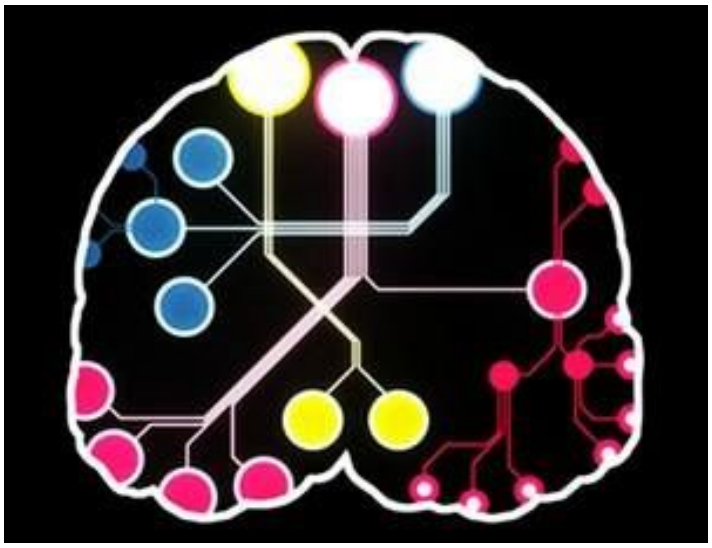
Brain Network Connectivity & Intelligence

CS170A Final Project

Alvin Vuong
[redacted]



Introduction

Why Brain Connectivity?

Connectivity across regions in the brain can be characterized as either functional (FC; correlated fluctuations in brain activity as measured by fMRI data) or structural (SC; white matter pathways as measured by diffusion-weighted MRI data, dMRI). Emerging studies suggest that the patterns of connectivity across brain regions that make up distinct cognitive networks can partially explain individual differences in behavioral traits.

Specific brain networks correlate with particular modes of behavior and cognition. The Fronto-Parietal Network (FPN) and Cingulo-Opercular Network (CON) have been shown to be associated with aspects of higher-order cognitive functions. The CON has been shown to down-regulate activity in the Default Mode Network (DMN) by acting as a cognitive "switchboard", preventing the DMN from engaging in self-referential thinking, and thus allowing one to concentrate on the task at hand

These networks are commonly associated to be important for problem solving abilities, or even fluid intelligence. One benchmark of fluid intelligence is the domain-invariant ability to identify and extrapolate patterns across distantly related ideas. It would be interesting to see if brain connectivity could predict or explain differences in individual fluid intelligence scores.

About the Dataset

The Human Connectome Project (HCP) is an inter-institutional effort attempting to map out the human connectome, a comprehensive map of neural connections in the brain, as accurately as possible in a large number of normal adults. Figuring out how the brain's connections are laid out will allow us humans to better understand how we think, why we act, and what makes each of us different from everyone else.

The 900-subject dataset is freely available online to the scientific community and was obtained using cutting-edge MR hardware. The MR data acquisition has already been optimized through refinements to pulse sequences and key pre-processing steps. The dataset mainly contains two different types of MR imaging data: diffusion imaging (structural data) and resting-state fMRI (functional data). There is also task-state fMRI and various behavioral test data as well. Any subset of the dataset can be requested, so it is possible to only grab a certain number of subjects and their tests/scans. Here is a link to the full dataset and options to grab a partial subset of it: http://www.humanconnectome.org/courses/2015/exploring-the-human-connectome.php. This contains the raw scan data for each subject as well as their behavioral attributes.

Modifications to the Dataset

Structural Connectivity (dMRI) Preprocessing

Each subject's data contains a diffusion imaging scan as well as a resting-state fMRI scan. Diffusion imaging is used to chart the trajectories of fiber bundles throughout the brain's white matter. In order to process these scans for this project, we must do some preprocessing.

