STOCKHOLM SCHOOL OF ECONOMICS
Department of Economics
5350 - Master's Thesis in Economics
Academic Year 2023-2024

A Simulated Approach to Agreeability of the Gini Coefficient and the Variance of Logarithms

Jack Dickinson (42435)

Abstract: This paper investigates the potential for researcher influence when selecting a specific measure of inequality. By using large amounts of simulated data following a one-parameter Weibull distribution, the paper employs two primary methods for investigating the agreement between how the Gini coefficient and the variance of logarithms rank distributions. We find that there are noticeable differences in rankings between the two measurement types, as well as internal inconsistency for the variance of logarithms.

Keywords: Inequality, Gini Coefficient, Variance of Logarithms, Researcher Influence
JEL Classification: B41, C15, D63

Supervisor: Jonathan Lehne
Date submitted: 13 May 2024
Date Examined: 24 May 2024
Discussants: Agnes Durbeej-Hjalt
Examiner: Magnus Johannesson

Acknowledgments

There are far too many people I would like to thank, but I will do my best. I could not have done any of this without the support of love of my parents, Lori and Dave. They mean more to me than they will ever know. I want to thank my grandparents, Richard and Jean Heckmann, for their support and for their constant reminder of the decency of human beings. I want to thank Michael Jerman and Fabio Blasutto for providing very early feedback at the idea stage of this paper. I want to thank Karin Hederos for revitalizing my excitement for economics and the research process as a whole. I especially want to thank my advisor Johathan Lehne for helping me in this process and allowing me to take a gamble on an unconventional topic for a thesis. Finally, I want to thank my cohort at SSE. They are a collection of some of the most talented and thoughtful people I have ever met and I would not have made it this far without them.

Contents

1 Introduction **3**

2 Measurement **4**

3 Literature Review **5**

4 Simulation **6**
4.1 Pairwise Comparison 6
4.2 Ranked Comparison 8
4.3 Distribution Type 9

5 Results **10**

6 Discussion **14**
6.1 Discussion of Results 14
6.2 Shape Parameter 16
6.3 Limitations 16
6.4 Future Work 18

7 Conclusion **19**

A Appendix A: Graphs **23**
A.1 Shape Parameter Changes 23
A.2 Robustness of Normal Agreement Rate 24

B Appendix B: Summary tables **24**

C Appendix C: Code **29**
C.1 "Pairwise" 29
C.2 "Agreement Graphs" 33
C.3 "Normal Robustness" 34
C.4 "Weibull Agreement Graph" 35
C.5 "Weibull Multi" 36
C.6 "Graph Weibull Multi" 42
C.7 "Histogram Gaps" 46

1 Introduction

The study of economic inequality is huge part of academic research is our field, and the effort to reduce global inequality is an important social issue. Reduced inequality represents one of the 17 sustainable development goals. [1] According to the UN, economic inequality is still rising within many developed countries, and gaps between countries continue to grow. [19] Despite the long standing relevance as a topic, there are still questions being asked about how best to understand and frame this issue.

As far as methodological issues go, the most researched question at the moment relates to what type of inequality should be measured. The three primary metrics used are income, wealth, and consumption. At a surface level, these metrics are similar: income is needed to fuel consumption and unused consumption is how wealth begins to generate. But there is clearly more going on than this simplification. In fact, many papers that have analysed two or all three of these channels have found differing results. Anghel et. al find wealth inequality responding in a different way than income or consumption when analysing household behavior in Spain following the 2008 financial crisis [3]. Brzozowski et. al find large increases in income inequality but minimal increases in wealth and consumption inequality over the same time period [7].

This paper attempts to examine another area of methodological ambiguity: the choice of instrument used to measure inequality. The goal here is to try and contribute to the growing literature of meta-science in economics by examining a potential source of researcher influence. [15] The motivation is simple: in a lot of academic work on economic inequality, the choice of measurement used is unjustified. In his 1970 paper "On the measurement of inequality", author Atkinson notes that "the conventional approach in nearly all empirical work is to adopt some summary statistic of inequality such as the variance, the coefficient of variation or the Gini coefficient with no very explicit reason being given for preferring one measure rather than another." [4] The problem is if there are large enough discrepancies between the outcomes for various measurement types, then that introduces a possible channel for the results to be swayed by factors outside the identification. Even if the choice is explained by the researchers, the presence of differences between how inequality is measured could still pose a problem for properly understanding the results in the context of other academic work.

We attempt to investigate this topic by using simulated income distributions and two approaches to analyze how the Gini coefficient and the variance of logarithms give results for the same sets of data. While the analysis of measurement types is not a novel research topic, the literature is still limited and this paper provides, as far as the author can tell, a completely unique approach by simulating large amounts of data to try and uncover patterns that may appear in real research.

The paper is organized in the following way. In section 2 we establish the measurement types this paper will explore. Section 3 briefly summarises important literature. Section 4 dives into the details of the simulation approach used to answer some questions. Section 5 presents results and section 6 analyzes those results and discusses the limitations and areas for further work. Finally, section 7 concludes.

2 Measurement

Economists have no shortage of ways to measure inequality. Attanasio and Pistaferri (2016) use the variance of logs to measure consumption inequality in the United States. [5] Coibion et. al (2021) use a coefficient of variation to look at the relationship between shopping habits and consumption inequality. [9] Aguilar and Bils (2015) look at differences between the ratio of spending on luxuries and necessities for high and low income households. [2] Xie and Zhou (2014) use the Gini coefficient for analyzing income inequality in China. [22]
We consider tools for measuring inequality to be divided into two broad categories. The first, which we refer to as "holistic measurements," use the entirety of a distribution of incomes to generate a value. An example of this kind of measurement would be the Gini coefficient, which subtracts every value in a distribution from every other value to generate an inequality index. The second, which we refer to as "sectioned measurements," only use some values in the distribution. One example of a sectioned measurement would be a 90-10 income ratio, where the income of the 90th percentile earner is divided by the income of the 10th percentile earner.

It would not be surprising to see two measurements from different categories disagreeing to some extent. A 90-10 income ratio fails to capture anything about the middle of the distribution, which a holistic measurement would be affected by. In fact, the Gini Coefficient, the most commonly used measure of inequality, has been shown to be more sensitive to changes in the middle of the distribution [12]. We might also expect that two sectioned measurements might fail to agree some of the time. Even changing the percentiles of the income ratio to 99-1 might pick up a different effect - perhaps one country has an extremely wealthy elite class but low inequality among those outside the highest percentile.

It is for those reasons that this paper instead chooses to look at the relationship between two holistic measurements. When comparing these two, no longer is some of the data being excluded. Two holistic measurements will take in and use the exact same data. If the results coming out are different, that might pose a problem in the form of researcher influence. The motivation for this analysis comes from the general lack of acknowledgement behind measurement choice in inequality research.

The two measurements focused on in this paper are the Gini coefficient and the variance of logs. The Gini coefficient was formulated over a century ago [8], and is the most commonly used measure of inequality [12]. If holistic measurements are to be examined in this paper, the Gini must be one of them. The variance of logs is another measurement often used, and was selected to serve as a more "classically mathematical" measurement.

3 Literature Review

When it comes to the discussion of measurement types for inequality research, the seminal papers on the topic are Dalton (1920) [10] and Atkinson (1970). Dalton's first paragraph contains the line "it is not generally agreed how this inequality should be measured", echoing a similar sentiment to the inspiration for this paper over a century ago. Dalton notes that there is no way for a single inequality measurement to be objectively correct, since it requires assumptions about the relationship between welfare and income given that "inequality ... may be defined in terms of economic welfare, [but] must be measured in terms of income." Dalton's most lasting contribution is the introduction of the "principal of transfers." The principal of transfers states that when income is transferred from a relatively poor individual to a relatively rich individual, inequality must increase. Today, this principal of transfers in included in the Lorenz criterion. [20]

Atkinson's paper serves as the foundation of the modern approach to measurement questions. Atkinson compared the behavior of five inequality measurements: variance, coefficient of variation, relative mean deviation, Gini index, and standard deviation of logarithms. Atkinson notes some structural disparity between these measurements. For example, it is shown that the relative mean deviation is not sensitive to transfers on the same side of the mean, a violation of the Lorenz criterion. Atkinson also showcases at the end of his paper how rankings can deviate depending on which measurement type is used. When looking at the difference between the rankings for the Gini coefficient and the standard deviation of logarithms[1] we do see differences. For example, out of the twelve countries examined, India is tied for 8th in equality under the Gini coefficient but 3rd under the standard deviation of logs, while Sweden is tied for 6th under the Gini but 11th under the standard deviation of logarithms.

The work in this paper strongly builds on the findings of Foster and Ok (1999). [11]. In that paper, the authors look to answer the question of whether the variance of logs follows the Lorenz criterion, which they call "a ranking generally accepted as the unambiguous arbiter of inequality comparisons." The Foster and Ok paper employs rigorous mathematical proofs to show that there are

[1] this ranking would be identical to the variance of logarithms

problems with the variance of logs as an inequality measurement. This paper does not do that, instead employing an alternate approach to try and answer a question posed in the concluding remarks of the Foster and Ok paper: "how likely are these errors to arise in practice?" - a question which Foster and Ok state they do not have a definitive answer to. This paper leverages the vastly increased computing power available to us in 2024 to try and help answer this question. An important difference between this paper and Foster and Ok is that this paper looks at discrepancies between multiple distributions, while Foster and Ok look at the behavior of transfers within distributions[2].

Unfortunately, despite the important results in Foster and Ok, the paper is rarely cited. Morduch and Sicular (2002) examine inequality decompositions and find disagreement between the Gini and Thiel indices when trying to account for how factors like regional segmentation, human capital accumulation, and political variables could explain the emerging inequality in China at the time. [18] Josa and Aguado (2020) perform a comprehensive analysis of published research to try and determine the situations where different measurement types are most applicable. A small number of papers, such as Simón (2010) [21], that use the variance of logarithms have cited Foster and Ok as part of their discussion on variable choice, but those are few and far between.

4 Simulation

To attempt to answer the question of similarity between the Gini coefficient and the variance of logs, two simulated approaches are used. The first, referred to as "pairwise comparison", is meant to provide a simple but illustrative look at the agreement between the two inequality measurements. The second, referred to as "ranked comparison", is a more comprehensive method designed to not just identify if differences exist but try to uncover more details about the underlying mechanisms. All coding is done in python and can be found in Appendix C.

4.1 Pairwise Comparison

The pairwise comparison method starts by generating two income distributions. It then calculates the Gini and variance of logs for both distributions and ranks the distributions on both measurements. If the order of the rankings match, then the measurements are said to agree. This process constitutes one iteration of the simulation.

[2]Since we only compare distributions of the same size in this paper, technically speaking this paper could also be considered as looking at the behavior of a series of transfers within a distribution. However given that this paper does not focus on shifting one distribution into another or on the Lorenz criterion, it should not be considered the focus and results between the two paper should not be directly compared.

