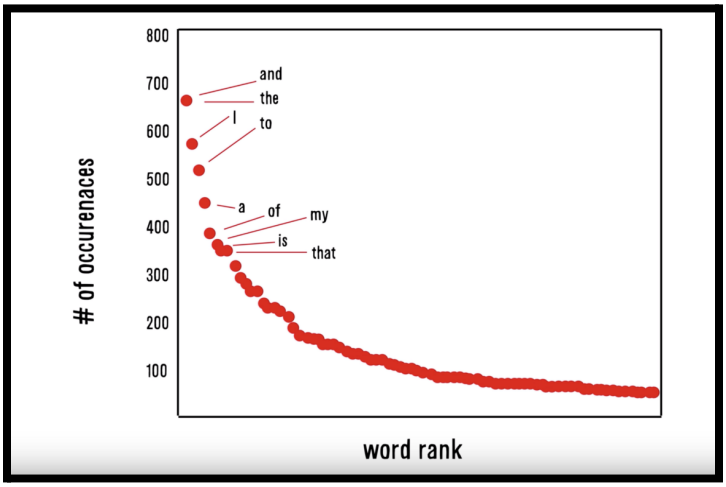
An Overview of the Pareto Principle and Zipf's Law

The Pareto Principle, colloquially known as the 80/20 rule is a common adage used in various, generally unrelated, contexts. Wikipedia defines the Pareto Principle as the idea that "for many outcomes, roughly 80% of consequences come from 20% of causes" - for instance, in the business world, a mere 20% of customers are responsible for 80% of sales.

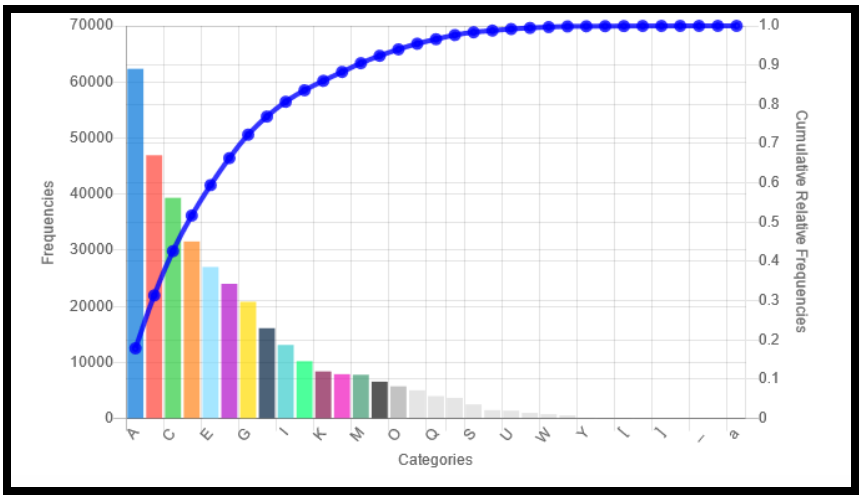
Zipf's Law, on a related note, is usually introduced in the context of words in a language. It has long been observed that a word's frequency is proportional to how often it is used in relation to other words. For instance the second most common word is used about half as often as the most common word, the third most common word is used a third as often, etc. Not only does this bafflingly apply to language, it also applies to a plethora of phenomena such as the diameter of moon craters, the rate at which we forget, and even the firing patterns of neural networks.



(1) Word rank vs frequency in "Romeo and Juliet".

The inspiration for this project was to examine whether Zipf's law was applicable to systems of people and how they interact with each other. This would require a few different things - a way to examine people's natural interactions in a natural system at scale, and an unambiguous way to examine this data. For this project I settled on scraping data from Discord, a group messaging platform. The goal was to chart a people vs. messages per person bar chart.