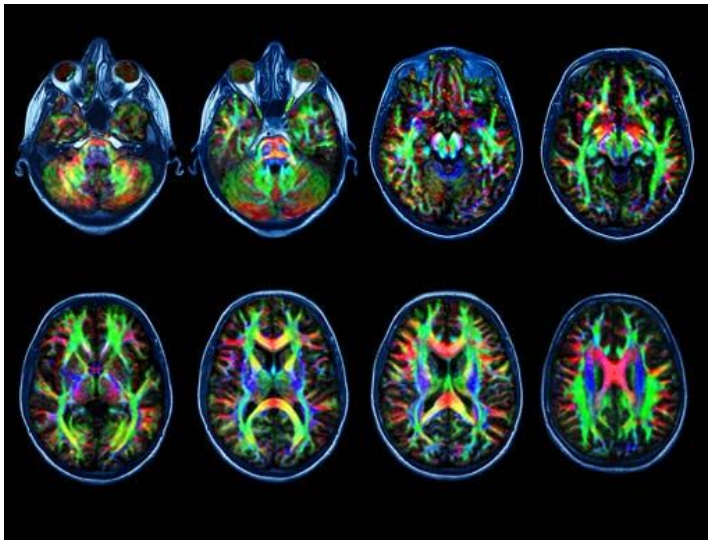


Figure 1: *Diffusion Tensor Images.* Various levels of a diffusion brain scan image; white matter tracts/pathways of the brain can be seen very clearly

The Oxford Centre for Functional MRI of the Brain (FMRIB) offers a software library (FSL) containing many algorithms for processing brain scan data. Their documentation and instructions can be found here: https://fsl.fmrib.ox.ac.uk/fsl/fslwiki. The FSL contains eddy current correction (EDDY), fitting/resizing/reslicing (FLIRT), Bayesian estimation of diffusion parameters (BEDPOSTX), and probabilistic tractography (PROBTRACKX) algorithms that we used to run on the diffusion images in order to produce a condensed structural "connectivity" map for each subject.

Each of these connectivity maps are a 264 x 264 matrix with each element in the matrix representing the connectivity between two regions of interest (ROIs) from the diffusion image. Each ROI is a part of a connected network in the brain defined by Steven E. Petersen in the following paper: Power, J. D. et al. (2011). Functional network organization of the human brain. Neuron, 72.4, 665-678. All 264 ROIs are parcellated into 14 cognitive networks that were identified by the meta-study by Power and Petersen as belonging to 14 distinct cognitive regions. Each ROI was then used as a "seed" to assess its anatomical connectivity with the rest of the brain, allowing us to infer the connectivity across all pairs of ROIs using a probabilistic tractography algorithm.

Figure 2: *Petersen Parcellation of Cognitive Networks*. Each colored node represents an ROI and each color represents one of the 14 cognitive networks as defined in the Petersen paper.

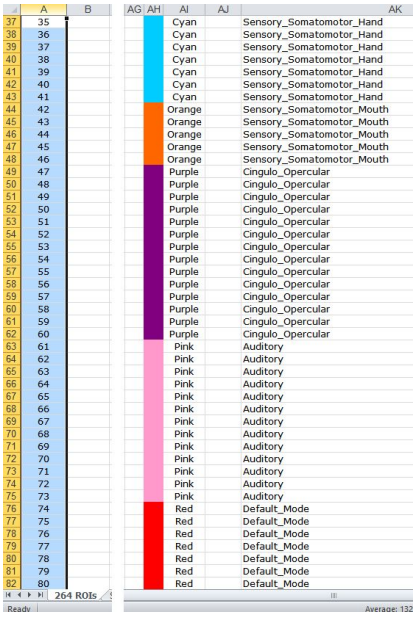
After doing this, we end up with a 264 x 264 "SC matrix" or structural connectivity matrix for each subject.

