GerryChain Practice Day 2: Conditionals and Constraints

Daryl DeFord

June 25, 2019

1 Introduction

The main purpose of today's session is to get some more experience working with the `Partition` class in GerryChain and the `constraints` that are used to define the state space of permissible partitions. This is also a good opportunity to practice using conditional expressions in python. This is a crucial step in building a good Markov chain sampler, since there are vastly more "bad" partitions than nice ones and keeping the bad ones out of our ensemble saves a lot of wasted computational effort.

One of the overall goals of these sessions is preparing you for the YOUR STATE project. Hopefully some of the functions you design in these sessions will make it in to your presentation ☺

1.1 Python Code

The initial code for these investigations is in a separate repo in the VRDI organization. You should be able to do everything in a Jupyter Notebook or whichever other IDE you used on Friday for the original templates. The more detailed guide is a good reference for more examples.

2 Proposals

The `proposal` method lies at the heart of the discrete MCMC approach to generating districting plans. As with the updaters and constraints, each proposal is a function that takes as input a `Partition` object[1] and returns a dictionary that maps node ids to their new district assignments. Only those nodes that change assignments need to be added to the dictionary. You can see the prebuilt collection of proposal functions to get a sense of the syntax here. You can read more theoretical discussion in Section 8.4 of this and more practical discussion in Section 6 of this.

The template provided in the Day3 has a couple examples of bad proposals from the GerryChain Guide. Note that we are using grids for the template today because it makes it easier to visualize what is happening at each step - when we look at geographic partitions, the interesting information is often hidden in the small nodes in the urban areas. The basic setup for many of these proposals is similar: select a set of nodes to flip and then choose from among their neighbors. As we saw in the original Grid Templates, one of the things we are most interested in is comparing the properties of various proposals. Here are some ideas to compare the examples in the current template, as well as to recom:

- How much time does it take to run each chain?
- Which chain outputs better looking partitions?
- Which chain sees more seats values with the randomized vote data?
- What are the average boundary lengths of the final plans returned by the chains?

[1]Are you starting to see a pattern here?

3 Smarter Flips

One interesting idea is to try to use

- Make a proposal that tries to flip all the nodes that lie along a single boundary between two districts
- Make a proposal that tries to flip all of the nodes that belong to the boundary of a single district
-
- Make a proposal that tries to improve population balance
- Make a proposal that tries to improve compactness

4 Bigger Proposals

One of the main advantages of ReCom is that it changes the assignments of many nodes at once.