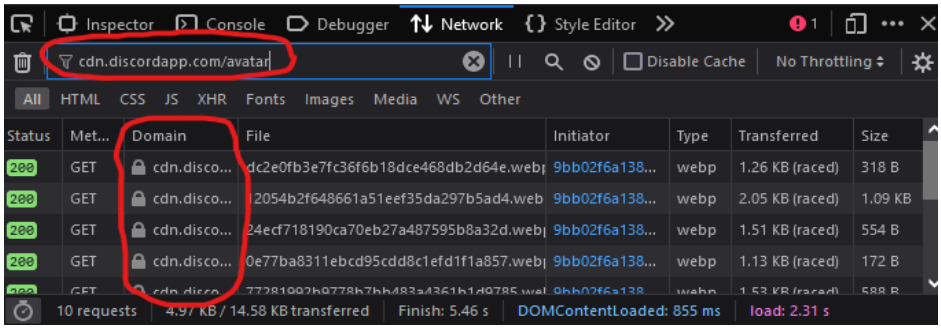
Firstly, just to check if my idea had any validity I manually collected the data for a small friend group server of which I was a part of.



(1) Initial proof-of-concept data.

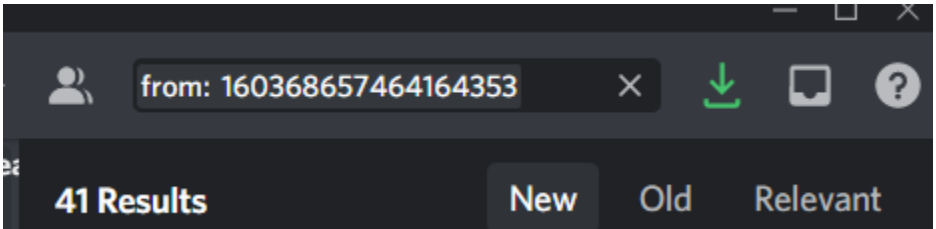
Looks pretty Zipfian, so I continued with the idea.

Now, although scraping data is usually a trivial task, the difficulty came because Discord themselves do **not necessarily want** people to have this data. Firstly, there is no easy way to collect a list of all user IDs in a server. In order to achieve this, I used a clever workaround involving scraping network traffic data. I realized that a user's ID was found in the url for their profile picture, and because we must load all profile pictures when we load a server, if we intercept and filter the network data, we can collect all user IDs.