Functional Connectivity (rs-fMRI) Preprocessing

A very similar procedure was used to process the resting-state fMRI scans as well. We extracted the mean blood-oxygen-level-dependent (BOLD) contrast signal from voxels within each ROI at each time-point of the fMRI scan. Then we correlate each ROI's mean time series to every other ROI's mean time series and each correlational value represents the functional "connectivity" of a pair of ROIs. These values can be arranged in the same fashion as the structural connectivity values to create functional connectivity matrices, where each value represents the synchronicity of a BOLD signal between two different ROIs. We call this an "FC matrix" or functional connectivity matrix.

Petersen Parcellation & Network-Membership

The indexing of the Petersen parcellation scheme (which ROIs belong to which cognitive network) can be found in the `parcellations_petersen.xlsx` file. We will extract this information later to index ROIs properly when performing our analyses. Below is a screenshot of a part of the file.



	A	B	AG	AH	AI	AJ	AK
37	35				Cyan		Sensory_Somatomotor_Hand
38	36				Cyan		Sensory_Somatomotor_Hand
39	37				Cyan		Sensory_Somatomotor_Hand
40	38				Cyan		Sensory_Somatomotor_Hand
41	39				Cyan		Sensory_Somatomotor_Hand
42	40				Cyan		Sensory_Somatomotor_Hand
43	41				Cyan		Sensory_Somatomotor_Hand
44	42				Orange		Sensory_Somatomotor_Mouth
45	43				Orange		Sensory_Somatomotor_Mouth
46	44				Orange		Sensory_Somatomotor_Mouth
47	45				Orange		Sensory_Somatomotor_Mouth
48	46				Orange		Sensory_Somatomotor_Mouth
49	47				Purple		Cingulo_Opercular
50	48				Purple		Cingulo_Opercular
51	49				Purple		Cingulo_Opercular
52	50				Purple		Cingulo_Opercular
53	51				Purple		Cingulo_Opercular
54	52				Purple		Cingulo_Opercular
55	53				Purple		Cingulo_Opercular
56	54				Purple		Cingulo_Opercular
57	55				Purple		Cingulo_Opercular
58	56				Purple		Cingulo_Opercular
59	57				Purple		Cingulo_Opercular
60	58				Purple		Cingulo_Opercular
61	59				Purple		Cingulo_Opercular
62	60				Purple		Cingulo_Opercular
63	61				Pink		Auditory
64	62				Pink		Auditory
65	63				Pink		Auditory
66	64				Pink		Auditory
67	65				Pink		Auditory
68	66				Pink		Auditory
69	67				Pink		Auditory
70	68				Pink		Auditory
71	69				Pink		Auditory
72	70				Pink		Auditory
73	71				Pink		Auditory
74	72				Pink		Auditory
75	73				Pink		Auditory
76	74				Red		Default_Mode
77	75				Red		Default_Mode
78	76				Red		Default_Mode
79	77				Red		Default_Mode
80	78				Red		Default_Mode
81	79				Red		Default_Mode
82	80				Red		Default_Mode

264 ROIs

Ready

Average: 132

Figure 3: *ROI Network Indices.* In the parcellations_petersen.xlsx file, each ROI (column A) belongs to a cognitive network (column AK).

Objective

For this project, we will examine these processed connectivity matrices. We can see if brain connectivity in certain networks or regions in the brain like the DMN, FPN, and/or CON inherently predict behavioral attributes like fluid intelligence. These connectivity matrices can be used to train a Support Vector Machine classifier or used to perform Ridge/LASSO regression to predict behavioral traits would be an extremely novel approach. While the raw dataset is large (terabytes), the connectivity matrices are no more than a couple hundred megabytes, allowing for fast, efficient analysis. Most of the effort for this project will be performing the regressions properly and ensuring that the predictions are based on a properly trained model. Then, evaluating the accuracy of the predictions should be straightforward.

IMPORTANT:

MODIFIED SC AND FC DATASET (masterCell.mat) UPLOADED to Google Drive and link sent through email.

(CCLE has a 100MB restriction, and this dataset is 900MB).

