# Sam Cowan - DS210 Final Project

**My Dataset**

https://data.humdata.org/dataset/social-connectedness-index
I am using the `gadm1_nuts3_counties-gadm1_nuts3_counties - FB…` dataset on here.

This dataset is a weighted, undirected graph measuring the social connectedness (on facebook) between subregions of different countries. It is a clique graph, where every node connects to every other node, and the weights of the edges are the level of connectedness between its two vertices. Initially, it has over 8000 vertices.

**My Project**

My project is implementing the minimum spanning tree algorithm on this dataset, paring down the dataset to the minimum number of edges that connect each node in one component. The idea behind this was that the algorithm would identify the most meaningful connections between various countries and US counties.

**Launch Instructions**

1. `git clone` the repository. Because of the 122 mb data file, this doesn't fully work. Once it freezes on 90% on "Filtering Content," ctrl-c to stop it. Everything except for the data file should be in place
2. Download the 122 MB `cleaned.tsv` file directly from the github website, under `proj/data/cleaned.tsv`.
3. Move the downloaded file into the same place in the cloned repository
4. `Cd` into `repo_name/proj`
5. `cargo test` to verify test cases
6. `cargo run --release` to run the main code

**Methodology**
The dataset comes in at over a gigabyte, so my first step was to reduce it in size. I decided to do this by folding all subregions of countries other than the US into a singular country-wide vertex. The code in `data_cleaning.rs` does this, and if you wish to do this yourself starting with the original 1.1 GB dataset (which should be renamed `data.tsv` and placed within `proj/data/`, you can call `data_cleaning::run_cleaner()` at the start of the main function. Because the dataset is so big, this takes ~15 minutes, so I have elected to upload the already reduced (127 MB) dataset using Git LFS. Thus my code as it stands does not directly

run the `data_cleaning` functions. These create `cleaned.tsv`, which has approximately 3400 vertices, down from 8000+.

Because minimum spanning trees operate on edge lists, I created a `Graph` struct which stores this edge list and other things necessary for MST. Graphs can be initialized with `Graph::create_undirected(path)`., where the path is the reduced data (`cleaned.tsv`).

This method utilizes two highly important helper functions - `read_to_counts` and `counts_to_vector`, which read in the data and convert it to a list of edges. As part of this process, it generates the `vec_to_num_map` variable of the `Graph` struct, which allows me to convert from the country and county codes (Strings) to a set of integers that I can use to store the edges as vectors. `Create_undirected` also sorts by connectedness to put the most connected countries and counties first in the vector. This is the opposite of standard MST, since instead of minimizing distance my aim is to maximize connectedness.
The `kruskal_mst` method (the center of the project) invokes `find` and `union` to calculate the minimum spanning tree of a previously initialized graph. This returns a vector of edges, which I write (in my main) to the file `output_MST.tsv`.

SInce a graph must have only one connected component for MST to be applicable, I also have a module for calculating the number of connected components This is called as `verify_connected_components` whenever the MST is run, and panics if there is more than one component in the given graph.

**Output**
The output, which represents the minimum spanning tree of the dataset, is written to `` `output_MST.tsv` ``, in this format, of `vertex 1`, `vertex 2`, `connectedness`

The three letter codes correspond to ISO-3166 country codes as found here.
Two letter codes correspond to NUTS-3 country codes as found here.
Codes starting with USA correspond to US county FIPS codes as found here

```
LAO	THA	2125
URY	USA30019	1838
KOR	TLS	1824
PER	USA16013	1642
CY	TR	1558
MNG	USA51013	1475
CHL	HTI	1441
MAR	MRT	1383
```

```
USA20155     VNM    1314
BLR     UKR     1307
QAT     TUN     1236
GUM     JPN     666
```

**Findings:**

The minimum spanning tree maintains over 28% of the total connectedness between the countries and US counties in the graph, despite eliminating 99.95% of connections. This indicates that closest relationships between regions are extremely important compared to more distant relationships.

Without doing a full or quantitative analysis of the results of the MST, most connections are consistent with real life connections. Some are geographic, like Armenia and Azerbaijan being connected. Others, especially the somewhat rare connections between a US county and a country, are based on demographics. For example, Guatemala is connected to the New Jersey county which contains Trenton, NJ, a city with a significant Guatemalan population. Another example is Elmore County, Idaho, to Singapore. As it turns out, Singapore has an air force base here. So the result of the minimum spanning tree is not always the *most* connected places to these countries, but once their actual closest connections are taken, the next best options.

US counties tend to be far more connected with each other than full countries are with even their neighbors. This makes sense just in terms of how many geographic regions make up a country, and also given the way I compressed the data (I did not weight regions in a country by population, so the connectedness of urban areas is likely strongly underestimated).

**Graph Image (generated with Python with help from ChatGPT**

https://colab.research.google.com/drive/1O_LT_PncWdMy8pmY8jVSEoJeJ2yVZexb?usp=sharing)

- If you want to generate it yourself and download it the 16MB image so that you can see all the labels, run the code in here

**Low-resolution image**

- Every US county and every country on Facebook is represented in this graph