**Introduction**

In my thesis, one thing I want to do is to examine whether the g-factor, fluid intelligence, and crystallized intelligence are related to state flexibility - the total number of changes in brain state over a given amount of time in different regions of interest. To do this, we are using the functional magnetic resonance imaging (fMRI) and cognitive data from the Human Connectome Project (HCP).

To represent brain activity, fMRI timeseries of activity in the brain can be averaged for each region in a set of ROIs of interest to reduce computational load. These mean timeseries can be correlated between each region over short overlapping windows of time to represent dynamically changing connections, in dynamic functional connectivity (dFC). The correlation values within each window can be used to classify the window into discrete brain states, through similarity-based k-clustering of windows together using each connection correlation as a feature and window as an observation. Changes in an individual’s windows’ affiliation with a given cluster over time represents the changes in state - the total number of changes over the entire scan can thus represent the brain state flexibility for the participant. The current project shown here uses the full brain’s regions, window width of 62 TRs slid in step sizes of 1 TR, 100 subjects, and the HCP REST1 LR and RL phase-direction MSMAll-preprocessed scans to explore the data as a pilot dataset.

The g-factor, crystallized intelligence, and fluid intelligence scores are conventionally derived from factor analysis. However, previous studies have also represented the first unrotated principal component from principal component analysis (PCA) of multiple cognitive tests with the g-factor. I selected specific tests from the HCP dataset based on a previous study analyzing the g-factor using the same dataset and conducted PCA on them for my pilot study. Previous studies have also measured fluid intelligence with the National Institute of Health (NIH)’s composite score of selected tests based on their own formula or just the Penn Progressive Matrices 24 (PMAT24). Crystallized intelligence has been measured by NIH’s composite score or just the Picture Vocabulary Test. Interestingly, the composite score and single test scores differ in the HCP (i.e. correlation is lower than expected), so they might measure different things biologically. I’m including both to examine them.

Finally, I will do multiple linear regression using the various cognitive scores as y-vars, the flexibility score as the main x-var, and age range and gender demographic information from the HCP as other x-vars to see if they improve the model - more detailed demographic information requires higher level access, which I am applying for but have not received yet.

**Module Versions**

python/3.6.8

R/3.6.3

**Steps**

1. Get the HCP dataset after obtaining all permissions based on DataLad website (http://handbook.datalad.org/en/latest/usecases/HCP\_dataset.html) using a command line command.

*cd inputs/data/*

*datalad clone https://github.com/datalad-datasets/human-connectome-project-openaccess.git*

2. Get all the resting-state MSMAll-preprocessed fMRI files. For convenience, I got all the MSMAll files in case I want to later use ones I am not using in my current analysis. I plan on using 100 3T REST1 LR and RL scans to explore the data, which I’m implementing in this project here, then the rest of the 3T scans (REST2-4) as validation later. But there’s also 7T scans which might use for other purposes, for example. I wrote a Bash script to automate this in Linux for ease of use.

*cd code*

*bash r\_MSMAll\_get.txt*

3. Use the CIFTIFY tool with the MSMAll-preprocessed fMRI files and Glasser Atlas cortical regions to find the mean fMRI timeseries of each region defined - 360 regions x 1200 TRs. I wrote a Bash script to automate this in Linux for ease of use.

*cd code*

*bash r\_meants\_do.txt*

4. Find all participants who have everything I need for my current analysis - the mean timeseries of all scans of interest (3T LR and RL REST1-4), all cognitive test scores of interests (based on previous papers), and the same pre-processing pipeline completed (based on HCP’s recommendations). Saves the intersection to a list. All missing values are removed at this step. I used Python due to my familiarity with it.

*cd code*

*python r\_subselector.py*

5. Find sliding-window dFC from the fMRI timeseries for this project - find the data for the 100-subject REST1 LR and RL pilot. Cut out overlapping windows of time and find correlation values, while denoising due to the limited number of time points per window - I tapered (i.e. make data points enter and leave the window smoothly by decreasing the intensity of edges in a slope in order for noisy spikes to not be counted as huge jumps in connectivity change) and regularized them (i.e. reducing all values by a factor to have surviving values likely be data). I used R for the Graphical LASSO L1 Regularization tool. Each R script is performed on one subject. A Slurm job script is submitted to SciNet to iterate through each subject for a given width. Results in a 360 region x (360 regions x 2278 windows) matrix per subject in this project.

*cd code*

*nano r\_sub100.txt*

* Open a text editor and copy and paste the first 100 subjects of interest from *r\_sub.txt*.
* Subjects are separated by on each line.