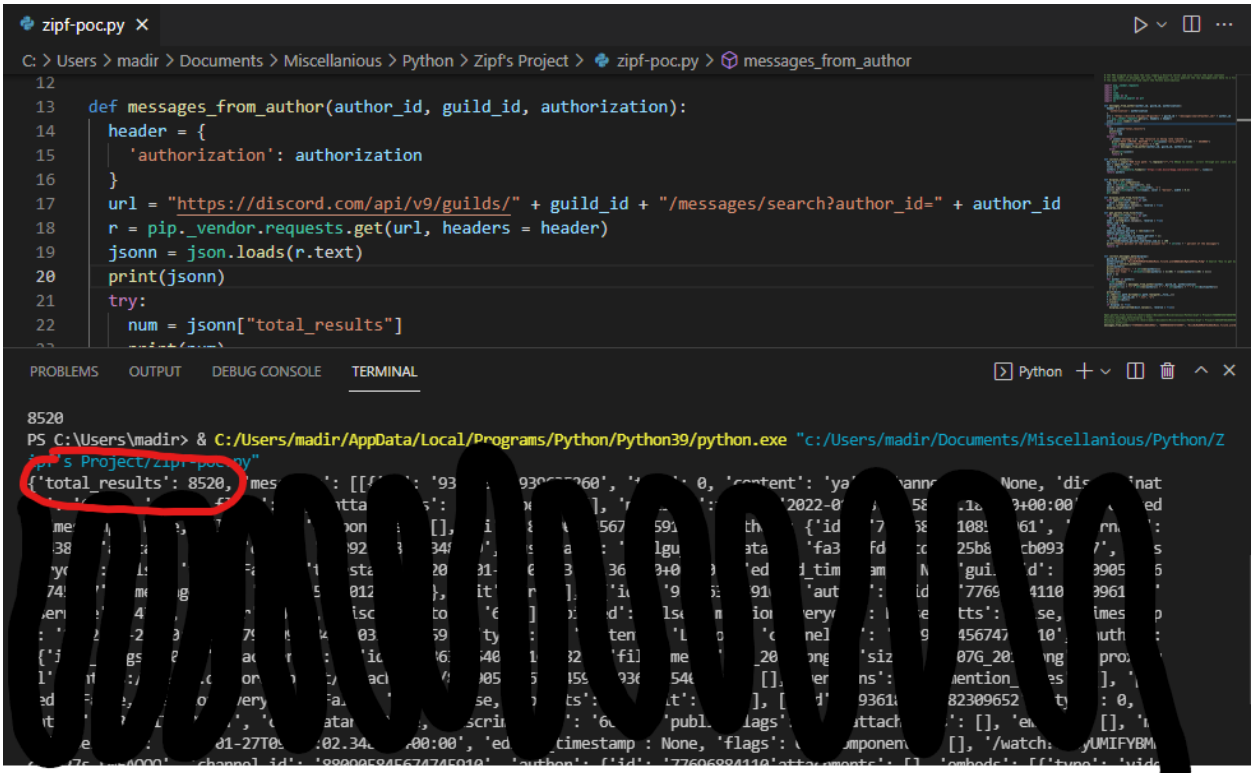


(2) Firefox's native network console.

Once we have the user ID list, the task becomes simple. Simply sending requests to the server for each user ID that mimic a user searching for messages from that user, and then scraping for "total_results" in the response, making sure to account for request rate limits set by discord, as to not set off any red flags regarding bot activity.

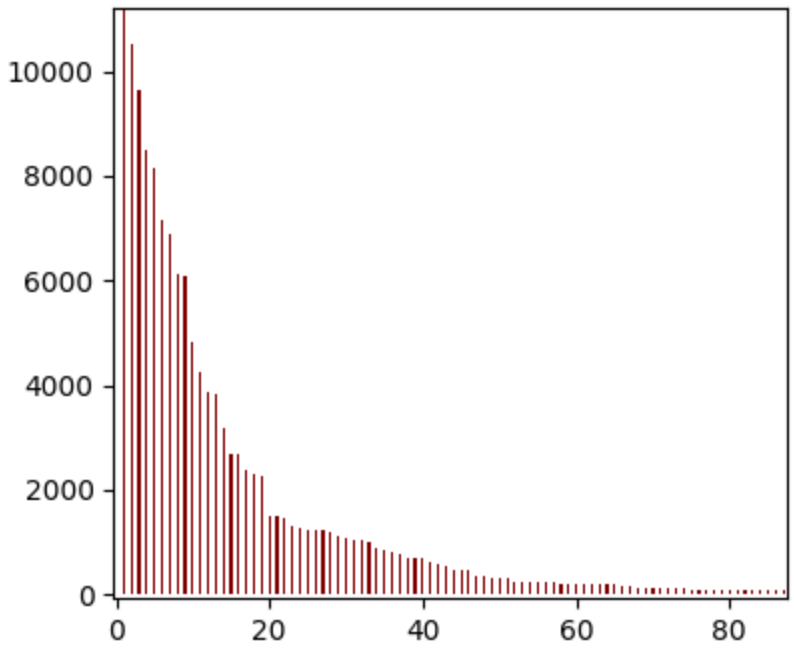


(3) Example of the input the program mimics in its request.



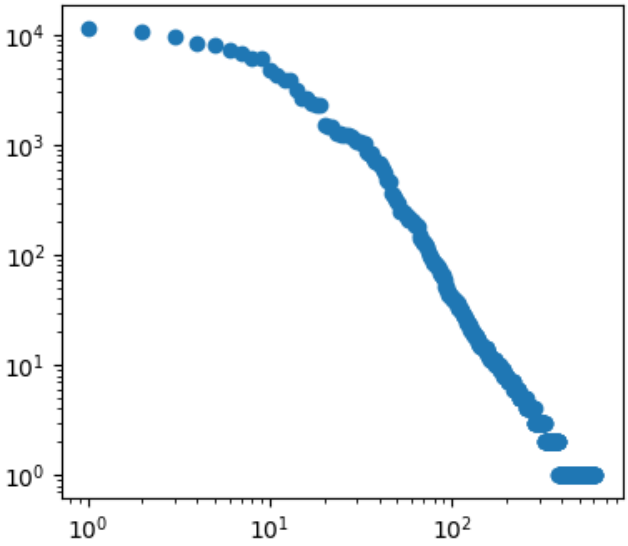
(4) Example of a response to said request. The key / value pair of interest is in the first line.