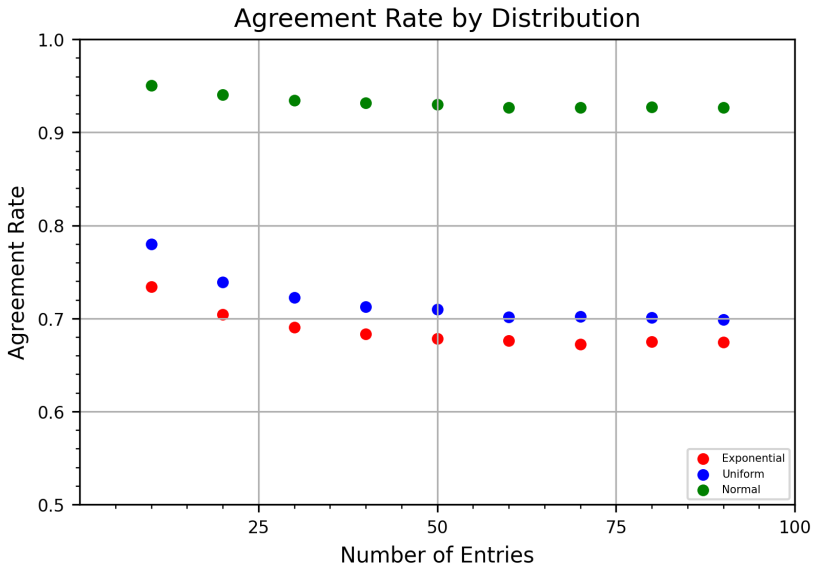


Figure 1: Agreement rate graph for three distribution types. The normal distribution was done with mean 10 and standard deviation 1. The uniform distribution was drawn from values between 0 and 1. The exponential distribution was done with $\lambda = 1$.

At the end of a single iteration, a variable for storing agreement is indexed by 1 if an agreement occurred, and unmodified if a disagreement occurred. The process is then repeated with a new iteration starting with two completely new income distributions. This process repeats for however many iterations are specified. Once all iterations have been completed, the agreement variable is divided by the number of iterations to get an "agreement rate."

Figure 1 shows the results when incomes are drawn from three different probability distributions: exponential, normal[3], and uniform. The number of entries refers to how many "people" were present in each distribution. While results slightly vary along entries when there are very few people in a distribution, the results remain very robust once we move beyond 30 entries. As such, to reduce the simulation time, an entry value of 50 is used throughout this paper rather than a larger value.

This graph provides us with two intuitions that we can use to further examine this issue. First, the shape of the income distribution seems to have an effect on how agreeable the two measurements are. We see that when incomes are drawn from a normal distribution, they align much more frequently than exponential or uniform, which both give similar results. This means we need to be particular

[3]The normal distribution requires selecting a mean value, since picking $\mu = 0$ results in negative values that the logarithm cannot evaluate. See Appendix A.2 for more

about what distributions we use and how we draw conclusions from them. We return to this topic shortly in section 4.3.

Second, the agreement rate for exponential and uniform is lower than one might expect, hovering around 70 percent. If this level of agreement holds for real income distributions, it cultivates some concern that the two measurements may not be as interchangeable as one would hope.

A similar metric is briefly commented on in Foster and Ok, where they estimate the probability of a "violation" of the Lorenz criteria from a transfer to be between 8-12 percent, depending on the data used. Importantly, they comment that this percentage cannot be dismissed as negligible. While the agreement rate calculated in this paper is not the same as the probability of violation in Foster and Ok, the magnitude of the effect we find is also unlikely to be negligible.

At this point, however, we look to shift our focus to a more robust method for examining this issue. While the pairwise method is illustrative and begins to probe at an underlying issue, it is not a method that sound results can be reasonably drawn from. While around 30 percent of exponential income distributions disagree, there is no indication of how severe that disagreement is. This could be investigated further, and may be an area for follow up research, but we choose to change methodology rather than dive further into the pairwise approach. Furthermore, simply comparing two distributions is not nearly as common as ranking a larger set of distributions on inequality.

4.2 Ranked Comparison

Our second method uses a more complex approach in the hope of being able to uncover specific discrepancies. Instead of drawing two income distributions, the ranked comparison method draws a much larger quantity of them. The exact number used varies depending on how time consuming the code is to run, but is typically 1000 in this paper. Anytime results are presented the exact number will be specified.

Once these distributions are generated, the Gini and variance of logs are calculated for each distribution. The distributions are then ranked twice, once by Gini and once by the variance of logs. For each distribution, the gap between the rankings is calculated. For example, if one distribution had the 4th highest Gini coefficient and the 17th highest variance of logs, a gap value of 13 would be scored. These gap values are the primary metric by which we can investigate the relationship between our inequality measurements. Further details on how this method is used in various ways appears in section 5.

4.3 Distribution Type

Before proceeding to any results, we need to return to the question of how to draw income distributions. Following the analysis of Bakar and Pathmanathan (2022), the strongest candidate for modeling income distributions is the Weibull distribution [6].

Given that the goal of this paper is not to match our generated incomes to any specific real world set of incomes, we feel comfortable using a one-parameter Weibull distribution. This parameter is called the shape parameter. [16] While analysis of how the scale and location parameters would also be useful, it fell outside the scope of this paper.

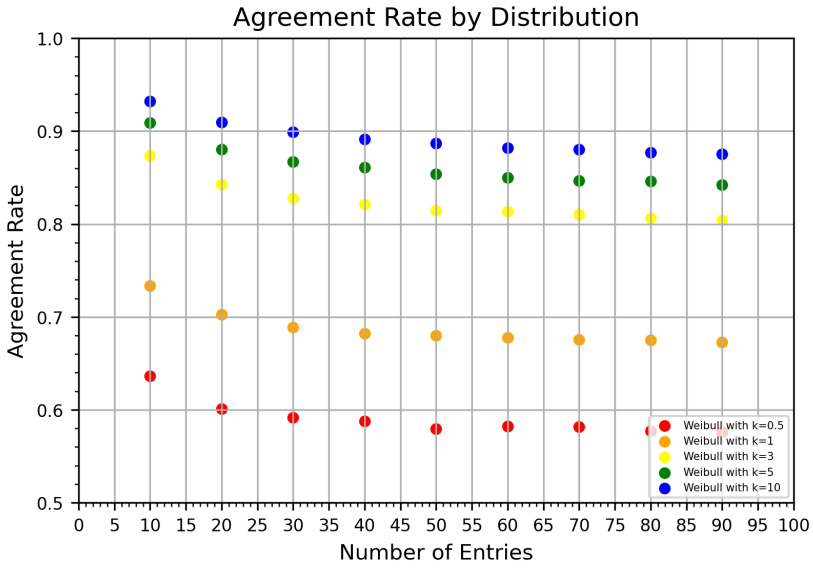
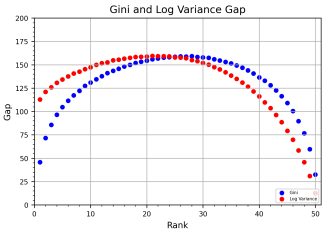


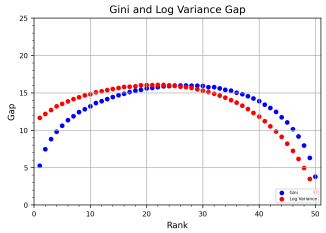
Figure 2: Agreement rate graph for different shape parameters.

The shape parameter can modify the distribution of Weibull to resemble many other distributions types. A shape parameter of 1 corresponds results in the exponential distribution, while a larger value, such as 5, begins to emulate the normal distribution. [17]

This understanding of the Weibull distribution is further illustrated when we repeat the pairwise simulation approach using various Weibull distributions. As we can see, the agreement rate when k=1 closely matches the exponential distribution in Figure 2, and the higher values of 5 and 10 closely match the normal distribution.



(a) 1000 distribtions over 10,000 iterations



(b) 100 distributions over 100,000 iterations

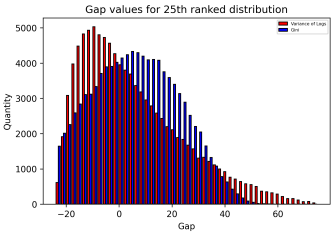
Figure 3: Average gap values for Gini and variance of logs

Bakar and Pathmanthan (2022) estimate the values of various parameters, including the shape parameter, for three countries: Singapore, Malaysia, and Indonesia. When looking at the Weibull-Weibull distribution in each of these, the shape parameter is estimated for all three to be between 2.5 and 3.7. Given this, we choose to use a shape parameter of 3 for our Weibull distributions in our primary results. Results using other shape parameters are discussed in section 5.

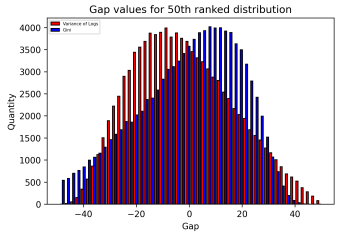
5 Results

As mentioned earlier, the main results of this paper will stem from the ranked comparison methodology. As discussed above, this method generates a list of income distributions and finds the gaps in how the two measurements rank them. These gap values are averaged within buckets. For example, if there are 1000 distributions that are ranked, a .1 percentile grouping based on Gini values would average the gap values of the distributions with the highest 100 Gini coefficients and store that value. It would continue to average for each subsequent grouping of 100.

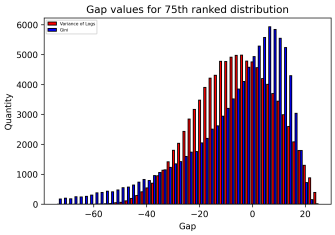
Figure 3 shows a preliminary result using this method. Note that gap values in this case are only ever recorded as positive. This was done to get an idea of how large the magnitude of the gaps are on average. We find that the two measurement regularly produce sizeable differences in ranking. In panel A, distributions ranked in the middle for the Gini coefficient are ranked, on average, more than 100 spots higher or lower in the variance of logarithms, and vice versa. As the distributions get closer to the extremes, they tend to be ranked closer. In panel B we see the vast majority of buckets averaging gaps of over 10. It's worth noting that for a distribution in the very middle, the highest the gap value could be is the number of entries divided by 2, meaning we are seeing the mean mismatch in ranking being 20 percent of the maximum possible value. This seems like a concerning value.



(a) 25th ranked distribution



(b) 50th ranked distribution



(c) 75th ranked distribution

Figure 4: Gap scores for specific ranking values over 100,000 iterations

We also provide some summary information about how gap values are distributed. With 100 distribution ranked, figure 4 showcases the frequency of gap values over 100,000 iterations for various ranking values[4].
We see decently sized gap values for these three distribution rankings. For example, among distributions that were ranked 25th for the Gini index, over 60 percent of them had gap values of 10 or higher, with over 25 percent having gap values of 20 of higher. The same holds for the variance of log distributions ranked 25th. Interestingly, the only level of distribution that was ranked the exact same more than half the time was the 100th ranked distribution, and no other value comes particularly close. Appendix B provides more detailed summary information for all ranking values between.

For the following results, the same simulated data set is used. It was done with 50 entries per distribution, 1000 distributions per iteration, and 100,000 iterations in total. Since a simulated approach does hypothetically allow for selective results, only one simulation was run and with these parameters. In practice, however, 100,000 simulations is well more than enough to get nearly identical results each time. Additionally, the gap values in this case are recorded as positive or negative depending on if the ranking for the analyzed measurement is higher or lower than the other.

