve Time (s)	Frequency	Density	Solve Time	Frequency	Density
7-8	1	0.00248	15-16	46	0.114
8-9	1	0.00248	16-17	17	0.0421
9-10	5	0.0124	17-18	7	0.0173
10-11	18	0.0446	18-19	8	0.0198
11-12	60	0.149	19-20	2	0.00495
12-13	83	0.205	20-21	2	0.00495
13-14	80	0.198	21-22	1	0.00248
14-15	73	0.181			

Table 2 My Solve Times grouped into bins by frequency and density (density rounded to 3 significant figures)

Fig. 4 My Solve Times plotted as a histogram by density

As I anticipated, both histograms show a right/positive skew. The skew is more obvious on the first histogram due to the larger bin size. Another reason that the skew could be more extreme on the first histogram is that towards the left side of the histogram is pushing up against the barrier of what is humanly possible. Since Yusheng Du's mean is near such a strong barrier, the drop off in frequency happens very fast. Meanwhile, my barrier is more at sub-10 seconds, which is a more artificial barrier, so it is easier to get solves *close* to that barrier such as 11 or 12 second solves. Therefore I have a higher frequency of solves close to the barrier and the histogram exhibits less skew behaves closer to a normal distribution.

Building Mathematical Models

Normal Distribution Models

In order to build normal distribution models for the two datasets, I simply took the probability density function describing a normal distribution: $f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2}$ and substituted the mean of the dataset for μ and the standard deviation of the dataset for σ.

For Yusheng Du's solves the model is:

$$N_1(x) = \frac{1}{1.92\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{x-8.81}{1.92}\right)^2}, x > 0$$

The model for my solves is:

$$N_2(x) = \frac{1}{1.94\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{x-13.57}{1.94}\right)^2}, x > 0$$

Where I have restricted the domains of the model because it is impossible to solve a Rubik's cube in zero or less than zero seconds.

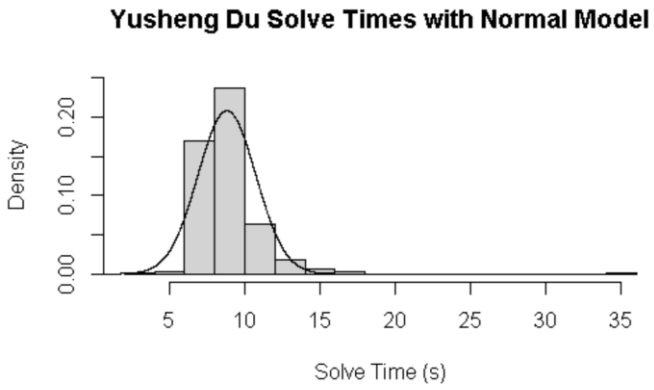

Fig. 5 Yusheng Du solve histogram compared to normal distribution model

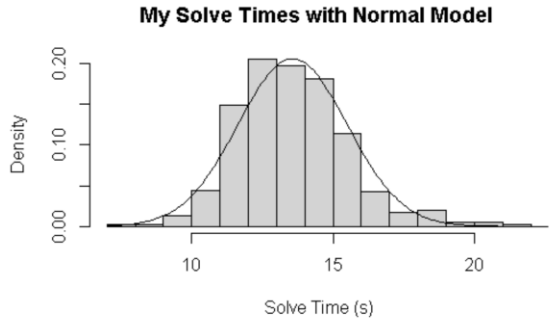

Fig. 6 My solve histogram compared to normal distribution model

Based on observation, the model does not fit the histogram well in Fig. 5 at all. While in Fig. 6 the discrepancy between the model and the histogram is less noticeable, the normal distribution is still not describing the skew in the data.

A good model should make accurate predictions corresponding to the real world. In the real world, it is essentially impossible to get solve under 1 second, however if I use the normal model to predict Yusheng Du's probability of getting a solve under 1 second:

$$P(S_1|N_1) = \int_0^1 \frac{1}{1.92\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{x-8.81}{1.92}\right)^2} dx = 2.17 \times 10^{-5*}$$

*Integrals calculated numerically

Let S_1 represent the event of getting a sub-1 second solve and N_1 the event that the probability follows the probability density function N_1.