All other files are in the uploaded ZIP file on CCLE.

Log

Preprocessing

The first thing to do here is to get the raw diffusion and functional scans to the SC and FC matrices that we discussed in the previous section. In the `Ordered` folder, shell and Matlab scripts can be found to do MRI processing in order to get the data to a condensed SC and FC set of matrices. These scripts are numbered in the order of what to run first. This preprocessing was mostly done for many of the subjects prior to the start of CS 170A Fall 2016, while I worked as a research assistant for the UCLA Rissman Memory Lab. Running this preprocessing stage on the raw data took months.

Since most of the filtering of noise has already been done on the scans (courteously done by the HCP), we mostly just have to worry about spatial preprocessing or standardizing the sizing/alignment for all the scans. Realigning/fitting the images comes first; this means converting the images from MNI space to MPRAGE space for our purposes. A more detailed description of these spaces and what they mean can be found here: http://www.nil.wustl.edu/labs/kevin/man/answers/mnispace.html and here: https://www.ncbi.nlm.nih.gov/pubmed/1535892. The tool we use to do this is FSL's FLIRT (image registration tool) script in order to resize our scans according to a template mask. The code is in `1_MNI_to_MPRAGE_Petersen.sh` . The documentation on FLIRT can be found here: https://fsl.fmrib.ox.ac.uk/fsl/fslwiki/FLIRT.

After doing this, we must also reslice our images into the proper resolution. `2_Reslice_Petersen.sh` does this using FLIRT as well. A good resource that explains how this all works and why can be found here: http://imaging.mrc-cbu.cam.ac.uk/imaging/Introduction_to_fMRI_2010?action=AttachFile&do=get&target=Intro_fMRI_2010_01_preprocessing.pdf.

Now, we must run BEDPOSTX on the spatially processed scans. BEDPOSTX stands for Bayesian Estimation of Diffusion Parameters Obtained using Sampling Techniques. The X stands for crossing fibres. The Bedpost algorithm essentially runs Markov Chain Monte Carlo sampling to build up distributions on diffusion parameters at each voxel of the diffusion scan. It creates all the files necessary for running a probabilistic tractography algorithm. This stage takes about 24 hours to run per scan, and parallelized code was written to run on a multiprocessor server. Documentation for BEDPOSTX can be found here: https://users.fmrib.ox.ac.uk/~behrens/fdt_docs/fdt_bedpost.html. The script used for this stage is `3_bedpostX_rerun` . Most of the subjects were already completed by the time I joined the Rissman Lab. So, my job was to find unfinished subjects and run them through this process, hence the name "rerun" on the script.

After this, we have to "seed" each ROI of a subject. This means for ROI 1, we need to create references to all of the other ROIs and supply these to the PROBTRACKX algorithm in the next step.

`4_create_target_txts_Petersen.m` does this by creating .txt files for each seed containing a reference to the other ROIs for each subject.

Then, we must run PROBTRACKX for each subject's scan. This algorithm repetitively samples from the distributions on voxel-wise principal diffusion directions, each time computing a streamline through these local samples to generate a probabilistic streamline from the distribution on the location of the true streamline. By taking many such samples, we can build up the posterior distribution on the streamline location, also known as the connectivity distribution. The result of this can be visualized below.