Figure 5 shows the main result of this paper. We see that for the vast majority of Gini and variance of logs, the average gap follows very similar trends. However, we see a large and noticeable deviation in the behavior for relatively high

[4]In this instance the gap values are not sorted into buckets

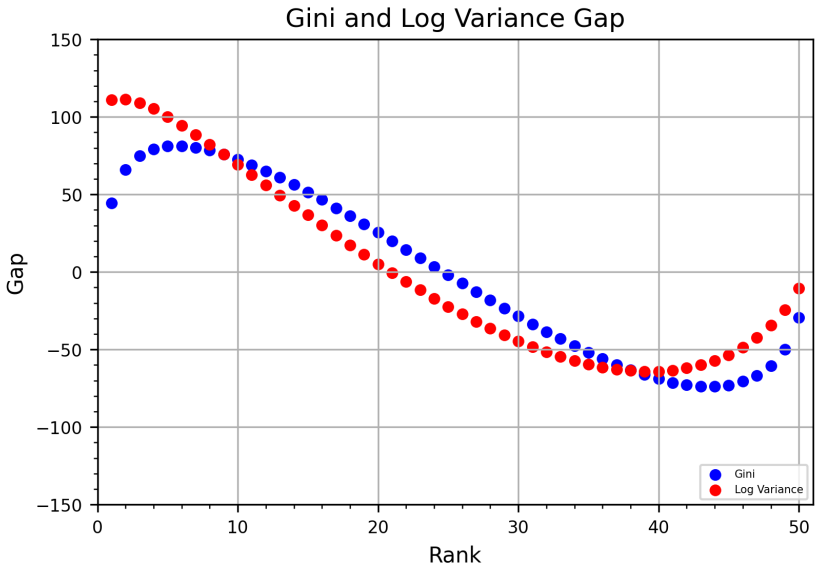


Figure 5

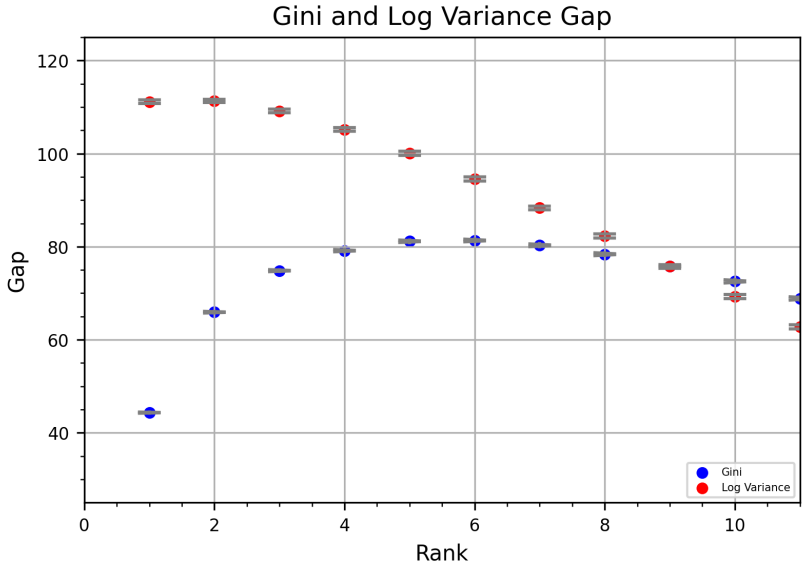


Figure 6

values of the two measurements. Figure 6 highlights the section of interest by zooming in. Note that confidence intervals at 99.9 percent confidence are included in this image, but they are almost imperceptibly narrow given the sample size of 100,000. They are included here for reference but are similarly small for all remaining graphs and are thus omitted moving forward for image readability.

To examine this further, we now plot the gap values of each measurement against themselves, with a comparison between the nth highest bucket and the nth lowest bucket. These pairs of distribution buckets are referred to moving forward as "corresponding." The results are presented in figures 7 and 8. For these figures, the gap values are presenting normally for the highest 50 percent of buckets and negative gap values are presented for the lowest 50 percent of buckets.

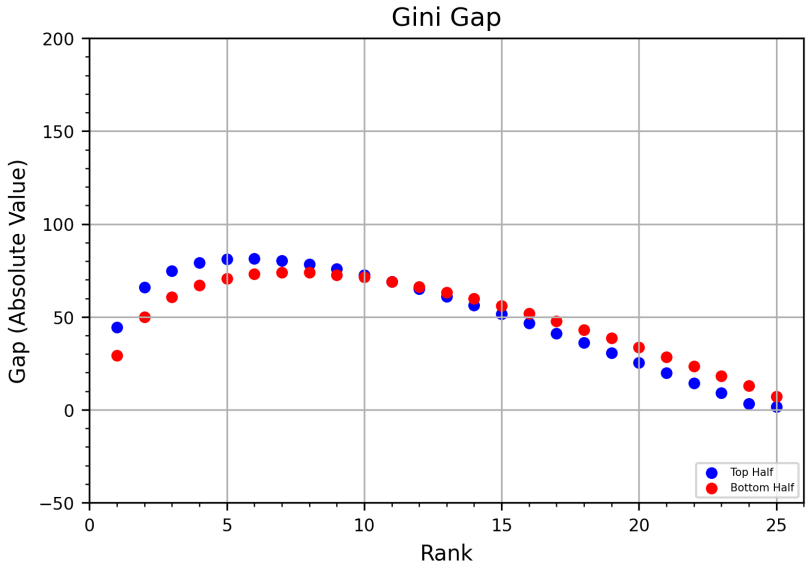


Figure 7: Gap values for the Gini compared between corresponding buckets

Figures 7 and 8 offer further details of the behavior of the two measurement types. We see that the Gini coefficient is internally consistent, with all distributions having gaps scores roughly equal in magnitude to their corresponding distribution. The variance of logs on the other hand displays remarkably different behavior between corresponding buckets. This is largely driven by the behavior of the 10 highest and lowest buckets. It's important to note that the sum of all gap values (before absolute value) is approximately zero[5] so the apparent deviation in the second half of figure 8 is moreso driven by the very high

[5]The results of the simluation used here have total gaps values in python of approximately -1.74e-13 and -2.13e-14 for the Gini and variance of logs respectively, which are close to the number of decimal places stored in Python.

gap values of the top buckets.

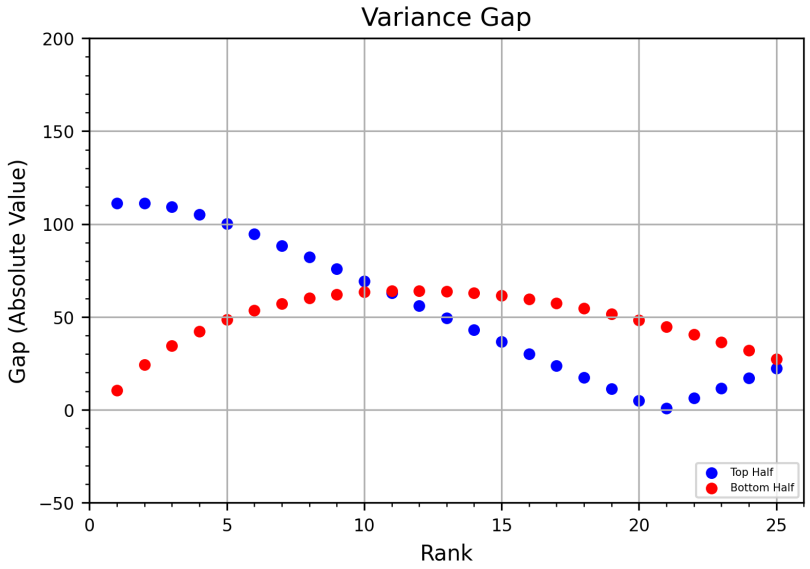


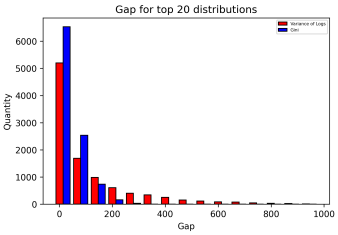
Figure 8: Gap values for the variance of logarithms compared between corresponding buckets

6 Discussion

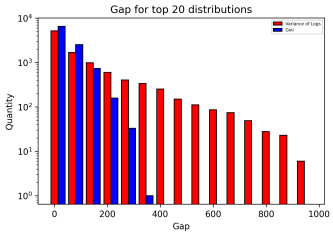
The discussion will be divided into multiple parts. First, we attempt to understand what is driving the highlighted results in section 4. Next, we discuss the limitations of the approach used in this paper and ideas for how to improve the method. Finally, we look at possible ways that this approach can be expanded on to find any related discrepancies.

6.1 Discussion of Results

The biggest hurdle of this paper is determining what constitutes a "good result." The graphs in the previous section showcase that some discrepancy has been uncovered, but given the novel nature of the approach used to get there it is very important that some context and discussion is associated. It seems clear that the focus should be on what is driving the behavior at the top 15 percent of the variance of log values, as those are the ones that both deviate from the Gini and from their lower counterpart. Foster and Ok discuss that previous research indicates that the likeliest violations of the Lorenz criterion occur when at the high end of distributions. Since a transfer from someone very poor to



(a) Linear Scale



(b) Logarithmic Scale

Figure 9: Gap scores for the top 20 distributions for each measurement type over 500 iterations

someone very rich would increase variance of logs more than a "closer" transfer, our findings of deviant behavior at the highest value distributions appears to have some consistency with established work.

First, we should think about how we would "expect" the gap values between two distributions to behave. While the Gini values and the variance of log values are in no way independent, looking at the behavior of two random independent rankings can provide some context. If lists of rankings are chosen at random, then the expected gap values would follow a linear trend. This is the behavior that we see for the majority of the buckets for both the Gini and the variance of logs. However, with the exception of the left tail of the variance of logs, we see a sharp bend back towards zero, indicating that the distributions that take on extreme values for the measurements tend to cluster with one another in a way that the other distributions do not. Again, the exception to this is the distributions that have a high variance of logs.

The fact that these distributions have large gap values while the high Gini distributions have low gap values is puzzling. If the distributions that rank highly for the variance of logs are ones that, on average, rank significantly lower for the Gini, then the distributions with the highest Gini score are not matching with their immediate counterparts as often as the low gap average might suggest. Our initial speculation for a reason is a few outlier distributions that have high variance of logs and low Gini scores.

To verify this, we generate a series of distributions as we have before, storing the gap values of the top 20 distributions for both measurements. Figure 9 shows the frequency of gap values over 500 iterations.

On inspection, it does seem like the effect is driven by some very large gaps among some of the high variance of log distributions. In this simulation, the largest gap among the high value Gini distributions was 346 while there was a distribution with a gap of 978 among the high value variance of log distributions, meaning it had a top 20 variance of log value alongside a bottom top 20 Gini coefficient.