In order to gauge probability results of events that are very unlikely, I will be comparing any probabilities I calculate to one of the most unlikely luck-based world records in the world, which was when Patricia Demauro rolled a pair of dice 154 times in a row without rolling a 7 in the game of craps. The probability of that world record happening is $\left(\frac{30}{36}\right)^{154}$ or 6.4×10^{-13}. I will

take any probability of a higher order of magnitude to be plausible as something that could happen if you were to get lucky enough, however if the probability is an even lower order of magnitude, then it will be deemed nearly impossible to get that lucky. This comparison is not perfect since Rubik's Cube speed solving is not entirely luck based, however it will serve as a good benchmark to see how low of odds is too low such that the event would never actually happen in real life. As you can see, the normal distribution model predicts a probability for Yusheng Du getting a sub-1 second solve that is 8 orders of magnitude higher than the benchmark I have set. That means using the normal model, it is predicted that Yusheng Du could plausibly get a solve that is faster than one second when in reality the probability of that happening is essentially zero.

Repeating the same calculations but with the model for my data gives:

$$P(S_1|N_2) = 4.14 \times 10^{-11}$$

Therefore, the normal distribution model predicts that even I could theoretically get a solve under 1 second, since the probability is still 2 whole orders of magnitude higher than the benchmark I set of 10^{-13}.

These two predictions from the model about the odds of Yusheng Du and I getting sub-1 second solves respectively are big red flags, as I would expect the model to predict odds that are many orders of magnitude lower than the benchmark for an event that is just about physically impossible to accomplish.

Intuitively, the normal distribution does not seem like a good choice to model the two datasets I have. Furthermore, it can be seen visually that the model does not fit the data very well. However, my biggest issue with the normal distribution models comes when considering the predictions which the model makes. Since the model makes inaccurate and unrealistic predictions compared to the real-world phenomena, it leads me to believe I need to find better probability distributions to model the data.

Although I have already ruled out the normal distribution as a good model, I would still like to use Chi-Square goodness of fit testing to quantify how well the models fit the data so that I can quantitatively compare between two models how well they fit the data.

Log-normal Distribution Models

As I discussed in the introduction, the μ and σ parameters for a log-normal distribution are not the μ and σ of the dataset itself. Instead, we need to find the right μ and σ parameters that create the log-normal distribution which best fits the data. In order to do this, I will use the method of Maximum Likelihood Estimation, or MLE for short. An important concept for this method is the Likelihood function, represented by $\mathcal{L}(x_i|\mu, \sigma)$ represents the likelihood of observing some data point x_i given the random variable follows the probability distribution described by μ and σ. The likelihood function is simply the value of the probability density function with x_i plugged in:

$$\mathcal{L}(\mathrm{x_i}|\mu,\sigma) = \frac{1}{\mathrm{x_i}\sigma\sqrt{2\pi}}\mathrm{e}^{-\frac{(\ln \mathrm{x_i}-\mu)^2}{2\sigma^2}}$$

Since each data point was collected by a Rubik's Cube solve which is an independent event, I can multiply together the likelihood function for each individual data point to find an expression for the likelihood of the entire dataset being produced given the underlying probability density function is described by the log-normal distribution specified by some μ and σ. I will represent the likelihood function of the entire dataset as $\mathcal{L}_T$:

$$\mathcal{L}_T(x_i|\mu,\sigma) = \prod_{i=1}^{n}\mathcal{L}(x_i|\mu,\sigma) = \prod_{i=1}^{n}\frac{1}{\mathrm{x_i}\sigma\sqrt{2\pi}}\mathrm{e}^{-\frac{(\ln \mathrm{x_i}-\mu)^2}{2\sigma^2}}$$

Now we want to find the values of parameters μ and σ which maximize the likelihood of the entire dataset, that is:

$$\frac{\partial}{\partial\mu}\left(\prod_{i=1}^{n}\frac{1}{\mathrm{x_i}\sigma\sqrt{2\pi}}\mathrm{e}^{-\frac{(\ln \mathrm{x_i}-\mu)^2}{2\sigma^2}}\right) = 0$$

$$\frac{\partial}{\partial\sigma}\left(\prod_{i=1}^{n}\frac{1}{\mathrm{x_i}\sigma\sqrt{2\pi}}\mathrm{e}^{-\frac{(\ln \mathrm{x_i}-\mu)^2}{2\sigma^2}}\right) = 0$$

These expressions are difficult to differentiate since there is a multiplication sum present in both of them, which means in order to differentiate them you would need to use product rule repeatedly. In order to simplify these expressions to maximize them I had to find something known as the log-likelihood. I will prove that I can do this by showing that if any arbitrary function $f(x)$ is at a maximum, then its natural logarithm $ln[f(x)]$ is also at a maximum, assuming that $f(x)$ is a continuous function with range {y|y>0} so that $ln[f(x)]$ is never undefined.