*sbatch r\_slid\_dFC.sh*

* Implements: *Rscript --vanilla* *r\_slid\_dFC.R <subject> 62*
  + Can select subject list and window size from command line. I selected the 100 subjects and the window size of 62 TRs for this project.
* Within, the script allows selection of several other things: I selected L1 and R1 only, 10 iterations for cross validation to find the best lambda to use with GLASSO, and the range of lambdas to test for cross validation based on where the minimum lambda is likely to be located from checking a few subjects.

6. Vectorize dFC matrices for the k-clustering function. I used Python for ease of use. Take one triangle in the correlation matrix window and stretch it out so that each column represents a unique connection, repeat this for every window and subject. Specify the connections for the ROI of the whole brain for this project - I coded in the ability to select only the connections for sets of ROIs for future ROI-limited analyses. Results in a 2279 windows x 64620 connections matrix per subject in this project.

*cd code*

*nano roi\_whole.txt*

* Open a text editor and enter the roi indices desired. Formatted with label in first cell and indices of the first window (i.e. between 0 and 359) separated by commas. In this case for the whole brain, contains the label *‘Whole’* in the first cell and the entire sequence of 0 to 359.

*sbatch r\_vectorizer.sh*

* Implements: *python r\_vectorizer.py <subject> 62 roi\_whole.txt*
  + Can select subject list, window size, and file with ROI indices from the command line. I selected 100 subjects, the window size of 62 TRs, and all regions of the brain for this project.

6. Find the elbow from k clustering to determine the best k. The script uses a variation called K-Medoids clustering due the use of Manhattan distance which is better for high dimensional data based on previous papers. Ranges of k are tested on exemplar windows of containing the highest variance in correlation values for more efficient computation, based on previous papers. I used Python for the function producing the elbow graph. Results in a 100 subject x variable exemplar indices matrix and an elbow plot for this project - saves the exemplar indices for reporting or other purposes in the future (e.g. how many exemplar windows per subject on average).

*cd code*

*python r\_stateelbow.py Whole 62 r\_sub100.txt*

* Can select ROI label based on what was set previously, window size, and subject list from command line. I selected the label for all of the regions of the brain *‘Whole’* based on the label from *roi\_whole.txt*, the window size of 62 TRs, and the 100 subjects for this project.
* Copied source code for KMeans Elbow Visualizer from the Yellowbrick package and altered parameters within to convert it into K-Medoids clustering because it didn’t have the option natively.
* Couldn’t use SciNet because it was too serial - there did not seem to be performance benefits and the memory usually grows larger than the memory allocated per node there based on my tests.

7. Find the k-clustered states based on the element-wise average of window vector values within a cluster and state flexibility based on the total number of times the subject’s window changes between clusters using the best k previously selected. I used Python for the k-clustering function. Results in a 100 subject x 1 state flexibility matrix for this project.

*cd code*

*python r\_stateflex.py 5 Whole 62 r\_sub100.txt*

* Can select k based on elbow, ROI label based on what was set previously, window size, and subject list from command line. I selected k = 5 based on the elbow, the label for all of the regions of the brain *‘Whole’* based on the label from *roi\_whole.txt*, the window size of 62 TRs, and the 100 subjects for this project.
* Couldn’t use SciNet because it was too serial, there did not seem to be performance benefits and memory can grow larger than the memory allocated per node there.

8. Perform PCA to find the first component of cognitive tests which previous papers have measured as the g-score. Also outputs factor loadings and explained variance of each component to verify validity. Used Python for the PCA function. Includes the NIH measures of fluid intelligence and crystallized intelligence based on summing their batteries and the single test measures of fluid intelligence (PMAT24) and crystallized intelligence (Picture Vocabulary Test) based on previous papers. Includes gender and age as potential regressors. Results in a 100 subject x (subject + gender + age + cognitive measures) matrix for the scores and a (cognitive test + variance explained) x principal component matrix to verify PCA validity for g.

*cd code*

*python cogscorer.py*

* Quick enough for no SciNet to be attempted to parallelize, so did not use docopt for command line arguments.
* Within, the script allows selection of subject list. I selected the 100 subjects for this project.

9. Test the linear regression model with y-var being the cognitive score, main x-var being the state flexibility measure, and other x-var being age, gender, and interactions. Used R for its statistical analysis capability. Will load in data, give statistical outputs interactively, and plot the relationships. Results in graphs for interpretation.

*cd code*

*r\_regress\_state.R*

* I designed the script to run interactively for the ease of statistics.
* Within, the script allows selection of ROI label, window width, and k. I selected the label for all of the regions of the brain *‘Whole’* based on the label from *roi\_whole.txt*, the window size of 62 TRs for this project, and k = 5 based on the output I got from the state flexibility script for this project.