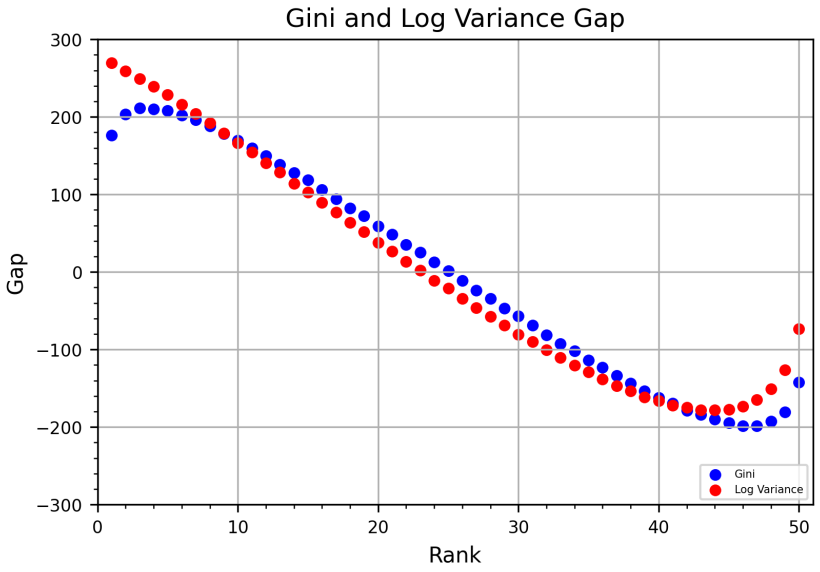


Figure 10: Gap scores with k = 1

6.2 Shape Parameter

Figures 10 and 11 showcase the gap comparison for shape parameters of 1 and 5 respectively. There are a couple things to note with these. First, we do still see some deviation between the two measurements for the high percentile buckets, once again driven by the behavior of the left tail of the Variance of Logs. When the shape parameter is 1, the Variance of Logs remains almost perfectly linear on the left tail. When k = 5, the difference in behavior is less pronounced. Second, the magnitude of the average gaps decreases as the shape parameter increases. Looking back to figure 2, we see something similar for the pairwise comparison method, with the agreement rate increasing as we increase k. This gives us further evidence that the larger our shape parameter gets, the more comparable the two measurements are.

Figure 9 is recreated with shape parameters of 1 and 5. These can be found in Appendix 1.

6.3 Limitations

The primary limitation of the method deployed in this paper is the lack of clearly meaningful results. By not following a standard economic procedure,

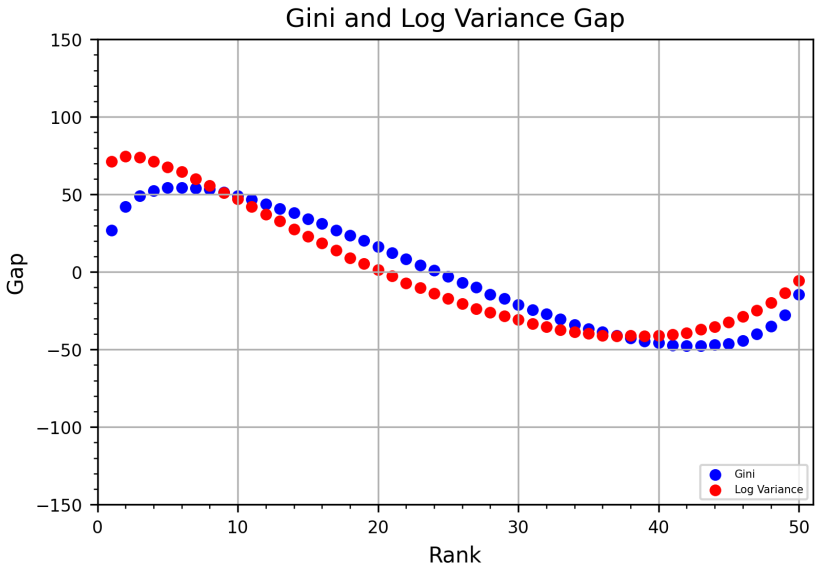


Figure 11: Gap scores with k = 5

there is no ability to point to a clear result that has an established context and meaning. While a clear discrepancy has been established, it is not yet clear to what extent those discrepancies matter in real research.

The other big limitation comes from the code used. While the code is perfectly functional, it takes a long time to run, especially when a large number of iterations are performed. For example, the program used to generate the 100,000 iterations in section 4 took around 60 hours to run. For any future work done with this methodology, it would be worth looking into ways to optimize the code, perhaps by using another language for the coding. The length of this process is primarily driven by the calculation of the Gini coefficient, since it requires comparing every value to every other value in the distribution rather than simply every value to the mean.

There are also some other smaller aspects of the process that are worth discussing:

- The gap values used before averaging cared about the order of the two distributions. In other words, they were positive if the ranking for the Gini coefficient was higher than the ranking for the variance of logs, but negative if it was the other way around.[6] That is why you see gap values close to zero in the middle of figure 5. While the results on the left would

[6]This is assuming we are looking at the Gini graph. This statement is reversed in reference to the variance of logs

be largely unaffected by changing this, you might pick up different results in the middle of the distribution if the absolute value of the gaps were used.

- As discussed in section 4.3, only a one-parameter Weibull distribution is used. This is an unfortunate limitation of this paper and one that should be improved on in future work.

- This paper provides no indication of which distribution type is better. While Foster and Ok clearly indicate that the variance of logarithms can violate accepted norms of inequality measures, and this paper does show inconsistencies for the distributions with high variance of logarithms, these results should not be interpreted as a clear endorsement of the Gini coefficient.

6.4 Future Work

There are two areas in which future work seems viable. The first is expanding and generalizing the work done in this paper. The clearest area for that is by examining other inequality measurements. Keeping the Gini would be a good idea given its prevalence in the field, but there is ample opportunity to compare it any of the other measures of inequality sometimes used in the field. While this paper chose to look at two holistic measurements for the reasons stated in section 2, there is value in comparing other measurement types, especially given how popular ratio or percentile metrics are. There is also still work that could be done further examining the causes of the variance of logs deviation, which might require much higher mathematical sophistication than possible in the scope of this paper. Alternatively, the simulation approach in this paper could be adapted to more directly analyze the results of Foster and Ok.

The second major area for further work is real world investigation of the results uncovered here. While originally intended to be a part of this paper, the length of the coding and simulating process ended up reducing the capacity for this sort of investigation. The following are some procedures that we think could be viable and complimentary to the work done here:

- An investigation of agreement between the Gini and variance of logs using real world data, rather than simulated data, similar to what was done in Atkinson (1970). The limiting factor here is sample size, as the simulation approach can generate far more "income distributions" much faster. However, real world data does not rely on assumptions about the shape of income distributions and is more in line with the data that academic research is published on.

- A review of measurement type usage in academic work. While part of the issue is certainly if discrepancies exist, another important part is how

they are used. Getting a comprehensive overview of how frequently measurement types are used could help indicate where any problems might lie.

- A review of published results using various measurement types.
- A review of specific researcher behavior in regards to measurement type. While the inspiration of this paper came from concern with "accidental researcher influence," it is not outside the realm of possibility that researchers could purposely select a specific inequality measurement to get a desired result or reach a desired significance level.

7 Conclusion

Proper understanding of how the various measurements of inequality work is incredibly important to the field. However, despite the importance, the issue of indifference to measurement type has persisted as a problem for at least a century since Dalton's 1920 paper was published. This paper provides a novel strategy for trying to understand how worrying that indifference is.

Through a process of simulating incomes by drawing from a one-parameter Weibull distribution, this paper finds that the Gini coefficient and the variance of logarithms have clear inconsistencies, though the exact importance of these inconsistencies is not clear. We find that the Gini coefficient and the variance of logarithms will rank two randomly drawn distributions differently 10-30 percent of the time when the shape parameter of the Weibull distribution is between 1 and 5.

Furthermore, when a more comprehensive ranking system is adopted we find that there can be significant differences in how two distributions are ranked. We see that for a ranking of 100 distributions, more than half of those distributions are ranked at least 10 positions differently under the two measurement types. The gap size is, on average, largest for the distributions that are ranked in the middle.

We also see that the variance of logarithms displays highly abnormal behavior for the high ranking distributions. This provides further evidence that the variance of logarithms has issues as an inequality measurement and the behavior seen here and in Foster and Ok's 1999 paper should be considered by any authors that choose to use this measurement system.

Finally, the author believes that this methodology of using a large amount of simulated data to test inequality measurements shows a lot of promise. This paper only scratches the surface of the potential, and there is a lot of improvement and alternate specifications that can be employed. The code used in this paper is made available and is open to be used and modified in any way that future

researchers looking at this topic see fit. It is the author's hope that choice of measurement type is more properly understood and rigorously justified in future research.

References

[1] The 17 goals: Sustainable development. Accessed on May 6, 2024.

[2] Mark Aguiar and Mark Bils. Has consumption inequality mirrored income inequality? *The American Economic Review*, 105(9):2725–2756, 09 2015. Copyright - Copyright American Economic Association Sep 2015; Last updated - 2023-11-18; CODEN - AENRAA.

[3] Brindusa Anghel, Henrique Basso, Olympia Bover, José María Casado, Laura Hospido, Mario Izquierdo, Ivan A. Kataryniuk, Aitor Lacuesta, José Manuel Montero, and Elena Vozmediano. Income, consumption and wealth inequality in Spain. *SERIEs*, 9(4):351–387, 11 2018.

[4] Anthony B Atkinson. On the measurement of inequality. *Journal of Economic Theory*, 2(3):244–263, 1970.

[5] Orazio P. Attanasio and Luigi Pistaferri. Consumption inequality. *The Journal of Economic Perspectives*, 30(2):3–28, Spring 2016. Copyright - Copyright American Economic Association Spring 2016; Document feature - ; Last updated - 2023-11-27; SubjectsTermNotLitGenreText - United States–US.

[6] Shaiful Anuar Abu Bakar and Dharini Pathmanathan. Income modeling with the weibull mixtures. *Communications in Statistics - Theory and Methods*, 51(11):3612–3628, 2022.

[7] Matthew Brzozowski, Martin Gervais, Paul Klein, and Michio Suzuki. Consumption, income, and wealth inequality in canada. *Review of Economic Dynamics*, 13(1):52–75, 2010. Special issue: Cross-Sectional Facts for Macroeconomists.

[8] Lidia Ceriani and Paolo Verme. The origins of the gini index: extracts from variabilita e mutabilita (1912) by corrado gini. *Journal of Economic Inequality*, 10(3):421–443, 09 2012. Copyright - Springer Science+Business Media, LLC. 2012; Document feature - ; Tables; Equations; People - Gini, Corrado (1884-1965; Last updated - 2023-11-27; SubjectsTermNotLitGenreText - Gini, Corrado (1884-1965.

[9] Olivier Coibion, Yuriy Gorodnichenko, and Dmitri Koustas. Consumption inequality and the frequency of purchases. *American Economic Journal: Macroeconomics*, 13(4):449–82, October 2021.

[10] Hugh Dalton. The measurement of the inequality of incomes. *The Economic Journal*, 30(119):348–361, 1920.

[11] James E. Foster and Efe A. Ok. Lorenz dominance and the variance of logarithms. *Econometrica*, 67(4):901–907, 1999.