Firstly, if $f(x)$ is at a maximum, then it's derivative $f'(x)$ must equal 0. Using chain rule, the derivative of $ln[f(x)]$ is $\frac{f'(x)}{f(x)}$. Since I know that $f'(x)$ is 0, $\frac{f'(x)}{f(x)}$ is also 0. I have shown thus far that if the derivative of $f(x)$ is 0, the derivative of $ln[f(x)]$ is also 0, but for a function to be maximized, the second derivative also needs to be negative. The second derivative of $ln[f(x)]$ is the derivative of $\frac{f'(x)}{f(x)}$, which using quotient rule is $\frac{f''(x)f(x)-f'(x)^2}{f(x)^2}$. Since $f'(x)$ is 0, this expression simplifies to $\frac{f''(x)}{f(x)}$. Because I restricted the range of our function $f(x)$ so that it can only be positive, if the second derivative of the function itself $f''(x)$ is negative, the second derivative of the natural log of the function $\frac{d^2}{dx^2}(ln[f(x)])$ is also negative. Thus, the proof is complete since I have shown that assuming $f(x)$ is at a maximum, we can deduce that $ln[f(x)]$ is also at a maximum, given the restriction in the range of $f(x)$. As it turns out, any likelihood function has just this restriction in range, since a likelihood cannot be ≤ 0. Therefore, I can take the natural logarithm of the likelihood function and then maximize it and get the same answer.

This is known as a log-likelihood, and it is useful because when I take the natural logarithm of the expression of all the likelihood functions being multiplied together, using the property of logarithms $ln(xy) = ln(x) + ln(y)$, I can convert the multiplication sum into an addition sum:

$$ln\left(\prod_{i=1}^{n} \frac{1}{x\sigma\sqrt{2\pi}} e^{-\frac{(\ln x_i - \mu)^2}{2\sigma^2}}\right) = \sum_{i=1}^{n} ln\left(\frac{1}{x_i\sigma\sqrt{2\pi}} e^{-\frac{(\ln x_i - \mu)^2}{2\sigma^2}}\right)$$

Simplifying the summation some more using logarithm rules:

$$= \sum_{i=1}^{n} -ln\left(x_i\sigma\sqrt{2\pi}\right) - \frac{(ln\, x_i - \mu)^2}{2\sigma^2}$$

Finally, I can once again set the derivatives with respect to μ and σ to zero to find the values which maximize the log-likelihood:

$$\sum_{i=1}^{n} \frac{\partial}{\partial\mu}\left[-ln\left(x_i\sigma\sqrt{2\pi}\right) - \frac{(ln\, x_i - \mu)^2}{2\sigma^2}\right] = 0$$

$$\sum_{i=1}^{n} \frac{\partial}{\partial\sigma}\left[-ln\left(x_i\sigma\sqrt{2\pi}\right) - \frac{(ln\, x_i - \mu)^2}{2\sigma^2}\right] = 0$$

First, I will start by maximizing μ:

$$\sum_{i=1}^{n} \frac{\partial}{\partial\mu}\left[-ln\left(x_i\sigma\sqrt{2\pi}\right) - \frac{(ln\, x_i - \mu)^2}{2\sigma^2}\right] = \sum_{i=1}^{n} \frac{(ln\, x_i - \mu)}{\sigma^2} = 0$$

I can then take σ^2 out of the summation and multiply both sides of the equation by it to get:

$$\sum_{i=1}^{n} ln\, x_i - \mu = 0$$

This expression can then be solved numerically for μ.

Next, I will find an expression to find the value of σ which maximizes the log-likelihood:

$$\sum_{i=1}^{n} \frac{\partial}{\partial\sigma}\left[-ln\left(x_i\sigma\sqrt{2\pi}\right) - \frac{(ln\, x_i - \mu)^2}{2\sigma^2}\right] = \sum_{i=1}^{n} \frac{(ln\, x_i - \mu)^2 - \sigma^2}{\sigma^3} = 0$$

After finding μ from the previous equation it can be substituted into this expression and solved numerically to find σ.

I used the python program below to solve the expressions for μ and σ of the Yusheng Du dataset:

```
import numpy as np
from scipy.optimize import fsolve
import pandas as pd
import io
from google.colab import files
uploaded = files.upload()
data = pd.read_csv(io.BytesIO(uploaded['YushengDuWCATimes - Copy.csv']))      # uploading excel data
data_numpy = data.values                                                      # data_numpy is the variable which stores the array of data read from the excel file of solve times
                                                                              # it is the x_i value being summed over


def f(x):                                                                     # defining the function for the sum expression to solve for mu *x represents mu
  sum2 = np.sum(np.log(data_numpy)-x)
  return sum2


def g(y):                                                                     # defining the function for the sum expression to solve for sigma *y represents sigma
  sum3 =np.sum(((np.log(data_numpy)-2.157172)**2-y**2)/y**3)
  return sum3

ans = fsolve(f,0)                                                             # fsolve is a capability in python which solves a function for a zero point numerically
print(ans)                                                                    # using a method called bisection which takes a guess point and searches for zeroes around that point
ans2 = fsolve(g,0.1)
print(ans2)                                                                   # ans solve for mu ans2 solves for sigma and then they are both printed so we can see their values
```

```
Choose Files  No file chosen    Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.
Saving YushengDuWCATimes - Copy.csv to YushengDuWCATimes - Copy.csv
[2.15717234]
[0.18619884]
/usr/local/lib/python3.7/dist-packages/scipy/optimize/minpack.py:162: RuntimeWarning: The iteration is not making good progress, as measured by the
  improvement from the last ten iterations.
  warnings.warn(msg, RuntimeWarning)
```

