Statistical Computing
Simulation Activity

In-class simulation (and most of your homework)

We are interested in exploring a model of party position taking in a world of incomplete information. We will assume that the parties use a heuristic (specified below) to decide where they should locate in a two-dimensional policy space.

The goal of our simulations is to determine whether and how party behavior changes as a function of the distribution of voter preferences changes.

Simulation set up

1. Write a function to create a matrix of "voters". The matrix should contain the policy preferences of voters on each dimension. (There should be N rows and 2 columns, where N is the number of voters).

2. The function for creating voters should be able to create voter preferences using the following methods:

   - Preferences on each dimension are drawn (independently) from a standard normal distribution.
   - Preferences on each dimension are drawn (independently) from a normal distribution where the variance of each dimension can be set separately.
   - Preferences on each dimension are drawn from a uniform distribution.
   - Preferences on each dimension are drawn from a multivariate normal distribution with a specified variance-covariance matrix. (HINT: You will need a package).
   - Preferences are drawn from a mixture of (up to) three multivariate normal distributions.

3. There will be two parties located somewhere in this two dimensional space. Write a function such that voters "affiliate" with the closest of the two parties. To start out with, just pick some random positions for each party. Affiliation should be stored as some object (or added to the voter matrix).

4. Write a function that visualizes the (a) positions of the parties (b) the positions of the voters and (c) their "affiliation" (HINT: use colors to denote affiliation). Try this out for several different voter distributions just to see how it looks, and that it is working correctly.

Get things moving

1. For each iteration t of the model, the parties will locate at the "mean" position of all voters who affiliated with them in period $t-1$.
2. Write a function that chooses the "starting" position of the parties at random.
3. Write a function that "re-locates" the parties according to this heuristic.
4. Write a "master" function that runs the simulation. This should involve:
    - Sets up the voters.
    - Sets up (at random) the initial positions of the two parties.
    - Iterates across the following steps:
        - Voters cast votes
        - Parties re-locate.
    - After a specified number of iterations, the simulation stops
5. You will find it helpful (and fun) to have some visualization of this process, but note that this will slow up the speed of your simulations considerably.

Explore your model

1. Alter your function so it takes as an input:
    - Parameters for choosing the distribution of voters.
    - A random seed
    - The number of total iterations
2. The output of the function should be the vector of positions the parties take throughout the simulation.
3. Use the `expand.grid()` function to set up a data frame of possible parameters to explore. Set it up so your model will "sweep" through this parameter space in parallel.
4. Use a plot to characterize some comparative static of interest. (How does the output change as function of some input of interest? This doesn't need to be elaborate, just something.)

Expand your model

1. Alter your model so that the number of parties is an optional input. How do your results change as a result?

2. Alter your model so that voters vote "probabilistically" as some function of the distance between the two parties. (That is, allow them to make the "wrong" decision if they are nearly indifferent between the parties.) Do the results change?

3. Laver (2005) ("Policy and dynamics of political competition") explores several additional heuristics parties might use to choose their position. Add at least one heuristic to the model (i.e, the party heuristic chosen should be a parameter for each party). How does that change the behavior of the model?

4. Have some fun with these last three points and don't freak out.