[12] Joseph L. Gastwirth. Is the gini index of inequality overly sensitive to changes in the middle of the income distribution? *Statistics and Public Policy*, 4(1):1–11, 2017.

[13] Charles R. Harris, K. Jarrod Millman, Stéfan J. van der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser, Julian Taylor, Sebastian Berg, Nathaniel J. Smith, Robert Kern, Matti Picus, Stephan Hoyer, Marten H. van Kerkwijk, Matthew Brett, Allan Haldane, Jaime Fernández del Río, Mark Wiebe, Pearu Peterson, Pierre Gérard-Marchant, Kevin Sheppard, Tyler Reddy, Warren Weckesser, Hameer Abbasi, Christoph Gohlke, and Travis E. Oliphant. Array programming with NumPy. *Nature*, 585(7825):357–362, September 2020.

[14] J. D. Hunter. Matplotlib: A 2d graphics environment. *Computing in Science & Engineering*, 9(3):90–95, 2007.

[15] John P A Ioannidis. Meta-research: Why research on research matters. *PLoS biology*, 16(3):e2005468–e2005468, 2018.

[16] Debasis Kundu and Mohammad Z. Raqab. Estimation of r=p(y¡x) for three-parameter weibull distribution. *Statistics Probability Letters*, 79(17):1839–1846, 2009.

[17] Ayşe Kızılersü, Markus Kreer, and Anthony W. Thomas. The Weibull Distribution. *Significance*, 15(2):10–11, 04 2018.

[18] Jonathan Morduch and Terry Sicular. Rethinking inequality decomposition, with evidence from rural china. *The Economic Journal*, 112(476):93–106, 2002.

[19] United Nations Department of Economic and Social Affairs. *World Social Report 2020*. United Nations, 2020 edition, 2020.

[20] John W Patty and Elizabeth Maggie Penn. Measuring fairness, inequality, and big data: Social choice since arrow. *Annual review of political science*, 22(1):435–460, 2019.

[21] Hipolito Simon. International differences in wage inequality: A new glance with european matched employer-employee data. *British journal of industrial relations*, 48(2):310–346, 2010.

[22] Yu Xie and Xiang Zhou. Income inequality in today's china. *Proceedings of the National Academy of Sciences - PNAS*, 111(19):6928–6933, 2014.

A Appendix A: Graphs

A.1 Shape Parameter Changes

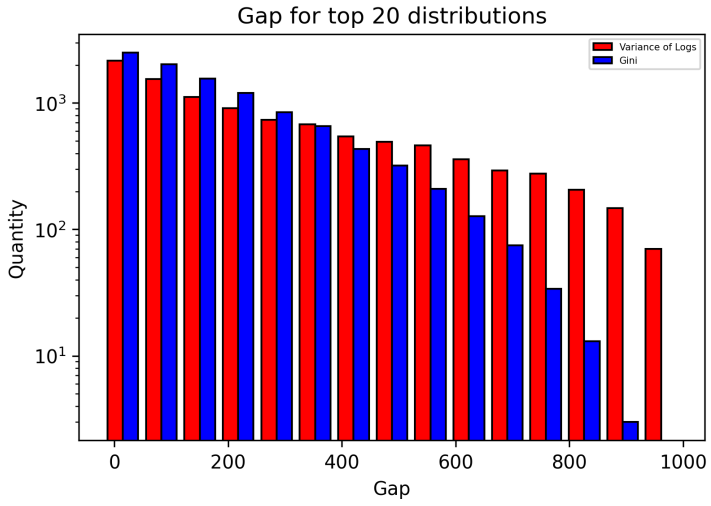


Figure 12: Gap scores for the top 20 distributions over 500 iterations on a logarithmic scale (k=1)

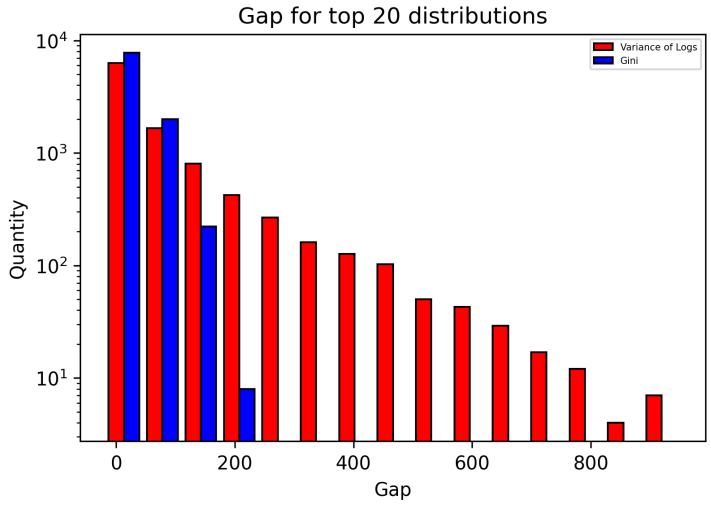


Figure 13: Gap scores for the top 20 distributions over 500 iterations on a logarithmic scale (k=5)

A.2 Robustness of Normal Agreement Rate

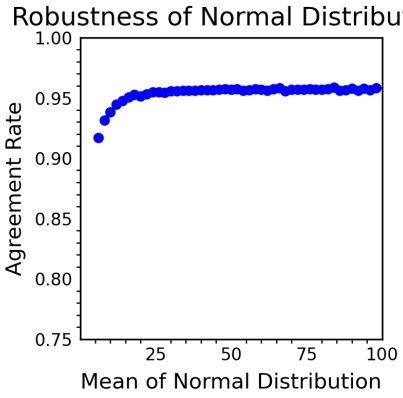


Figure 14: Agreement rate for normal distribution with varying means. Outside of a few very low values, the mean matters very little.

B Appendix B: Summary tables

The following tables give information about the gap values for one of the simulations in section 5. This simulation had 100 distributions per iteration, and was performed over 100,000 iterations. The exact ranking column gives a count of how many distributions were matched in the exact same spot for the Gini and variance of logarithms. The "gap of x or more" columns give a count of how many of the distributions had gap values equal to or greater than the specified number.

Distribution Rank	Median Gap	Exact Ranking	Gap of 10 or More	Gap of 20 or more	Gap of 30 or More
1	3	20385	13873	2000	207
2	4	10484	22748	4425	651
3	5	7465	28462	6918	1131
4	5	6061	32203	9129	1777
5	6	5047	35283	11223	2366
6	6	4594	37022	12825	3000
7	6	4153	38578	14241	3632
8	7	3879	39694	15691	4159
9	7	3462	40408	16596	4699
10	8	3239	41370	17965	5454
11	8	3053	43780	18446	5671
12	9	2889	46430	19439	6255
13	9	2831	48297	19768	6580
14	10	2822	50172	20274	6797
15	10	2648	52097	20877	7274
16	10	2510	53552	21519	7602
17	11	2481	54468	21365	7593
18	11	2388	55977	21883	8001
19	11	2370	56972	22175	8130
20	11	2409	57816	22257	8301
21	12	2228	58622	22907	8063
22	12	2089	59533	24563	8290
23	12	2190	60069	25339	8299
24	12	2103	61334	26831	8383
25	13	2079	61268	27246	8363

Figure 15: Various summary information for the Gini distributions ranked 1-25

Distribution Rank	Median Gap	Exact Ranking	Gap of 10 or More	Gap of 20 or more	Gap of 30 or More
26	13	2069	62180	28066	8143
27	13	2040	62619	29201	8320
28	13	2063	62899	29604	8192
29	13	1968	63645	30477	8205
30	13	1906	63879	30635	7942
31	14	1899	64190	31385	8610
32	14	1905	64796	32059	9051
33	14	1856	64840	32390	9322
34	14	1856	65147	32891	9775
35	14	1907	65206	33106	10128
36	14	1833	65425	33331	10484
37	14	1835	65694	33667	10770
38	14	1818	66088	34275	11156
39	14	1855	65883	34288	11415
40	14	1844	66199	34479	11467
41	14	1796	66445	34761	11701
42	14	1831	66148	34530	11993
43	14	1801	66575	34831	12051
44	14	1776	66363	34698	12264
45	14	1781	66641	35078	12454
46	14	1795	66314	34857	12423
47	14	1849	66350	34755	12504
48	14	1722	66405	34753	12575
49	14	1771	66466	34712	12717
50	14	1821	66038	34308	12350

Figure 16: Various summary information for the Gini distributions ranked 26-50

Distribution Rank	Median Gap	Exact Ranking	Gap of 10 or More	Gap of 20 or more	Gap of 30 or More
51	14	1841	66161	34331	12697
52	14	1825	65770	34014	12489
53	14	1842	65842	33750	12629
54	14	1855	65736	33666	12751
55	14	1888	65540	33137	12457
56	14	1871	65541	33115	12664
57	14	1844	64864	32427	12480
58	14	1866	64720	32164	12433
59	14	1939	64338	31391	12423
60	14	1939	64214	31387	12419
61	13	1890	64062	30777	12548
62	13	1879	63647	30472	12690
63	13	1985	63182	29639	12677
64	13	2000	62688	29150	12830
65	13	1949	62104	28308	12744
66	13	2048	61813	27882	12580
67	12	2058	61346	27097	12680
68	12	2080	60727	26336	12847
69	12	2117	59882	25679	12840
70	12	2164	59473	25306	13053
71	12	2168	58201	24230	12768
72	11	2201	57941	23926	12961
73	11	2314	56776	23094	12987
74	11	2406	56192	22750	13227
75	11	2379	54969	22025	12941

Figure 17: Various summary information for the Gini distributions ranked 51-75

Distribution Rank	Median Gap	Exact Ranking	Gap of 10 or More	Gap of 20 or more	Gap of 30 or More
76	10	2522	53957	21699	12796
77	10	2405	52953	21268	12800
78	10	2607	51580	21026	12660
79	10	2629	50018	20752	12548
80	9	2741	48746	20519	12476
81	9	2808	46969	20193	12250
82	9	2923	45278	19884	12048
83	8	3107	43690	19666	11911
84	8	3149	41789	19549	11629
85	8	3303	39834	19106	11341
86	7	3506	37536	18699	11219
87	7	3643	35276	18088	10569
88	6	3932	33131	17487	10339
89	6	4114	31511	17053	10040
90	6	4356	29795	16332	9450
91	5	4852	28963	15715	9168
92	5	5377	28035	14969	8665
93	4	5761	26562	13790	7930
94	4	6627	25178	12835	7267
95	3	7741	23747	11919	6547
96	3	9115	21940	10744	6034
97	2	11471	19515	9249	5107
98	2	15872	16623	7660	4149
99	1	25142	12959	5708	3038
100	0	53214	8241	3506	1943

