Connectivity Algorithm (Rough Draft)

Description:

Ailey and company are interested in the connectivity between different regions of the brain. From a neuroscientist's' perspective, connectivity between brain regions is often measured using tractography. Our pipeline includes a Python implementation of the latest version of the CAPTURE pipeline (see here) for generating axon tracts and tracings.

However, for data that may not conform to the CAPTURE requirements, our pipeline also includes a graph theory approach for estimating connectivity.

Inputs:

The connectivity algorithm takes as inputs a graphML containing a networkX graph object (an explanation of how we generated this graphML is below[1]) and a dictionary with keys being a tuple (x, y, z) representing coordinates and values being the region in the Allen Reference Atlas.

Algorithm:

Given this epsilon-ball representation of the brain, we can quantify the connectivity between different regions of the brain. The main steps are:

1. Loading the object into networkX, we directly calculate the normalized Laplacian (networkX has a function to do so). networkX does this by calculating the unnormalized Laplacian (D - A) and then normalizing by the degree matrix raised to the power of -1/2. (note that the degree matrix is a diagonal matrix where the diagonal contains the number of vertices for every given node). In short, $L = D^{-1/2}(D - A)D^{-1/2}$. The reasoning for using a normalized Laplacian are in Chung's first chapter of the Spectral Graph Theory paper here - therein she states that the normalized Laplacian can be used equally well for regular and irregular graphs. Since many of the propositions (eg: where she describes how to interpret algebraic connectivity) used later in the book rely on her definition of the normalized Laplacian, and since she seems to have an idea of what she's doing, we went with it.

---

[1] Our pipeline takes raw brains and registers them to the Allen Reference Atlas (version 3) in order to label each point in the brain with a particular region label. From there, the pipeline finds the brightest points for regions of the brain after apply contrast-limited adaptive histogram equalization. From these "brightest points", we create a graph by defining each bright point as a vertex. Edges between vertices are drawn if neighboring vertices lie within an epsilon-ball radii preset to 20 voxels.

2. Using the normalized Laplacian, we calculate the eigenvalues and eigenvectors using built in linear algebra methods from Python. We want to save the three eigenvectors associated with the three smallest non-zero eigenvalues. This is a similar approach to using a Fiedler vector (the eigenvector associated with the 2nd smallest eigenvalue) -- we chose to use three vectors instead of just the one Fiedler vector because our original data was in three dimensions. Since the eigenvalues of the Laplacian are non-negative, the three smallest eigenvectors are just the columns of the eigenvector matrix associated with the three smallest eigenvalues. Thus, we save these eigenvectors as columns in an (n x 3) matrix.
3. From here, this "3D" Fiedler vector can be used to determine the relative connectivity of each pair of vertices. We know for each vertex in the "3D" Fiedler vector its ARA region label, so after labeling each vertex with its ARA label we can find the "average" algebraic connectivity value for each entire region.
    EG: Each vertex has three values associated with it (derived from the 3D Fiedler vector), which we can interpret as the Laplacian "x", "y", and "z" coordinates. For all vertices in a region (eg: all vertices with region label "7", for instance), we average these "x", "y", and "z" coordinates to get the average "x", "y", and "z" coordinates of the entire region.
4. We then loop through each of the averages and effectively sort each region average as a list in decreasing order of its connectivity with other regions. By "decreasing order", we mean that regions that have algebraic connectivity closest (determined by Euclidean distance between the new Laplacian "x", "y", and "z" coordinates) to the given region are outputted first, while regions with the furthest algebraic connectivities are outputted last. We'll return this as a list of lists, where the first element corresponds to the region ID and the second element as the region's algebraic connectivity. For example, for a given region ID, we output a list of lists:
    EG: For region 172 with algebraic connectivity (0.3, 0.2, 0.1):
    Output: [[128, (0.31, 0.21, 0.11)], [195, (0.28, 0.21, 0.09)], [520, (0.1, 0.2, 0.11)], …]
5. From here, we're still interested in creating some matrix showing the connectivity between each region present in the raw brain. We can create a (d x d) matrix, where "d" is the number of ARA regions present in the raw data and then fill in this matrix with the algebraic connectivities of each region with each other corresponding region using the values we generate from step 4. We could then output a matrix showing the connectivity of the different regions, and also output a list that contains in increasing order the regions of the ARA that were present in the raw brain (which also is the order of the rows/columns of the output "d x d" matrix).

Current Problems We See:

1. The epsilon ball radii leaves some components unconnected. Since the Fiedler vector/eigenvalue is only non-zero if all components are connected, we'll have to find some way to remove unconnected components from the graph.

2. From a previous conversation with Jovo, we think we learned that the order of the vertex elements in the eigenvector column matrix is the same as the order of the vertices in the original graph object. We make this assumption because we need to assign each vertex with its' region label. We weren't able to find any resources telling us if this was or wasn't true.
3. We need some way to validate if this connectivity makes sense. This will crystallize as we actually get a connectivity estimate.