The same program was used to find μ and σ for the dataset of my solves. All that was changed was uploading a different excel file with the different dataset.

Using this program to find the log-normal distribution which maximizes the likelihood of observing the Yusheng Du dataset, I found that μ = 2.15 and σ = 0.186 (rounded to 3 sig. figs).

Applying the same process to the dataset of my solves, I found that μ = 2.60 and σ = 0.141. Thus, the following functions ℓ_1 and ℓ_2 are the log-normal models for the Yusheng Du dataset and my dataset respectively:

$$\ell_1(x) = \frac{1}{x(0.186)\sqrt{2\pi}} e^{-\frac{(lnx-2.15)^2}{2(0.186)^2}}$$

$$\ell_2(x) = \frac{1}{x(0.141)\sqrt{2\pi}} e^{-\frac{(lnx-2.60)^2}{2(0.141)^2}}$$

Unlike the normal distribution models, I do not need to impose the restriction on the domain of the function $x > 0$, since log-normal distributions naturally have that restriction of domain, as $ln(x)$ is not defined for $x \leq 0$.

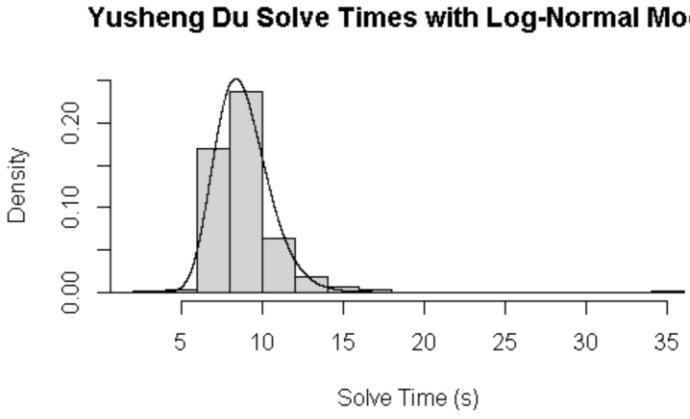

Fig. 7 Yusheng Du solve histogram compared to log-normal distribution model

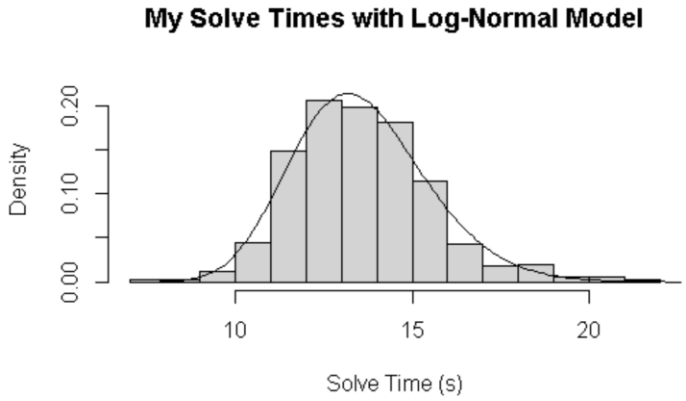

Fig. 8 My solve histogram compared to log-normal distribution model

I can immediately see visually that these log-normal distribution models are doing a better job of describing the right-skewed nature of the data. The fit looks very good, however now I am going to judge whether the predictions the model makes are more reasonable by looking at the predicted value of the probability of getting a sub-1 second solve using the log-normal model for Yusheng Du's data:

$$P(S_1|\ell_1) = \int_0^1 \frac{1}{x(0.186)\sqrt{2\pi}} e^{-\frac{(lnx-2.15)^2}{2(0.186)^2}} dx = 3.32 \times 10^{-31}$$

Recalling the benchmark I set earlier where any probability smaller than the 10^{-13} order of magnitude was considered virtually impossible, this theoretical probability is nearly 20 orders of magnitude smaller, therefore this model predicts that the odds of getting a solve faster than one second are vanishingly slim, which accurately reflects the odds of getting a solve that fast in the real world. I won't need to check the probability of getting a solve under 1 second using the model for my data since it will just be an even smaller number. Therefore it seems like the log-normal distribution is the best distribution for modelling the behavior of Rubik's Cube solves.