Figure 18: Various summary information for the Gini distributions ranked 76-100

Distribution Rank	Median Gap	Exact Ranking	Gap of 10 or More	Gap of 20 or More	Gap of 30 or More
1	5	20385	35535	20364	12254
2	5	10484	36152	20641	12522
3	5	7465	37009	21364	12887
4	5	6061	37597	21882	13122
5	5	5047	37999	22409	13617
6	5	4594	38464	22733	13768
7	6	4153	38280	22942	14008
8	6	3879	38278	23199	13927
9	7	3462	38417	23310	14259
10	7	3239	38008	23395	14288
11	8	3053	40672	23369	14105
12	8	2889	42895	23101	14037
13	9	2831	45426	23292	14129
14	9	2822	47649	23196	14025
15	9	2648	49800	23189	14129
16	10	2510	51735	23214	14022
17	10	2481	52974	23076	14019
18	10	2388	54418	22896	13913
19	11	2370	55641	22892	13788
20	11	2409	56580	22678	13612
21	11	2228	57784	22848	13335
22	12	2089	58674	23398	13343
23	12	2190	59330	24013	13036
24	12	2103	60476	24754	13008
25	12	2079	61091	25665	12752

Figure 19: Various summary information for the variance distributions ranked 1-25

Distribution Rank	Median Gap	Exact Ranking	Gap of 10 or More	Gap of 20 or More	Gap of 30 or More
25	12	2079	61091	25665	12752
26	13	2069	61715	26630	12711
27	13	2040	62245	27229	12385
28	13	2063	62393	27868	12004
29	13	1968	63168	28868	12101
30	13	1906	63573	29433	11854
31	13	1899	64044	30207	11745
32	13	1905	64252	30552	11579
33	14	1856	64674	31282	11434
34	14	1856	64969	31865	11572
35	14	1907	65118	32405	11437
36	14	1833	65452	33014	11672
37	14	1835	65436	33096	11871
38	14	1818	65889	33790	11992
39	14	1855	65788	33757	11871
40	14	1844	66085	34246	12201
41	14	1796	65866	34153	12164
42	14	1831	66424	34773	12550
43	14	1801	66391	34646	12380
44	15	1776	66474	35075	12503
45	14	1781	66410	34895	12547
46	15	1795	66434	34907	12675
47	15	1849	66545	35338	12810
48	15	1722	66649	35399	12924
49	15	1771	66505	35260	12754

Figure 20: Various summary information for the variance distributions ranked 26-50

Distribution Rank	Median Gap	Exact Ranking	Gap of 10 or More	Gap of 20 or More	Gap of 30 or More
51	14	1841	66428	35045	12622
52	14	1825	66104	34724	12450
53	14	1842	66275	34748	12558
54	14	1855	65899	34440	12205
55	14	1888	65712	34260	12223
56	14	1871	65602	33809	12097
57	14	1844	65343	33766	11791
58	14	1866	64878	33236	11433
59	14	1939	65028	33279	11343
60	14	1939	64426	32691	11195
61	14	1890	64351	32393	10636
62	14	1879	64209	31854	10479
63	13	1985	63513	31351	10069
64	13	2000	63044	30571	9977
65	13	1949	62770	30325	9558
66	13	2048	62391	29843	9196
67	13	2058	61868	28987	8912
68	13	2080	61227	28310	8572
69	12	2117	60922	27753	8348
70	12	2164	60326	26746	7867
71	12	2168	59339	25950	7602
72	12	2201	58944	25279	7633
73	12	2314	58176	24414	7325
74	11	2406	57554	23612	7078
75	11	2379	56430	22573	6835

Figure 21: Various summary information for the variance of log distributions ranked 51-75

Distribution Rank	Median Gap	Exact Ranking	Gap of 10 or More	Gap of 20 or More	Gap of 30 or More
76	11	2522	55682	21725	6644
77	11	2405	54639	20549	6249
78	10	2607	53278	19382	5968
79	10	2629	52267	18502	5519
80	10	2741	50617	17285	5240
81	9	2808	49613	16718	4925
82	9	2923	47975	15937	4551
83	9	3107	46463	15118	4118
84	8	3149	44390	14200	3745
85	8	3303	42881	13536	3397
86	8	3506	40374	12290	2865
87	7	3643	38146	11307	2595
88	7	3932	35525	10298	2122
89	6	4114	32944	9085	1687
90	6	4356	30435	7904	1355
91	6	4852	27467	6701	1023
92	5	5377	25289	5526	786
93	5	5761	23162	4502	553
94	4	6627	20180	3314	308
95	4	7741	16685	2302	197
96	3	9115	13573	1479	112
97	2	11471	9813	789	38
98	2	15872	6193	362	16
99	1	25142	3138	115	3
100	0	53214	965	28	0

Figure 22: Various summary information for the variance of log distributions ranked 76-100

C Appendix C: Code

The following sections are various files used in the code. Special thanks to the Numpy [13] and MatPlotLib python packages. [14] Code files can be found at https://github.com/NotTheTanner/A-Simulated-Approach-to-Agreeability-of-the-Gini-Coefficient-and-the-Variance-of-Logarithms/tree/main, and may be updated to reflect and changes/improvements made. The code in this appendix is the code used for the original submission of this paper.

C.1 "Pairwise"

```
    # -*- coding: utf-8 -*-
"""
Jack Dickinson Thesis 2024
"""
import numpy as np

#Global values to use if no imputs are specified:

g_sets = 2
g_entries = 100
g_iterations = 1000
g_distribution_type = "e"


#Two main functions that print an agreement rate. Only differ on if
#they ask for user input in the function call or the console

def main():

    sets, entries, iterations, dist = ask_for_inputs()
    if dist == "n" and sets == 2:
        agreement_normal = simulate_pairwise_normal(sets, entries,
                                        iterations, 10)
        print(agreement_normal)
    elif dist == "u" and sets == 2:
        agreement_uniform = simulate_pairwise_uniform(sets, entries
                                        , iterations)
        print(agreement_uniform)
    elif dist == "e" and sets == 2:
        agreement_exponential = simulate_pairwise_exponential(sets,
                                         entries, iterations)
        print(agreement_exponential)

def main_no_ask(sets, entries, iterations, dist):
    if dist == "n" and sets == 2:
        agreement_normal = simulate_pairwise_normal(sets, entries,
                                        iterations)
        print(agreement_normal)
    elif dist == "u" and sets == 2:
        agreement_uniform = simulate_pairwise_uniform(sets, entries
                                        , iterations)
        print(agreement_uniform)
    elif dist == "e" and sets == 2:
```

```
        agreement_exponential = simulate_pairwise_exponential(sets,
                                            entries, iterations)
        print(agreement_exponential)


#Optional function to allow user modifications to inputs:

def ask_for_inputs():
    resolved = 0
    while resolved == 0:
        print("Would you like to modify the parameters of the
                                        simulation? Type Yes or
                                        No")
        response1 = input()

        if response1 == "yes" or response1 == "Yes":
            print("How many entries would you like each
                                            distribution to
                                            contain? (Current
                                            value = " + str(
                                            g_entries)+")")
            entries_local = int(input())
            print("How many iterations would you like to run? (
                                            Current value = " +
                                            str(g_iterations)+")"
                                            )
            iterations_local = int(input())
            print("What distribution type would you like to use? n
                                            = normal, u = uniform
                                            , e = exponential")
            distribution_type_local = input()

            resolved = 1

            return(g_sets, entries_local, iterations_local,
                                            distribution_type_local
                                            )

        elif response1 == "No" or response1 == "no":

            resolved = 1

            return(g_sets, g_entries, g_iterations,
                                            g_distribution_type)
        else:
            print("Please enter either yes or no")



#Creating sets based on certain distributions

def create_sets_uniform(sets,entries):
    mainarray_unif = []
    for x in range(sets):
        mainarray_unif.append(np.random.uniform(size=entries))
```

```
    return mainarray_unif

def create_sets_normal(sets,entries,center):
    mainarray_norm = []
    for x in range(sets):
        mainarray_norm.append(np.random.normal(loc=center, scale =
                                        1, size=entries))

    return mainarray_norm

def create_sets_exponential(sets,entries):
    mainarray_exp = []
    for x in range(sets):
        mainarray_exp.append(np.random.exponential(scale=1, size=
                                        entries))

    return mainarray_exp

def create_sets_weibull(sets,entries,k):
    mainarray_exp = []
    for x in range(sets):
        mainarray_exp.append(np.random.weibull(k,size=entries))

    return mainarray_exp



#Calculating relevant variables from an array of distributions

def log_var(distributions):
    log_vars = []
    for x in range(len(distributions)):
        log_varx = np.var(np.log(distributions[x]))
        log_vars.append(log_varx)

    return(log_vars)

def gini_discrete(distributions):
    gini = []
    for x in range(len(distributions)):
        ginix = 0
        for y in range(len(distributions[x])):
            for z in range(len(distributions[x])):
                giniz = abs(distributions[x][y]-distributions[x][z]
                                                )
                ginix += giniz

        ginix = ginix/(2*len(distributions[x])**2*np.mean(
                                        distributions[x]))
        gini.append(ginix)


    return(gini)
```

```
#Functions for comparing two distributions

def compairwise(something):
    logs = log_var(something)
    ginis = gini_discrete(something)
    winner = []

    if logs[0] > logs[1]:
        winner.append(0)
    elif logs[0] < logs[1]:
        winner.append(1)
    elif logs[0] == logs[1]:
        print("How did we get here?") #Should be near impossible
                                        for this to happen but
                                        including it as a
                                        failsafe

    if ginis[0] > ginis[1]:
        winner.append(0)
    elif ginis[0] < ginis[1]:
        winner.append(1)
    elif ginis[0] == ginis[1]:
        print("How did we get here? (Gini's version)")

    return winner



#Functions for calculating agreement rate for different
                                    distribution types

def simulate_pairwise_uniform(u_sets,u_entries,u_iterations):
    agreement = 0
    for x in range(u_iterations):
        sets = create_sets_uniform(2, u_entries)
        results = compairwise(sets)
        if results[0] == results[1]:
            agreement += 1

    agreement_rate = agreement/u_iterations

    return agreement_rate

def simulate_pairwise_normal(n_sets,n_entries,n_iterations,center):
    agreement = 0
    for x in range(n_iterations):
        sets = create_sets_normal(2, n_entries, center)
        results = compairwise(sets)
        if results[0] == results[1]:
            agreement += 1

    agreement_rate = agreement/n_iterations

    return agreement_rate

def simulate_pairwise_exponential(e_sets,e_entries,e_iterations):
    agreement = 0
```

```
    for x in range(e_iterations):
        sets = create_sets_exponential(2, e_entries)
        results = compairwise(sets)
        if results[0] == results[1]:
            agreement += 1

    agreement_rate = agreement/e_iterations

    return agreement_rate

def simulate_pairwise_weibull(w_sets,w_entries,w_iterations,k):
    agreement = 0
    for x in range(w_iterations):
        sets = create_sets_weibull(2, w_entries, k)
        results = compairwise(sets)
        if results[0] == results[1]:
            agreement += 1

    agreement_rate = agreement/w_iterations

    return agreement_rate
```