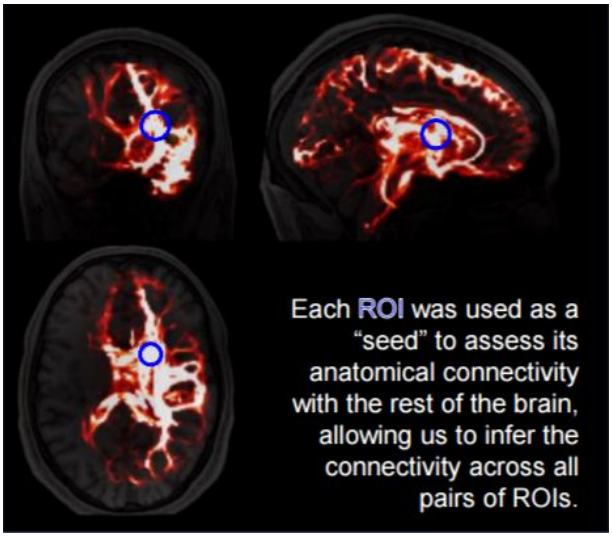


Figure 4: *Connectivity Distribution for a Single ROI Seed.* Parts of the image where it is red/white indicate areas that show high connectivity with the region circled in blue (the ROI).

A connectivity distribution was formed for each of the 264 ROIs using PROBTRACKX. Documentation for this algorithm can be found here: https://users.fmrib.ox.ac.uk/~behrens/fdt_docs/fdt_probtrack.html. This algorithm also takes a long time to run per subject, since each ROI has to be run separately. Parallelized code using a multiprocessor server was once again used to cut down processing time. This is shown in `5_probtrackx_max.sh` .