Weibull Distribution Models

I will be using the same method of Maximum Likelihood Estimation in order to find the optimal parameters μ, γ, and α that best fit the data. In my introductory discussion around the Weibull Distribution, I realized that the location parameter μ describes some failure-free time period, or in this context some initial time period where it is impossible to solve the Rubik's cube that fast. As I was conducting calculations however, I came across the problem that when trying to find the values of all 3 parameters using the MLE method, there is no proper solution. In order to find a solution, the μ parameter needs to have a set value. This is unfortunate because I will have to arbitrarily choose the μ parameter for each model. I decided to set the μ parameter to 3 for the Yusheng Du dataset, since I do not believe he could get a solve faster than 3 seconds since for

his 3.47 second solve he already had just about the luckiest and best solve he could have. For the dataset of my own solves, I decided to set the μ parameter to 6. While my personal best solve is only 7.37 seconds, since I tracked some of my advanced data while gathering my solve data, I was able to come up with a best-case scenario. The lowest move solution I had was 47 moves, and the fastest turn speed I had during my 404 solves was 7.77 turns per second. In the best-case scenario where my solve is 47 moves long and I turn as fast as I can at 7.77 turns per second, this would lead to a 6.05 second solve. Therefore, I believe the absolute best solve I could get would be 6 seconds.

To perform the MLE method, I will first set up the total likelihood function for an entire dataset for the Weibull Distribution. Before I do that, since it is more convenient to work with a 2-parameter Weibull distribution, I will define a new dataset $\beta_i \equiv x_i - \mu$ in order to eliminate μ since it has already been set to a constant.

$$\mathcal{L}_T(\beta_i|\gamma,\alpha) = \prod_{i=1}^{n} \frac{\gamma}{\alpha}\left(\frac{\beta_i}{\alpha}\right)^{\gamma-1} e^{-\left(\frac{\beta_i}{\alpha}\right)^{\gamma}}$$

The log-likelihood is then:

$$ln[\mathcal{L}_T(\beta_i|\gamma,\alpha)] = \sum_{i=1}^{n} ln\left(\frac{\gamma}{\alpha}\left(\frac{\beta_i}{\alpha}\right)^{\gamma-1} e^{-\left(\frac{\beta_i}{\alpha}\right)^{\gamma}}\right)$$

The process of finding the values of γ and α which maximize the log-likelihood function would continue on from here, but it is very long and contains lots of expanding and simplifying which I will not include in this IA. Instead, I will use a computer program to maximize the log-likelihood functions numerically using a statistics program called RStudio.

```
library("readxl")
library("survival")
dat<- read_excel("C:/Users/[redacted]/OneDrive/Desktop/YushengDuWCATimes.xlsx")


Solves<-dat$Solves
Solves2<-dat$Solves -3

hist(Solves, freq=F, xlab="Solve Time (s)", main="Yusheng Du Solve Times with Weibull Model",ylim=c(0,0.25))
histdat=hist(Solves,plot=F)

status=rep(1,length(Solves2))
fit1 = survreg(Surv(Solves2,status)~1,dist = "weibull")         # survreg is a function in RStudio which allows you to find the coefficients
                                                                # to fit a various amount of statistical distributions such as the weibull distribution


alpha = exp(fit1$coefficients)                                  # survreg uses an alternative version of the weibull distribution where the coefficients are
gamma = 1/fit1$scale                                            # different and need to be converted in order to get the coefficients alpha and gamma we need
x=seq(0,35,0.1)

y=((gamma/alpha)*((x-3)/alpha)^(gamma-1)*exp(-((x-3)/alpha)^gamma)) #graphs the expression for the weibull distribution
lines(x,y)
```

The same program was used while making some adjustments to find parameters for the dataset of my solves.