C.2 "Agreement Graphs"

```
    import matplotlib.pyplot as plt

from pairwise import simulate_pairwise_exponential,
                                simulate_pairwise_normal,
                                simulate_pairwise_uniform

entries_values = list(range(100,1000,100))

def exponential_on_entries():
    agreement_rates_exp = []
    for x in entries_values:
        rate_exp = simulate_pairwise_exponential(2, x, 5000)
        agreement_rates_exp.append(rate_exp)

    return agreement_rates_exp

def uniform_on_entries():
    agreement_rates_unif = []
    for x in entries_values:
        rate_unif = simulate_pairwise_uniform(2, x, 5000)
        agreement_rates_unif.append(rate_unif)

    return agreement_rates_unif

def normal_on_entries():
    agreement_rates_norm = []
    for x in entries_values:
        rate_norm = simulate_pairwise_normal(2, x, 5000, 25)
        agreement_rates_norm.append(rate_norm)

    return agreement_rates_norm
```

```
def graph():

    plt.rcParams['figure.dpi'] = 300

    e = exponential_on_entries()
    print("Done with exponentials") #Progress check, can be removed
    u = uniform_on_entries()
    print("Done with uniforms")
    n = normal_on_entries()
    print("Done with normal")

    ax = plt.gca()
    ax.set_ylim([.5,1])
    ax.set_xlim([0,1000])

    plt.minorticks_on()

    plt.xticks(ticks = list(range(100,1001,100)))
    plt.tick_params(axis='both', labelsize = 8)

    plt.grid()

    plt.scatter(entries_values, e, s=20, c="red", label="
                                        Exponential")
    plt.scatter(entries_values, u, s=20, c="blue", label="Uniform")
    plt.scatter(entries_values, n, s=20, c="green", label='Normal')

    plt.title("Agreement Rate by Distribution")
    plt.xlabel("Number of Entries")
    plt.ylabel("Agreement Rate")

    plt.legend(loc=4, fontsize='5')



    plt.show

    return
```

C.3 "Normal Robustness"

```
    import matplotlib.pyplot as plt

from pairwise import simulate_pairwise_normal

x1 = 6
x2 = 100

centers = list(range(x1,x2,2))

def norm_robust_on_center(entries, iterations):
    agreements=[]
    for x in centers:
        print(x)
```

```
        result = simulate_pairwise_normal(2, entries, iterations, x
                                          )
        agreements.append(result)

    return agreements

def graph_normal_robustness():

    plt.rcParams['figure.dpi'] = 300

    n = norm_robust_on_center(20, 100000)

    plt.scatter(centers, n, s=15, c="blue")

    plt.title('Robustness of Normal Distribution')
    plt.xlabel("Mean of Normal Distribution")
    plt.ylabel("Agreement Rate")

    ax = plt.gca()
    ax.set_ylim([.75,1])
    ax.set_xlim([0,100])

    plt.minorticks_on()

    plt.xticks(ticks = list(range(25,101,25)))
    plt.tick_params(axis='both', labelsize = 8)

    plt.grid()

    plt.show

    return
```

C.4 "Weibull Agreement Graph"

```
    # -*- coding: utf-8 -*-
"""
@author: Jack
"""

import matplotlib.pyplot as plt

from pairwise import simulate_pairwise_weibull

entries_values = list(range(10,100,10))

ks=[.5,1,3,5,10]

colors = ['red', 'orange', 'yellow', 'green', 'blue']

def weibull_on_entries(k):
    agreement_rates_weib = []
    for x in entries_values:
        rate_exp = simulate_pairwise_weibull(2, x, 100000, k)
        print(x)
```

```
        agreement_rates_weib.append(rate_exp)

    return agreement_rates_weib

def graph_weib():

    plt.rcParams['figure.dpi'] = 300

    ax = plt.gca()
    ax.set_ylim([.5,1])
    ax.set_xlim([0,100])

    plt.minorticks_on()

    plt.xticks(ticks = list(range(0,101,5)))
    plt.tick_params(axis='both', labelsize = 8)

    plt.grid()

    for x in range(len(ks)):
        print(x)
        w = weibull_on_entries(ks[x])
        plt.scatter(entries_values, w, s=20, c=colors[x], label="
                                        Weibull with k="+str(ks[x
                                        ]))


    plt.title("Agreement Rate by Distribution")
    plt.xlabel("Number of Entries")
    plt.ylabel("Agreement Rate")

    plt.legend(loc=4, fontsize='5')



    plt.show

    return
```

C.5 "Weibull Multi"

```
    # -*- coding: utf-8 -*-


import numpy as np

import copy

def create_sets_weibull(sets,entries,k):
    mainarray_exp = []
    for x in range(sets):
        mainarray_exp.append(np.random.weibull(k,size=entries))

    return mainarray_exp
```

```
#Calculating relevant variables from an array of distributions

def log_var(distributions):


    log_varx = np.var(np.log(distributions))


    return(log_varx)

def gini_discrete_multi(distributions):

    ginix = 0
    for y in range(len(distributions)):
        for z in range(len(distributions)):
            giniz = abs(distributions[y]-distributions[z])
            ginix += giniz

    ginix = ginix/(2*len(distributions)**2*np.mean(distributions))



    return(ginix)


def generate_multi(distributions):
    ginis = []
    log_vars = []
    for x in range(len(distributions)):
        g = []
        ginix = gini_discrete_multi(distributions[x])
        g.append(ginix)
        g.append(x)
        ginis.append(g)

        l = []
        log_varx = log_var(distributions[x])
        l.append(log_varx)
        l.append(x)
        log_vars.append(l)

    return ginis, log_vars

def bubble_sort_a(gini, log_var):
    for x in range(len(gini)-1):
        counter = 0
        for y in range(len(gini)-1):
            if gini[y][0] < gini[y+1][0]:
                save = gini[y]
                gini[y] = gini[y+1]
                gini[y+1] = save
                counter += 1
            if log_var[y][0] < log_var[y+1][0]:
                save = log_var[y]
                log_var[y] = log_var[y+1]
                log_var[y+1] = save
```

```
                counter += 1
        if counter == 0:
            break
            print("I must break you")

    return gini, log_var


def gap_sort(distribution):
    dist = copy.deepcopy(distribution)
    for x in range(len(dist)-1):
        counter = 0
        for y in range(len(dist)-1):
            if dist[y][2] < dist[y+1][2]:
                save = dist[y]
                dist[y] = dist[y+1]
                dist[y+1] = save
                counter += 1

        if counter == 0:
            break
            print("I must break you")

    return dist

def calculate_gaps(gini, log_var):

    gini_local = copy.deepcopy(gini)
    log_var_local = copy.deepcopy(log_var)

    for x in range(len(gini_local)):
        dist_number = gini_local[x][1]
        for y in range(len(log_var_local)):
            if dist_number == log_var_local[y][1]:
                gap = y-x
                gini_local[x].append(gap)
                log_var_local[y].append(-gap)
                break
            else:
                pass

    return gini_local, log_var_local

def generate_gapped(sets, entries, k):
    dists = create_sets_weibull(sets, entries, k)
    gini, log_var = generate_multi(dists)
    sorted_gini, sorted_var = bubble_sort_a(gini, log_var)
    gapped_gini, gapped_var = calculate_gaps(sorted_gini,
                                             sorted_var)

    return gapped_gini, gapped_var


def avg_gap_percentile(p, gini, log_var):
    bucket_size = int(len(gini)*p)
    bucket_quant = int(1/p)
    gini_gaps = []
```

```
    log_gaps = []
    section = 0
    for x in range(bucket_quant):
        gini_sum = 0
        log_sum = 0
        for y in range(bucket_size):
            gini_sum += gini[y+section][2]
            log_sum += log_var[y+section][2]
        gini_sum = gini_sum/bucket_size
        log_sum = log_sum/bucket_size
        gini_gaps.append(gini_sum)
        log_gaps.append(log_sum)
        section += bucket_size

    return gini_gaps, log_gaps

def expected_gap_random(sets):
    gap_average = []
    for x in range(sets):
        gap_x = 0
        for y in range(sets):
            gap_x += y-x

        gap_x = gap_x/sets
        gap_average.append(gap_x)

    return gap_average

def expected_gap_percentile(p, sets):
    gaps_exp = expected_gap_random(sets)
    exp_gaps = []
    section = 0
    bucket_quant = int(1/p)
    for x in range(bucket_quant):
        x_sum = 0
        for y in range((int(sets*p))):
            x_sum += gaps_exp[y+section]
        x_sum = x_sum/(sets*p)
        exp_gaps.append(x_sum)
        section += int(sets*p)

    return exp_gaps

def gap_percentile_iterate(p, sets, iterations, k):
    gaps = []
    var_gaps_gini = []
    var_gaps_gini_grouped = []
    var_gaps_var = []
    var_gaps_var_grouped = []
    tot_gaps_gini = []
    tot_gaps_var = []
    bucket_quant_2 = int(1/p)
    for x in range(bucket_quant_2):
        tot_gaps_gini.append(0)
        tot_gaps_var.append(0)

    for x in range(iterations):
```

```
        gaps_x = []
        g_gini, g_var = generate_gapped(sets, 50 , k)
        gini_gaps_x, var_gaps_x = avg_gap_percentile(p, g_gini,
                                                     g_var)
        gaps_x.append(gini_gaps_x)
        gaps_x.append(var_gaps_x)
        gaps.append(gaps_x)
        for y in range(bucket_quant_2):
            tot_gaps_gini[y] += gini_gaps_x[y]
            tot_gaps_var[y] += var_gaps_x[y]

        everything_should_be_deepcopy = copy.deepcopy(gini_gaps_x)
        var_gaps_gini.append(everything_should_be_deepcopy)

        everything_should_be_deepcopy_2 = copy.deepcopy(var_gaps_x)
        var_gaps_var.append(everything_should_be_deepcopy_2)
        print(x)

    for x in range(bucket_quant_2):
        reorder = []
        reorder_2 = []
        for y in range(iterations):
            reorder.append(var_gaps_gini[y][x])
            reorder_2.append(var_gaps_var[y][x])
        var_gaps_gini_grouped.append(reorder)
        var_gaps_var_grouped.append(reorder_2)

    for x in range(bucket_quant_2):
        tot_gaps_gini[x] = tot_gaps_gini[x]/iterations
        tot_gaps_var[x] = tot_gaps_var[x]/iterations
        var_gaps_gini_grouped[x] = np.var(var_gaps_gini_grouped[x])
        var_gaps_var_grouped[x] = np.var(var_gaps_var_grouped[x])
    return tot_gaps_gini, tot_gaps_var, var_gaps_gini_grouped,
                                        var_gaps_var_grouped


def gap_absolute_iterate(sets, iterations, k):
    tot_gaps_gini = []
    tot_gaps_var = []
    for x in range(sets):
        tot_gaps_gini.append(0)
        tot_gaps_var.append(0)
    for x in range(iterations):
        g_gini, g_var = generate_gapped(sets, 50 , k)
        for y in range(sets):
            tot_gaps_gini[y] += abs(g_gini[y][2])
            tot_gaps_var[y] += abs(g_var[y][2])

    for y in range(sets):
        tot_gaps_gini[y] = tot_gaps_gini[y]/iterations
        tot_gaps_var[y] = tot_gaps_var[y]/iterations

    return tot_gaps_gini, tot_gaps_var

def summary_stats():
    blank = []
    ginis = []
```

```
    variances = []
    for x in range(100):
        everything_should_be_deepcopy = copy.deepcopy(blank)
        everything_should_be_deepcopy_2 = copy.deepcopy(blank)
        ginis.append(everything_should_be_deepcopy)
        variances.append(everything_should_be_deepcopy_2)
    for x in range(100000):
        g,v = generate_gapped(100, 50, 3)
        for y in range(100):
            ginis[y].append(g[y][2])
            variances[y].append(v[y][2])
        print(x)


    return ginis, variances

def summary_summary(a, b):
    uniquea = []
    uniqueb = []
    for y in range(len(a)):
        thingy = np.unique(a[y], return_counts = True)
        uniquea.append(thingy)

    for y in range(len(b)):
        thingy = np.unique(b[y], return_counts = True)
        uniqueb.append(thingy)

    #10 or more gap value, then 20 or more, then 30 or more,
    #then median, then exactly 0
    ten_or_more_a = []
    ten_or_more_b = []

    twenty_or_more_a = []
    twenty_or_more_b = []

    thirty_or_more_a = []
    thirty_or_more_b = []

    median_a = []
    median_b = []

    exactly_zero = []

    for y in range(len(a)):
        ten_or_more_a.append(np.sum(uniquea[y][1][10:]))
        ten_or_more_b.append(np.sum(uniqueb[y][1][10:]))

        twenty_or_more_a.append(np.sum(uniquea[y][1][20:]))
        twenty_or_more_b.append(np.sum(uniqueb[y][1][20:]))

        thirty_or_more_a.append(np.sum(uniquea[y][1][30:]))
        thirty_or_more_b.append(np.sum(uniqueb[y][1][30:]))

        median_a.append(int(np.median(a[y])))
        median_b.append(int(np.median(b[y])))

        exactly_zero.append((uniquea[y][1][0]))
```

```
    return median_a, median_b, ten_or_more_a, ten_or_more_b,
                                        twenty_or_more_a,
                                        twenty_or_more_b,
                                        thirty_or_more_a,
                                        thirty_or_more_b,
                                        exactly_zero

def absolute_value_list(a):
    c = copy.deepcopy(a)
    for y in range(len(c)):
        for x in range(len(c[y])):
            c[y][x] = abs(c[y][x])



    return c
```