So, with that being said, let's look at the results. The first server I examined had around 600 members, and for the sake of the experiment, I made sure all servers I examined were community servers (i.e. there was no specific topic) so that they could most closely mimic a "natural" group setting.



(5) Pareto chart for the data collected for just one server.

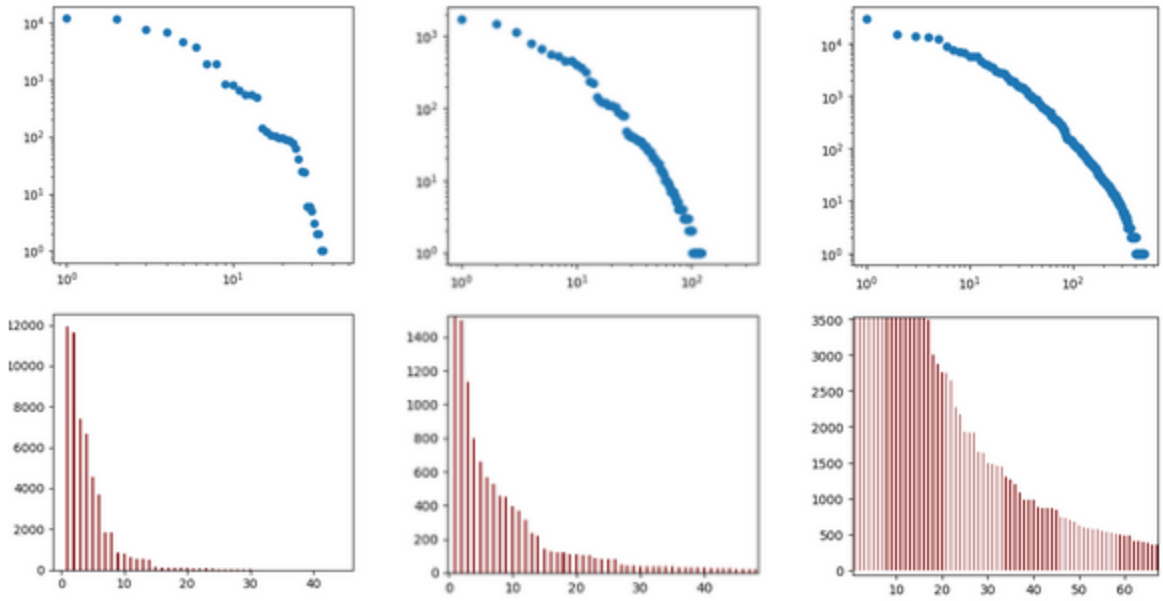
Now, although this data looks promising, we need a better way to examine it. The easiest way to do this is to create a log-log graph. As a general rule of thumb, if a dataset appears to be straight on a log-log graph, there is a definite power law relating the dataset.



(6) Logarithmic graph of the same data from (6).

So there does seem to be a power law. Excluding the first few points, which are all server bots (tools used in servers to automate certain tasks), we have a near perfect linear trend.

The same pattern shows up in nearly **every** server. The first few of which I have shown below.



(7) Data from subsequent trials.Note that the first few largest data points are Discord bots, and so it is safe to mentally exclude them from the data when analyzing linearity.

Note that we get a better estimation with more data, but even for smaller servers, it is true that the top few members (~20%) of a group make up a very large portion (~85%+) of the interactions.

If I was to continue further with the experiment, I would add a system to concretely filter out bots, and use regression to find a line of best fit on the log-log graph.