Finally, once all ROIs for all subjects have a connectivity distribution formed, we can read in those values and fill up an SC matrix for each subject, where each value in the 264 x 264 matrix is a "connectivity" value representing the connectedness between any pair of ROIs. For example, at row 2 column 3 of the matrix, the value stored there is the value associated with the connection strength between ROI 2 and ROI 3. We simply loop through each subject and loop through each of the ROIs and targets, then run `fslstats` (a tool for calculating values from image intensities) to get the connectivity distribution for each target ROI and store them in the matrix. This is done in `6_Create_Compiled_Values.m` . Now we have SC matrices for each subject!

```
>> masterCell{1}.Petersen_SC_unavg
         ans =

           scalar structure containing the fields:

             mean =

              Columns 1 through 8:

               0.0000e+000  6.9603e+000  2.5645e+000  2.5237e+001  3.6864e+000  2.2640e+001  2.0618e+001  2.4077e+001
               1.8229e+000  0.0000e+000  4.6267e+001  2.5594e+000  9.7883e+000  4.5000e+000  2.4841e+001  6.0103e+000
               1.1373e+000  2.7558e+000  0.0000e+000  2.4308e+000  6.7259e+002  5.2380e+000  2.4060e+001  3.9236e+000
               1.0495e+001  1.7353e+000  3.9818e+000  0.0000e+000  3.8844e+000  1.9169e+001  2.5417e+001  1.8645e+002
               1.2529e+000  1.4211e+000  2.3933e+002  1.8381e+000  0.0000e+000  8.8402e+000  1.3645e+001  3.8745e+000
               2.3513e+000  1.5500e+000  3.1849e+000  8.5784e+000  9.3189e+000  0.0000e+000  7.9010e+001  2.1786e+002
               1.4715e+000  2.7920e+000  1.3201e+001  8.9109e+000  1.1126e+001  4.8250e+001  0.0000e+000  1.3397e+001
               5.5569e+000  2.1028e+000  2.0930e+000  1.1796e+002  2.5141e+000  5.9674e+002  1.6207e+001  0.0000e+000
               1.4861e+000  5.6790e+000  3.9531e+001  3.5101e+001  1.0039e+001  3.0206e+001  4.7063e+001  2.7133e+001
               1.4967e+000  1.6695e+000  3.7716e+000  7.3854e+000  2.8827e+000  6.8206e+000  2.6910e+001  7.5987e+000
               2.1988e+000  2.5263e+000  2.6861e+001  2.8348e+001  9.7397e+000  2.6285e+001  3.6727e+001  2.5355e+001
               1.1979e+000  6.4875e+000  1.6185e+003  3.6722e+000  3.1351e+002  8.1877e+000  4.8763e+001  6.4401e+000
               1.3567e+000  1.3258e+000  7.4786e+000  6.3193e+001  8.9222e+000  6.0083e+001  7.0840e+001  2.8719e+001
               1.2422e+000  1.0957e+000  7.3394e+000  2.6247e+000  3.3445e+001  1.0334e+002  2.4924e+001  5.9575e+000
               1.1471e+000  1.0926e+000  4.5242e+000  2.7444e+000  1.9992e+001  1.6724e+001  2.4278e+001  5.3912e+000
               1.4620e+000  1.1371e+000  4.0744e+000  4.7490e+000  2.2269e+001  2.7142e+001  3.5986e+001  7.1160e+000
               1.2565e+000  1.0727e+000  5.4586e+000  2.3506e+000  2.1824e+001  3.7272e+002  2.6307e+001  6.1720e+000
               1.3082e+000  1.4391e+000  6.6679e+000  2.8579e+000  6.1841e+000  3.2345e+002  4.7040e+001  6.3838e+000
               1.3914e+000  1.1230e+000  7.8158e+000  1.2201e+001  4.5921e+000  2.9414e+001  9.3421e+001  1.7110e+001
               2.0283e+000  1.0172e+000  1.7002e+000  9.8166e+000  3.2014e+000  3.3451e+001  6.5776e+000  1.7254e+001
               1.1019e+000  1.0789e+000  1.3350e+002  1.5905e+000  8.8313e+000  1.7897e+001  2.8434e+002  2.4570e+000
               2.0847e+000  1.1714e+000  2.9413e+000  4.1132e+001  4.1329e+000  3.3661e+001  2.5145e+001  6.4110e+001
               1.1716e+000  1.0943e+000  1.7293e+000  1.9746e+000  4.0623e+000  9.9451e+001  1.4579e+001  4.6150e+000
               1.2614e+000  1.0000e+000  1.4940e+000  2.5528e+000  4.0385e+000  7.4367e+001  8.4120e+000  4.6324e+000
               1.2515e+000  1.2010e+000  2.2866e+001  1.6092e+001  5.1905e+000  1.1411e+001  1.6863e+002  1.6182e+001
               1.0918e+000  1.0865e+000  4.5287e+001  1.9120e+000  5.2183e+000  7.8284e+000  2.3979e+001  2.8018e+000
               1.3140e+000  1.1000e+000  1.3668e+000  2.1934e+000  4.0643e+000  4.1212e+001  6.9336e+000  4.0410e+000
               1.5074e+000  1.1625e+000  2.8087e+001  7.7907e+000  5.8620e+000  2.4353e+001  4.1723e+002  9.4799e+000
               1.0441e+000  1.0946e+000  2.7809e+001  2.1477e+000  6.6605e+000  5.8120e+000  7.8126e+001  2.0298e+000
               1.1242e+000  1.0690e+000  1.4437e+000  3.3374e+000  3.2741e+000  2.2753e+001  4.9731e+000  4.3610e+000
               1.1554e+000  1.0656e+000  1.9495e+001  2.0191e+000  5.5965e+000  5.8925e+001  1.6688e+002  3.4279e+000
               1.8278e+000  1.2338e+000  8.9313e+000  2.9357e+001  5.7566e+000  2.4807e+001  2.2978e+002  3.6896e+001
               1.2063e+000  1.1111e+000  1.7821e+000  2.7868e+000  2.4620e+000  1.4110e+001  6.0929e+000  5.1752e+000
```

Figure 5: *Petersen Unaveraged Structural Connectivity Matrix*. What part of the Petersen_SC_unavg matrix looks like. The diagonal values are zero, and each value represents the anatomical ROI to ROI connection strength.

