In this session you will:

- Learn about so-called power laws, a special case being Zipf's law
- Adjust a number of datasets to a power law
- Investigate whether words in text seem to fit power laws
- Discover Heaps' law, that tells how many words there are (usually) in a text of given size

1 Power Laws

Consider a function $y = f(x)$, that we will call a "law" relating y with x.

Example 1. You are given a bucket of some radioactive isotope. x is the number of seconds since you were given the bucket. y is the number of atoms that disintegrate at the xth second. Each atom decides independently from all other atoms whether to disintegrate in the next second (or nanosecond, or whatever). From here you can see that f will have the form $y = c \cdot a^{-x}$, where c depends on the number of atoms you were given, and a depends on the isotope. In particular, $a > 1$ determines the half-life of the isotope.

Example 2. x is number of seconds and y is the number of people entering a metro station during the next 10 minutes. You may have been told in statistics that f is given by the Poisson distribution (at least for ideal people and metro stations).

Many natural and artificial phenomena come in distributions that are neither exponential nor Poisson, but so-called power laws. Intuitively, we have y evolves like x^a, where a is a constant called the exponent of the power law. If a is positive, y is increasing, and if a is negative, y decreases.

More precisely, a power law is

$$y = c \cdot (x + b)^a$$

for three constants a, b, and c.

There is a lot of theory about why powerlaws are ubiqutous. They are often related to self-organization, fractality, complex dynamical systems, etc. The Web and social networks are full of powerlaws. So is human language, social sciences (population of cities), and natural sciences (intensity of earthquakes, relation of size and metabolism in animals).

A powerlaw with exponent -1 is called Zipf's law, after Mr. Zipf. By extension, sometimes powerlaws with negative exponents are also called Zipfian laws.

In what follows we will fit some datasets to the best powerlaw that we can find - and see that they pretty close. Note that this does NOT mean that the phenomenon generating the dataset is exactly a powerlaw. Proving seriously that a law is (or is not) a powerlaw is another matter.

2 Distribution of family names

The file `apellidos.csv` is taken from `http://www.ine.es/daco/daco42/nombyapel/apellidos_frecuencia.xls`. It gives family names in the Spanish census of 2015, sorted by frequence.

Exercise 1. Use python + matplotlib to plot the frequence of `apellidos` in decreasing order. More precisely, put the "Orden" column in the x axis and the "Apellido 1" in the y axis. (In the csv, ; is used as a separator, so you want to use `sep=';'` or `delimiter=';'` or so to read it).

Exercise 2. (joke) RESIST THE TEMPTATION to say that this is an exponential distribution "because it decrases very fast".

Is it a powerlaw? Or, can it be approximated by a powerlaw?

A trick about powerlaws is the following. Let's forget about the b parameter for a second (or equivalently assume that $b = 0$), so our powerlaw looks like

$$y = c \cdot x^a.$$

Taking logs on both sides, it becomes

$$\log y = a \cdot \log x + \log c$$

I.e., $\log y$ is a linear function of $\log x$.

Exercise 3. Tell matplotlib to use logarithmic x and y axes. (this may help: `https://matplotlib.org/examples/pylab_examples/log_demo.html`).

If our distribution is a powerlaw, this plot should be a straight line, whose slope is a and intercept is $\log c$. If we put back the b parameter, it distorts a bit the low values, so in order to estimate a and c we have to pay attention to the large values.

Exercise 4. Let's find a and c more analytically. Assume we have $\log y = a \cdot \log x + \log c$. Take two distinct large values of x, find their corresponding values of y, set up a system of two linear equations, and solve for a and c. The solution will probably not fit very well the low values of x. You can try to make it better by adding the b parameter, but don't agonize over it.

Exercise 5. All of the above can also be done pretty easily with a spreadsheet. Think how (and, optionally, do it).

3 Distribution of river lengths

The file `rivers.csv` contains info about the longest rivers in Earth.

Exercise 6. Check whether the distribution of lengths follows something like a powerlaw.

For curiosity, see the humongous outlier in basin area.

4 Words in Text

Exercise 7. Write python code that

- processes all the files in a given directory, assumed to be text files
- for each file, reads it line by line
- it removes the punctuation signs (say, .,;-?! or whatever) from the line, transforms it to lowercase,
- and maintains a dictionary of (word, frequence of the word in the files).
- Then writes the dictionary as a .csv file with three columns, rank, words, and their frequences, sorted decreasingly by frequence.

Your code might look like (but don't feel forced to follow this pattern):

```
from os import listdir
from os.path import isfile, join, isdir

def readWords(path,output_filename):
    dicc = {}
    for f in listdir(path):
        ff = join(path,f)
        print("procesing ",ff)
        for line in open(ff,"r",encoding="utf8"):
            # transform punctuation to spaces in line
            # translate line to lowercase
            for word in line.split():
                # ... something with dicc and word
    # from dicc, get list of triples (w,freq(w))
    # sort it by freq(w)
    # write it to file output_filename adding index in front

readWords(yourpath,yourfile)
```

Exercise 8. Try your code e.g. on the file collection in `novels.zip`. Proceeding as before, do the word frequencies look like a powerlaw?

Exercise 9. Now do the following test. Change your program so that for some k, it prints the number of different words in the collection after it has read k, $2k$, $3k$, $4k$, etc. words from the collection. Collect the output of pairs $\{(ik, \text{distinct words after reading } ik \text{ words})\}$. Create a plot and check whether the distribution looks like a powerlaw.

Actually $y(N)$ is called Heaps' law, and tends to follow a powerlaw with an exponent in [0..1].

5 Rules of delivery

1. No plagiarism; don't discuss your work with other teams. You can ask for help to others for simple things, such as recalling a python instruction or module, but nothing too specific to the session.

2. If you feel you are spending much more time than the rest of the classmates, ask us for help. Questions can be asked either in person or by email, and you'll never be penalized by asking questions, no matter how stupid they look in retrospect.

3. Write a short report with your results and thoughts on the Zipf's and Heaps' tests. Make it at most 2 pages, so you cannot put large tables or plots - we are not interested in these. Explain things like: What best values of (a,b,c) did you find? How did you find them - what method did you use? You are welcome to add conclusions and findings that depart from what we asked you to do. We encourage you to discuss the difficulties you find; this lets us give you help and also improve the lab session for future editions.

4. Turn the report to PDF. Make sure it has your names, date, and title.

5. Submit your work through the Racó. There will be a "Pràctica" open for each report. **Both members of the team must submit identical PDF files.**

   Deadline: Work must be delivered within 2 weeks from the end of the lab session. Late submissions risk being penalized or not accepted at all. If you anticipate problems with the deadline, tell us as soon as possible.

6 Potentially useful links and tips

- matplotlib: https://matplotlib.org/users/pyplot_tutorial.html, https://matplotlib.org/3.1.1/gallery/index.html, https://xn--llions-yua.jutge.org/upc-python-cookbook/index.html
- putting logarithmic axes: https://www.science-emergence.com/Articles/How-to-put-the-y-axis-in-logarithmic-scale
- reading csv: With pandas https://xn--llions-yua.jutge.org/python/fitxers-i-formats.html or with csv https://realpython.com/python-csv/
- punctuation: A for loop is ok, but if you want to see other methods, e.g. regular expressions, https://stackoverflow.com/questions/34860982/replace-the-punctuation-with-whitespace
- For more examples of powerlaws everywhere, https://en.wikipedia.org/wiki/Power_law#Examples
- Specifically in Big data, https://near.co/engineering/making-big-data-small-data-the-role-o php)