Chapter 1

Case study: Zipf's Law and Music

Max Ward (20748588)

1.1 Music as Language

Zipf's law was introduced in Chapter 5. To recapitulate, it posits that the frequency of a word is inversely proportional to its rank. The rank of a word being defined as its index in a frequency table. The canonical example of this is the words of long piece of literature, usually a novel. It is a classical example of a scale-free distribution, and is defined by $f = r^{-a}$ where f is frequency and r is rank. A frequency distribution is said to be Zipfian if $a \approx 1$. Because of this, when plotted on a log-log scale (equivalent to taking the log of both sides), a Zipfian distribution appears to be a line with slope $m \approx -1$. Though Zipf's law is typically described in terms of novels, and thus written language, it actually refers to any natural language. Natural language is generally regarded to be any non-contrived language—which is to say that its use and invention is unpremeditated. Clearly most written and spoken languages are natural languages, however, written and performed music is also a type of natural language.

Music, among other art forms, is often regarded as communication. Like other forms of human communication it has structure and rules. While I shall not give a formal definition of these, the reader may refer to `http://en.wikipedia.org/wiki/Music_theory` for a concise overview. This chapter investigates Zipf's law in music. This is done using the MIDI file type, which is a relatively simple and widely used way to encode music digitally.

Exercise 1.1. *There are specific parts of the human brain involved in language. For example, damage to Broca's area can profoundly affect ones ability to understand spoken and written language. Does an analogous region exist for processing music? (Hint: try searching for 'Amusia'.)*

1.2 The MIDI Format

The MIDI file format comprises a header and a series of one or more tracks. Ever MIDI track also contains a header, which is followed by a sequence of events. There are many event types which refer to any time related possibility. However, the only two which we consider in this investigation are the 'Note On' and 'Note Off' events. A Note On event has a time, a velocity, and a note. The time is when it is played, the velocity is the force with which it is played, and the note is the specific sound frequency of the event. It should be noted that velocity is not volume. It is possible than a MIDI instrument could have a linear velocity-volume relationship, but it could be exponential, logarithmic, or anything else. There are 128 distinct notes in the MIDI format, these are analogous to an extended piano (which has only 88 keys), with each increment indicating a semitone increase in sound frequency.

Figure of a piano keyboard here.

Parsing the MIDI file format can be difficult, as such, it is often relegated to a library. Python does not come with an easy to install midi parsing library. However, an open source effort called `python-midi` is available on Github under the MIT licence. It can be found here `https://github.com/vishnubob/python-midi`. Download or clone the repository and use the source files in your project. Once they are in the same folder, you should be able to import them like any Python library.

Exercise 1.2. *Research the MIDI format. How hard you think it would be to write your own parser? Remember, if you improve `python-midi`, submit a pull request to Github!*

Despite its simplicity, the model generates outcomes that resemble the real world. When modeling wealth with Sugarscape, a long-tailed distribution appears where some agents are vastly richer than others. Similar behavior is seen in most real economies, where a small part of the population holds a large fraction of the wealth. Extreme wealth inequality is generally considered a problem, because it means there are many people barely surviving while others are fabulously rich.

1.3 MIDI files

Wealth inequality has partly fueled a modern social movement known as the Occupy movement. The first significant Occupy protest was on Wall Street in New York City, where thousands of protesters gathered to express their dismay with the distribution of wealth, among other things. The movement's motto is "We are the 99%", reminding politicians to serve the majority, not the 1% who control more than a third of the nation's wealth. A major goal of the movement is to achieve a more equal distribution of income, which protesters hope to accomplish by implementing a more progressive tax policy.

One of the effects of taxation is to redistribute wealth from the rich to the poor. But opponents of the Occupy movement (and many fiscal conservatives) claim that high tax rates for the rich actually hurt the population as a whole. The logic is that wealthy people employ the poor, redistributing the wealth without the need for tax levies.

1.4 Words and Music

Our implementation of Sugarscape aims to study the effect of taxation on the wealth of a society. We want to show how extreme under- or over-taxation can affect the society and its individual agents, and what happens in between these two extremes. The model tests a "flat tax" system where every agent gets taxed a constant rate (say 10% of its total wealth) and the tax pool is redistributed evenly among all the agents. We recreate the original Sugarscape and expand on it with the end goal of determining whether it is possible to shrink the wealth gap without crippling the society.

1.4.1 Pygame

In the process of implementing Sugarscape, we made a GUI to better understand what was happening on the grid. The visualization of the Sugarscape is done with Pygame, a set of Python modules that allows easy graphic drawing. Pygame can blit images onto the screen (see `http://en.wikipedia.org/wiki/Bit_blit`) and it has built-in methods for handling user input like mouse clicks and button presses, making it ideal for designing games or other programs that receive a lot of input.

Below is an abbreviated version of our event loop that draws the cells in the GUI at each time step. `Sugarscape.nextstep` moves every agent forward by one time step and the rest of the code redraws the update. Redrawing the entire grid is slightly less efficient than changing existing rectangle objects but is a common convention for pygame. A square is drawn for each location, and the color of the square changes based on the amount of sugar contained there. Agents are represented by circles drawn on top of their current location.

```
def event_loop(self,sugarscape):
   while True:
      sugarscape.nextstep()
      for i in range(sugarscape.length):
          for j in range(sugarscape.width):
              loc = sugarscape.get_location(i,j)
              health_color = (0, 0, loc.get_sugar_amt()/loc.get_max_sugar())
              pygame.draw.rect(self.window, healthColor,(12*i,12*j,10,10))
      pygame.display.update()
```

Users can control certain attributes of the Sugarscape by moving sliders underneath the grid. A histogram, implemented using the matplotlib library, shows the current distribution of wealth, and text fields show certain characteristics of the distribution.