The FC matrices for each subject were already created by the Rissman Lab, and it was only a matter of loading them in when performing analyses later. Although, we do compile all the FC matrices into a large dataset in `24_reorg.m`, which is the script that I used to compile all the SC and FC matrices into a single .mat file: `masterCell.mat` . This file contains a 1x505 cell array with each index representing a different subject. Each subject's SC and FC matrices are stored here.

```
fieldnames(masterCell{1})

      ans =
      {
        [1,1] = ID
        [2,1] = Petersen_SC_unavg
        [3,1] = Petersen_SC_avg
        [4,1] = Petersen_FC
        [5,1] = Gordon_SC_unavg
```

```
}
```

The above output shows the field names for each index in `masterCell`. For each subject, we can access the subject's ID and Petersen SC and FC matrices (as well as Gordon, but we don't use these in this project). Note that the SC and FC matrices have values of zero along the diagonal; this is because we don't run PROBTRACKX from an ROI seed to itself.

SC Matrix Averaging

Because the BEDPOSTX and PROBTRACKX algorithms can be ran from either side of an ROI pair, we can also average the two directional values to get a mean connectivity value between two ROIs. Or we can also keep the directional value for either direction.

For example, ROI 2 could have been the seed when calculating the connectivity to ROI 20, but then ROI 20 was also the seed at some point when calculating the connectivity to ROI 2. We can keep these values distinct or we can average them and create an averaged SC matrix. This is what the script `7_average_connectivity.m` does. It creates an averaged SC matrix for each subject based off of their unaveraged SC matrix. This resulting matrix is symmetrical.

FC Matrix Averaging

FC matrices are already symmetrical. They do not need to be averaged. This is because when calculating the correlation of the mean time courses between each ROI, the resulting value is the same both ways.

Behavioral Data

Now that we have all of the SC and FC matrices in place, we should organize the behavioral data in a useful fashion. The `HCP_Q3_Release_Appendix.pdf` file contains information on all the behavioral data that was acquired for each subject. Ranging from dexterity test scores to intelligence scores, we used this to identify which subject attributes we wanted for our analyses.

The `hcp_behavioral.csv` file contains all the subject behavioral data. The .csv file has been converted to an .xls file for easier use. Note that the .xls file had to be cleaned in the following ways:

- TRUE/FALSE values had to be treated as text instead of generic types in Excel
    - This was done by selecting columns (one-by-one) and doing Data → Text-to-Columns → Next → Next, select 'Text' data format, finish.
- Some behavioral_var values are invalid field names. These were altered accordingly:

- RS-fMRI_Count
- Non-TB_Compl
- NEO-FFI_Compl
- ASR-Syn_Compl
- ASR-DSM_Compl
- FS_L_Non-WM_Hypointens_Vol
- FS_R_Non-WM_Hypointens_Vol

- Subject IDs must begin with a character. See code for handling.

The `8_Create_Behavioral_Struct.m` script parses the now cleaned .xls file and stores the data into a 542 x 1 struct that contains all the behavioral values for every subject.

```
all_behave =

  scalar structure containing the fields:

    Subject100307 =

      scalar structure containing the fields:

        Release = Q1
        Gender = F
        Age = 26-30
        Full_MR_Compl = TRUE
        T1_Count =  1
        T2_Count =  1
        RS_fMRI_Count =  4
        Full_Task_fMRI = TRUE
        fMRI_WM_Compl = TRUE
        fMRI_Gamb_Compl = TRUE
        fMRI_Mot_Compl = TRUE
        fMRI_Lang_Compl = TRUE
        fMRI_Soc_Compl = TRUE
        fMRI_Rel_Compl = TRUE
        fMRI_Emo_Compl = TRUE
        dMRI_Compl = TRUE
        dMRI_3T_ReconVrs = r227
        fMRI_3T_ReconVrs = r177
        MEG_AnyData = TRUE
        MEG_FullProt_Compl = FALSE
        MEG_HeadModel_Avail = TRUE
        MEG_CortRibn_Avail = TRUE
        MEG_Anatomy_Avail = TRUE
        MEG_Anatomy_Compl = TRUE
        MEG_Noise_Avail = TRUE
        MEG_Noise_Compl = TRUE
        MEG_RS_Avail = TRUE
        MEG_RS_Compl = TRUE
        MEG_WM_Avail = TRUE
        MEG_WM_Compl = TRUE
        MEG_StoryMath_Avail = TRUE
        MEG_StoryMath_Compl = TRUE
- less -- (f)orward, (b)ack, (g)uit
```

Figure 6: *Behavioral Data.* The data structure can be accessed by subject ID number and then the corresponding behavioral value of interest.