For the Yusheng Du dataset, the values of each parameter were found to be: μ = 3, $\gamma = 2.71$, and $\alpha = 6.43$. For my dataset, the values of each parameter were found to be: μ = 6, $\gamma = 4.01$, and $\alpha = 8.31$. These constants yield the following probability density functions:

$$W_1(x) = \frac{2.71}{6.43}\left(\frac{x-3}{6.43}\right)^{1.71} e^{-\left(\frac{x-3}{6.43}\right)^{2.71}}$$

$$W_2(x) = \frac{4.01}{8.31}\left(\frac{x-6}{8.31}\right)^{3.01} e^{-\left(\frac{x-6}{8.31}\right)^{4.01}}$$

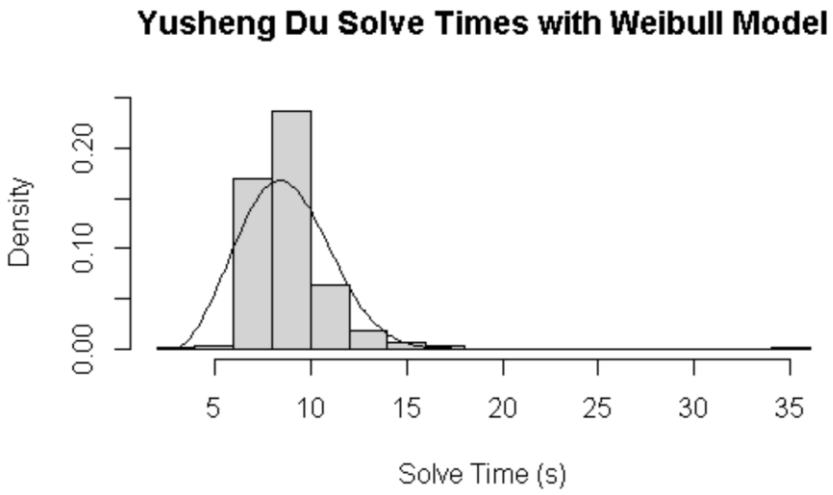

Fig. 7 Yusheng Du solve histogram compared to Weibull distribution model

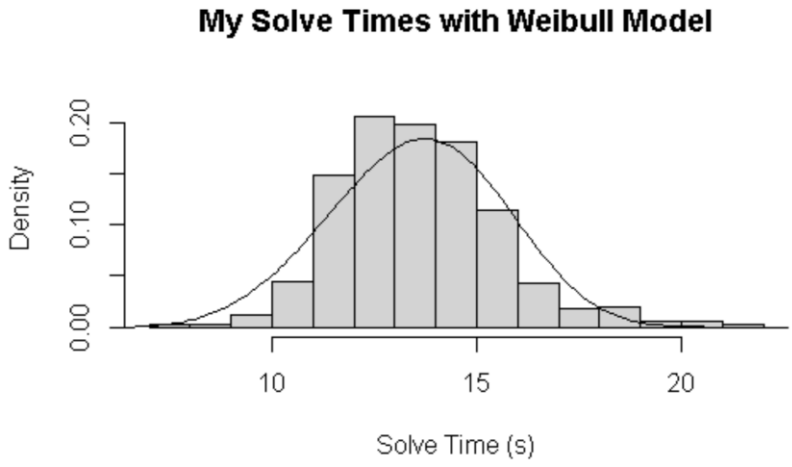

Fig. 7 My solve histogram compared to Weibull distribution model

From looking at the graphs, I can see that the Weibull models come nowhere close to matching the actual data. This really surprised me for a couple of reasons. Firstly, I thought that the ability to model a failure-free time period would allow for a model which described the data more accurately. Secondly, I thought that there would be a very good fit created considering there were three separate parameters that were maximized to try to create the best fit possible. I

assumed that the behavior of Rubik's Cube solve times would be analogous to the typical "time to failure" behavior a Weibull distribution is used to model, however it seems this was not the case. Since it can be visually seen how poorly the Weibull distribution fits the data, I will not be looking at the predictions it makes or its χ^2 values to judge whether it is a viable model as it will not be necessary.

Analysis and Conclusion

In order to judge how well each model fits compared to the data, I will be using the Chi-square goodness of fit test and comparing it to a critical value found from a Chi-square probability table. For the normal distribution models, in order to find the Chi-squared test statistic I will need to find the values for the expected frequency E_i in the expression: $\chi^2 = \sum_{i=1}^{k}(O_i - E_i)^2/E_i$.

In order to find E_i from the equation $E_i = n\big(F(Y_u) - F(Y_l)\big)$, I will need to use the cumulative distribution function of the models from the probability density functions. The cumulative distribution function (cdf) for a normal distribution is: $\frac{1}{2}\left[1 + erf\ (\frac{x-\mu}{\sigma\sqrt{2}})\right]$, where erf is the error function defined as $erf\ (z) = \frac{2}{\pi}\int_0^z e^{-t^2}dt$.