1.5 Scale-free Music

Taxation in our implementation of Sugarscape is handled with a Government object. Every ten time steps, the Government object collects a fraction of each agent's sugar reserve, then distributes the collected sugar to each agent equally. This transfer represents services provided by the government as well as explicit redistribution of wealth.

But if opponents of the Occupy movement are correct, transferring wealth from rich to poor makes society as a whole less productive. According to this theory, the rich create more wealth than the poor because they can open factories, fund research, and generally make investments into the economy.

In order to simulate this effect, we need to augment the model with a mechanism of wealth creation. We implement a simple "leave behind" feature, where agents leave some sugar behind as they leave a location:

$$leave\_behind = \frac{1}{5}\left(\frac{wealth \times N}{total\_wealth}\right)^{1.1}$$

where N is the total number of agents, $wealth$ is the amount of sugar the agent has, and $total\_wealth$ is the total sugar owned by all the agents. Agents who own a large proportion of the total wealth leave behind larger amounts of sugar, making an investment into the Sugarscape, and increasing the total wealth.

1.6 Zipf's Law and Beauty

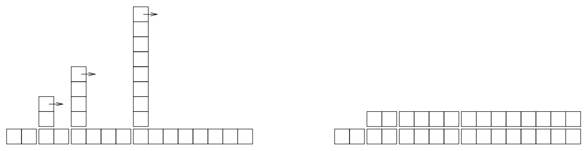

Figure 1.1: Histogram of wealth with no tax.

To compare the effect of taxation on wealth distribution, we need a metric that measures how distributed or flat a certain wealth distribution is. We use the Gini coefficient, which is often used in economics to measure the wealth gap (see `http://en.wikipedia.org/wiki/Gini_coefficient`). The Gini coefficient is between 0 and 1, with 0 the measurement of a perfectly uniform distribution, and 1 the measurement of a distribution with complete inequality.

Figure 1.1 shows a histogram describing the wealth distribution when there is no tax system in place. For most initial conditions without taxation, the Sugarscape quickly develops a long-tailed distribution of wealth, skewed to the right. In these cases, some agents die quickly, particularly in an environment with many agents or one with low sugar regrowth rate. The separation between the rich and the poor is significant, and there aren't many agents occupying the middle ground. This is seen in real life in societies where there is no tax structure and there isn't much of a middle class.

1.7 Reductionism or Holism?

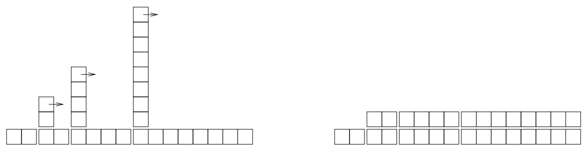

Figure 1.2: Histogram of wealth, with tax.

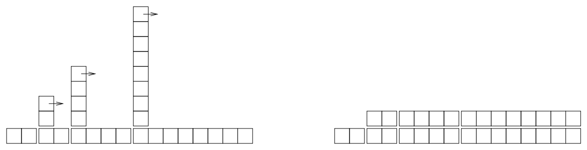

Figure 1.3: The Gini coefficient versus the tax rate.

Figure 1.2 shows the effect of a relatively high tax rate. The agents have a similar amount of sugar, and the economy has a low Gini coefficient, 0.02.

Figure 1.3 shows that higher taxes in general result in lower Gini coefficients. This makes sense, since the point of our tax system is to redistribute wealth.

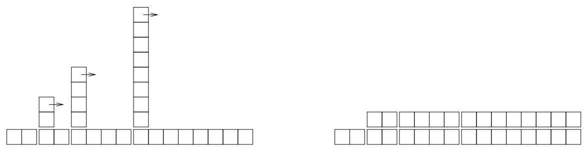

Figure 1.4: Mean wealth versus tax rate.

In this model, perfect equality comes at a price. With no taxation the mean wealth was 358; with a 20% tax rate it drops to 157. Figure 1.4 shows the effect of tax rate on wealth; mean wealth gets smaller as taxes get higher.

1.8 Instrumentalism and Music

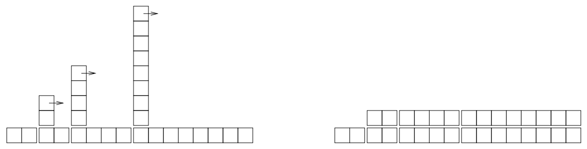

Figure 1.5: Bottom quartile value versus tax rate. At 4% the average wealth of the bottom quartile is maximized.

It's up to a society to determine its ideal wealth distribution. In our model, there is a conflict between the goals of maximizing total wealth and minimizing inequality.

One way to reconcile this conflict is to maximize the wealth of the bottom quartile. Figure 1.5 shows the mean wealth of the poorest 25% for a range of tax rates. The optimal tax rate is around 4%. At lower rates, there is more total wealth, but the poor do not share it. At higher rates, the poor have a bigger share of a smaller pie.

Of course, this result depends on the details of our Sugarscape, especially the model of productivity. But this simple model provides a way to explore relationships between wealth creation, taxation and inequality.

Exercise 1.3. *You can download our implementation of Sugarscape from `thinkcomplex.com/Sugarscape.zip`. Launch it by running* `Gui.py`*. The sliders allow you to control the parameters of the simulation. Experiment with these parameters to see what effect they have on the distribution of wealth.*

Exercise 1.4. *You can download our implementation of Sugarscape from `thinkcomplex.com/Sugarscape.zip`. Launch it by running* `Gui.py`*. The sliders allow you to control the parameters of the simulation. Experiment with these parameters to see what effect they have on the distribution of wealth.*