C.6 "Graph Weibull Multi"

```
    import matplotlib.pyplot as plt

from weibull_multi import gap_percentile_iterate,
                                    expected_gap_percentile

import math

def graph_gap_vs_expected_gini(p, sets, iterations, k):
    gap_expected = expected_gap_percentile(p, sets)
    gap_simulated, blank = gap_percentile_iterate(p, sets,
                                        iterations, k)

    plt.rcParams['figure.dpi'] = 300

    ax = plt.gca()
    ax.set_ylim([-sets/2,sets/2])
    ax.set_xlim([0,1])

    plt.minorticks_on()

    #plt.xticks(ticks = list(range(0,1,p)))
    plt.tick_params(axis='both', labelsize = 8)

    plt.grid()

    p_cummulative = p/2
    for x in range(len(gap_simulated)):
        plt.scatter(p_cummulative, gap_expected[x], s=20, c="blue")
        plt.scatter(p_cummulative, gap_simulated[x], s=20, c="red")
        p_cummulative += p
        print(p_cummulative)

    plt.title("Expected vs Simulated Gap")
    plt.xlabel("Percentile")
    plt.ylabel("Gap")
```

```
    plt.legend(loc=4, fontsize='5')



    plt.show

    return


def graph_gap_var_vs_gini(p, sets, iterations, k, avg_gini,
                                avg_variance, var_gini,
                                var_variance):
    #avg_gini, avg_variance, var_gini, var_variance =
                                    gap_percentile_iterate(p,
                                    sets, iterations, k)

    plt.rcParams['figure.dpi'] = 300

    ax = plt.gca()
    ax.set_ylim([-150,150])
    ax.set_xlim([0,51])

    plt.minorticks_on()

    #plt.xticks(ticks = list(range(0,1,p)))
    plt.tick_params(axis='both', labelsize = 8)

    plt.grid()


    for x in range(len(avg_gini)-1):
        plt.scatter(x+1, avg_gini[x], s=20, c="blue")
        plt.scatter(x+1, avg_variance[x], s=20, c="red")
        #plt.plot([x+1,x+1], [avg_gini[x]-3.3*math.sqrt(var_gini[x
                                    ]/10000),avg_gini[x]+3.3*
                                    math.sqrt(var_gini[x]/
                                    10000)], color="grey")
        #plt.plot([x+.85,x+1.15], [avg_gini[x]+3.3*math.sqrt(
                                    var_gini[x]/10000),
                                    avg_gini[x]+3.3*math.sqrt
                                    (var_gini[x]/100000)],
                                    color="grey")
        #plt.plot([x+.85,x+1.15], [avg_gini[x]-3.3*math.sqrt(
                                    var_gini[x]/10000),
                                    avg_gini[x]-3.3*math.sqrt
                                    (var_gini[x]/100000)],
                                    color="grey")
        #plt.plot([x+1,x+1], [avg_variance[x]-3.3*math.sqrt(
                                    var_variance[x]/10000),
                                    avg_variance[x]-3.3*math.
                                    sqrt(var_variance[x]/
                                    100000),], color="grey")
        #plt.plot([x+.85,x+1.15], [avg_variance[x]-3.3*math.sqrt(
                                    var_variance[x]/10000),
                                    avg_variance[x]-3.3*math.
                                    sqrt(var_variance[x]/
```

```
                                        10000)], color="grey")
        #plt.plot([x+.85,x+1.15], [avg_variance[x]+3.3*math.sqrt(
                                        var_variance[x]/10000),
                                        avg_variance[x]+3.3*math.
                                        sqrt(var_variance[x]/
                                        10000)], color="grey")
        print(x)

    plt.scatter(len(avg_gini), avg_gini[x+1], s=20, c="blue", label
                                    = "Gini")
    plt.scatter(len(avg_gini), avg_variance[x+1], s=20, c="red",
                                    label = "Log Variance")


    plt.title("Gini and Log Variance Gap")
    plt.xlabel("Rank")
    plt.ylabel("Gap")

    plt.legend(loc=4, fontsize='5')



    plt.show

    return avg_gini, avg_variance, var_gini, var_variance, sets, p,
                                    iterations


def graph_gap_top_vs_bottom_gini(p, sets, iterations, k, avg_gini,
                                avg_variance, var_gini,
                                var_variance):
    #avg_gini, avg_variance, var_gini, var_variance =
                                    gap_percentile_iterate(p,
                                    sets, iterations, k)

    plt.rcParams['figure.dpi'] = 300

    ax = plt.gca()
    ax.set_ylim([-sets/20,sets/5])
    ax.set_xlim([0,len(avg_gini)/2+1])

    plt.minorticks_on()

    #conf = 1

    #plt.xticks(ticks = list(range(0,1,p)))
    plt.tick_params(axis='both', labelsize = 8)

    plt.grid()

    for x in range(int(len(avg_gini)/2)-1):

        #plt.plot([x+1,x+1], [avg_gini[x]-conf,avg_gini[x]+conf],
                                        color="grey")
        #plt.plot([x+.85,x+1.15], [avg_gini[x]+conf,avg_gini[x]+
                                        conf], color="grey")
```

```
        #plt.plot([x+.85,x+1.15], [avg_gini[x]-conf,avg_gini[x]-
                                        conf], color="grey")
        #plt.plot([x+1,x+1], [abs(avg_gini[-x-1])-conf,abs(avg_gini
                                        [-x-1])+conf], color="
                                        grey")
        #plt.plot([x+.85,x+1.15], [abs(avg_gini[-x-1])+conf,abs(
                                        avg_gini[-x-1])+conf],
                                        color="grey")
        #plt.plot([x+.85,x+1.15], [abs(avg_gini[-x-1])-conf,abs(
                                        avg_gini[-x-1])-conf],
                                        color="grey")
        plt.scatter(x+1, abs(avg_gini[x]), s=20, c="blue")
        plt.scatter(x+1, abs(avg_gini[-x-1]), s=20, c="red")



    plt.scatter(len(avg_gini)/2, abs(avg_gini[x+1]), s=20, c="blue"
                                        , label = "Top Half")
    plt.scatter(len(avg_gini)/2, abs(avg_gini[x+2]), s=20, c="red",
                                         label = "Bottom Half")

    print(x)

    plt.title("Gini Gap")
    plt.xlabel("Rank")
    plt.ylabel("Gap (Absolute Value)")

    plt.legend(loc=4, fontsize='5')



    plt.show

    return

def graph_gap_top_vs_bottom_variance(p, sets, iterations, k,
                                    avg_gini, avg_variance, var_gini,
                                     var_variance):
    #avg_gini, avg_variance, var_gini, var_variance =
                                        gap_percentile_iterate(p,
                                        sets, iterations, k)

    plt.rcParams['figure.dpi'] = 300

    ax = plt.gca()
    ax.set_ylim([-sets/20,sets/5])
    ax.set_xlim([0,len(avg_variance)/2+1])

    plt.minorticks_on()

    #plt.xticks(ticks = list(range(0,1,p)))
    plt.tick_params(axis='both', labelsize = 8)

    plt.grid()

    for x in range(int(len(avg_gini)/2)-1):
        plt.scatter(x+1, abs(avg_variance[x]), s=20, c="blue")
```

```
        plt.scatter(x+1, abs(avg_variance[-x-1]), s=20, c="red")


    plt.scatter(len(avg_gini)/2, abs(avg_variance[x+1]), s=20, c="
                                        blue", label = "Top Half")
    plt.scatter(len(avg_gini)/2, abs(avg_variance[x+2]), s=20, c="
                                        red", label = "Bottom Half")
    print(x)

    plt.title("Variance Gap")
    plt.xlabel("Rank")
    plt.ylabel("Gap (Absolute Value)")

    plt.legend(loc=4, fontsize='5')



    plt.show

    return
```

C.7 "Histogram Gaps"

```
    # -*- coding: utf-8 -*-


import matplotlib.pyplot as plt

from weibull_multi import generate_gapped

def store_gaps(iterations):
    var10 = []
    gin10 = []
    for x in range(iterations):
        gin, var = generate_gapped(1000, 50, 5)
        for x in range(20):
            var10.append(var[x][2])
            gin10.append(gin[x][2])

    return var10, gin10

def plot_gini(a):
    plt.rcParams['figure.dpi'] = 300


    plt.hist(a, color="blue", edgecolor = "black")

    plt.title("Gap for top 20 distributions (Gini)")
    plt.xlabel("Gap")
    plt.ylabel("Quantity")


def plot_var(a):
    plt.rcParams['figure.dpi'] = 300
```

```
    plt.hist(a, color="red", edgecolor = "black")

    plt.title("Gap for top 20 distributions (Variance of Logs)")
    plt.xlabel("Gap")
    plt.ylabel("Quantity")

def plot_dual(a,b):
    plt.rcParams['figure.dpi'] = 300


    plt.hist([a,b], color=["red", "blue"], label=["Variance of Logs
                                        ", "Gini"], edgecolor = "
                                        black", bins = 15, log=True)

    plt.title("Gap for top 20 distributions")
    plt.xlabel("Gap")
    plt.ylabel("Quantity")
    plt.legend(loc=1, fontsize='5')
```