I will include a table showing the calculations of χ^2 between Yusheng Du's solve data and a log-normal distribution model in order to illustrate the process. This specific calculation is also a good example to illustrate some issues I encountered calculating Chi-Square values which I will be discussing later: The calculations will be shown in the table below rounded to 3 sig. figs:

Solve Time (s)	Observed Frequency (O_i)	Value of cdf at lower limit ($F(Y_l)$)	Value of cdf at upper limit ($F(Y_u)$)	Expected Frequency (E_i)	χ_i^2*
2-4	1	2.39×10^{-15}	2.01×10^{-5}	0.0157	61.6
4-6	3	2.01×10^{-5}	0.0271	21.1	15.5
6-8	265	0.0271	0.352	254	0.480
8-10	371	0.352	0.794	345	1.96
10-12	100	0.794	0.964	133	8.12
12-14	28	0.964	0.996	24.7	0.445
14-16	9	0.996	~1.00**	3.02	11.8
16-18	3	~1.00	~1.00	0.292	25.1
18-20	0	~1.00	~1.00	~0	0
20-22	0	~1.00	~1.00	~0	0
22-24	0	~1.00	~1.00	~0	0
24-26	0	~1.00	~1.00	~0	0
…	0	~1.00	~1.00	~0	0
34-36	1	~1.00	~1.00	~0	***

Table 3 Calculations of Chi-Square value for Yusheng Du's data and log-normal model (rounded to 3 significant figures)

*χ_i^2 represents the value of the i-th term in the summation for χ^2

**actual value was 0.99959…

*** The value of this number would be incredibly large as for: $y = \lim_{x \to 0} \frac{(1-x)^2}{x}, y \to \infty$

Using the formulas above, χ^2 was found to be 252, which is very high. To find the entry in the Chi-Square probability table, I need to first find $k - c$. k is the number of non-empty bins, of which there are 9. c is the number of parameters in the distribution + 1, so since there are two parameters in a normal distribution μ and σ, c = 3. Therefore $k - c$ = 6, and I am looking for the 6th row and column labelled 0.05 for the critical value. The critical value λ rounded to 3 sig. figs is 12.6. $\chi^2 > \lambda$, therefore, Yusheng Du's solve data does not follow the normal distribution model.

Repeating the same calculations for comparing the dataset of my solves to the normal distribution model, I found a value of $\chi^2 = 86.3$. Using the same method to find the critical value, I found that $\lambda = 21.0$. Therefore, once again $\chi^2 > \lambda$, so the dataset of my solves also does not follow a normal distribution.

Now I will calculate the χ^2 values for each of the two log-normal models. The only thing which changes for this calculation is the way in which I will find the expected frequency E_i, as for log-normal models I will use the cdf for a log-normal distribution. The cdf for a log-normal distribution is: $\frac{1}{2} + \frac{1}{2}\ erf\ \left[\frac{lnx - \mu}{\sqrt{2}\sigma}\right]$.

The Chi-Square value for Yusheng Du's solve data fitted to a log-normal distribution was found to be $\chi^2 = 125$. The critical value λ is the same for the log-normal distribution as it is for the normal distribution since there are an equal number of parameters, therefore $\lambda = 12.6$. Once again $\chi^2 > \lambda$, however the chi-square value for the log-normal distribution is half of what it was for the normal model.

The Chi-Square value for my solve data fitted to a log-normal distribution was found to be $\chi^2 = 35.3$. This Chi-Square value is the lowest I have found so far, yet it is still greater than the critical value needed, $\lambda = 21.0$.

I have noticed that for every Chi-Square test which I calculated, the Chi-Square values tend to get artificially driven up a lot by the discrepancies between the observed frequency and expected frequency towards the two extremes of distributions where the expected frequency starts to tail off and become very low. This was particularly true for the log-normal models, where all the Chi-Square values in higher frequency areas showed that the observed and expected frequencies almost perfectly matched each other. This phenomenon of artificially inflated Chi-Square values towards the tails of the distributions is likely caused by the sample sizes of my datasets being too small. This effect can especially be seen in the example of the calculations I provided in Table 3, where the outlier values are contributing to the majority of the Chi-Square value. This makes sense since Yusheng Du getting a 3.47 second solve was a very unlikely occurrence as the log-normal model would only expect 0.0157 solves to be in-between 2 and 4 seconds in a 781 solve sample. The problem is that experimentally 1 out of 781 solves were between 2 and 4 seconds. Obviously, the model is more accurate, since the real probability of Yusheng Du getting a solve

between 2 and 4 seconds is far below 1/781, however this leads to an extra 61.6 being added to the Chi-Square value.

Since the Chi-Square value for every single model was over the critical value it needed to pass to be considered a good fit according to the Chi-Square goodness of fit test, I will have to make some qualitative judgements.

I believe that the log-normal model fits for both datasets well based on observing the graphs as well as the fact that the Chi-Square values for the log-normal models were significantly lower than those for the normal models. Secondly, the predictions that the log-normal models make seem to properly model the real-life situation they are meant to describe based on my intuition. Thirdly, the reason that the Chi-Square values exceed the critical values can be explained as an issue caused by limited sample sizes in the data. For these reasons, I think it can be justified that both sets of Rubik's Cube data follow a log-normal distribution.

Now that I have justified that both sets of Rubik's Cube solve data can be accurately modelled using the log-normal models I created, I can find results from the models which I stated I was interested in back in the introduction. First, I will look at the probability of Yusheng Du solving a Rubik's cube in 3.47 seconds or faster. Earlier I found these probabilities by integrating the probability density function from 0 to n, but this is really the same as evaluating the cumulative distribution function at n:

$$\frac{1}{2}+\frac{1}{2}\,erf\left[\frac{ln(3.47)-2.15}{0.186\sqrt{2}}\right]=5.58\times10^{-7}$$

While this world record was certainly very lucky, recalling the benchmark of 10^{-13}, it is certainly a reasonable amount of luck. Therefore, based on my model, Yusheng Du's 3.47 second solve was not too lucky to be true.

Now just for fun, I will look at the predicted probability I have of breaking the world record:

$$\frac{1}{2}+\frac{1}{2}\,erf\left[\frac{ln(3.47)-2.60}{0.141\sqrt{2}}\right]=3.42\times10^{-22}$$

It is quite obvious that since the probability predicted is 9 orders of magnitude smaller than the benchmark, it is practically impossible for me to break the world record at my current Rubik's cube skill level.

Evaluation and Extensions

- Overall, I think I was able to come up with good models for my data that made accurate predictions so in that sense it was a success
- I had a lot of fun with this project and learned a lot. It was easier to connect with the math behind these concepts in probability and statistics since it was in the context of Rubik's cubes, something familiar and enjoyable to me.
- As an aspiring physicist, I definitely gained valuable knowledge from this experience. The two distributions I learned about will likely have applications in physics, and in

general any knowledge about probability and statistics will have applications in physics to fields such as statistical mechanics, quantum mechanics, condensed matter physics etc.

- I concluded that Rubik's cube solves tend to be distributed like a log-normal distribution, however I did not fully understand why that might be the case besides my intuition about why the data would be skewed. During my research, I encountered something known as Gibrat's law, which supposedly gives the reasoning behind why certain behaviors lead to data following a log-normal distribution. Unfortunately, I was not able to learn enough about probability and statistics in order to get a good understanding of Gibrat's Law. In the future I look forward to learning more about this topic.
- The thing I would like to improve on the most is the Chi-Square goodness of fit testing. Since the issue was created by data points towards the tail ends of distributions, I could consider removing outliers as a way to conduct more accurate Chi-Square testing. The problem with that is that removing outliers is generally a controversial statistical practice, and I would have to question whether I should remove outliers before creating the models as well.
- Similarly to do with the Chi-Square testing, the Chi-Square test is dependent on the choice of bin width/number of bins. If I had more insight into the topic of how the choice of bins affects the results, perhaps I would have chosen a more optimal bin width.
- While Chi-Square testing is dependent on an arbitrary choice in bin width, there are some ways of testing the goodness of fit which are not dependent on this such as a Q-Q plot (quantile-quantile plot). I do not fully understand Q-Q plots, however I would consider using them in the future.

References

Lognormal Distribution. (n.d.). Retrieved April 23, 2021, from https://www.itl.nist.gov/div898/handbook/eda/section3/eda3669.htm

Log Normal Distribution. (n.d.). Retrieved from https://mathworld.wolfram.com/LogNormalDistribution.html

Weibull Distribution. (n.d.). Retrieved from https://www.itl.nist.gov/div898/handbook/eda/section3/eda3668.htm

Discussion of Maximum Likelihood Estimation for the 3-Parameter Weibull Distribution. (2013, June). Retrieved April 23, 2021, from https://www.weibull.com/hotwire/issue148/hottopics148.htm

Jones, J., (n.d.). Statistics: Lecture Notes. Retrieved April 23, 2021, from https://people.richland.edu/james/lecture/m170/tbl-chi.html

Chi-Square Goodness-of-Fit Test. (n.d.). Retrieved April 23, 2021, from https://www.itl.nist.gov/div898/handbook/eda/section3/eda35f.htm

1.2 - Maximum Likelihood Estimation: STAT 415. (n.d.). Retrieved from https://online.stat.psu.edu/stat415/lesson/1/1.2

Craps player sets record roll at Borgata casino. (2009, May 24). Retrieved from https://www.nj.com/news/2009/05/craps_player_sets_record_roll.html

Yusheng Du (杜宇生): World Cube Association. (n.d.). Retrieved April 23, 2021, from https://www.worldcubeassociation.org/persons/2015DUYU01