Because we want to look at intelligence scores specifically, we are mostly interested in the PMAT24_A_CR, PMAT24_A_SI, and PMAT24_A_RTCR values. Although for our purposes, we will only be really looking at PMAT24_A_CR.

Raven's Progressive Matrices

The Raven's Progressive Matrices (RPM), a pattern completion task, is one widely used measure of general fluid intelligence. The Penn variant, or Penn Progressive Matrices (PMAT), is a shorter version of the test with 24 items, arranged in order of increasing difficulty. In order to assess the relationship between brain connectivity and fluid intelligence, we can examine the subjects' scores on this test. Subjects taking this test were presented with a series of evolving patterns. They were tasked with selected the correct next pattern from a list of options.

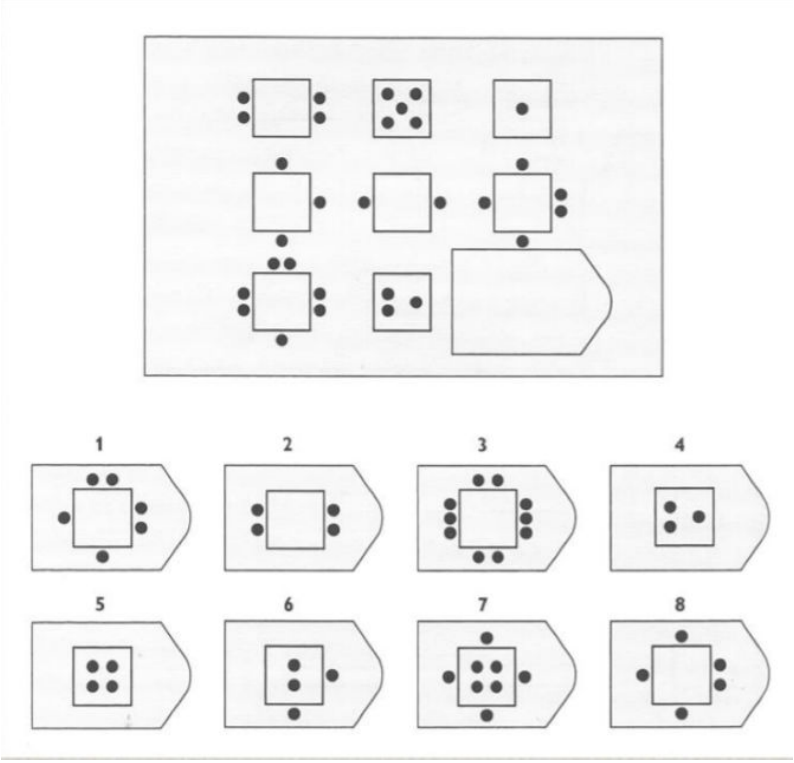


Figure 6: *Raven's Progressive Matrices.* A sample question from a progressive matrices test. The test taker must identify the correct pattern out of the options that corresponds to the missing spot.

The identifier in our behavioral data structure for this test is PMAT24_A_CR (number of correct responses out of 24). There are also PMAT24_A_SI (total number of skipped items) and PMAT24_A_RTCR (the median reaction time for correct responses), but for our purposes we only really are interested in PMAT24_A_CR, to see if brain connectivity can accurately predict overall score on the PMAT.

Network Indexing

We have to also extract the Petersen parcellations and the corresponding ROI indices of each network. The petersen_parcellation.xlsx file had to be cleaned in the following way:

- Some network names were altered so that they could comply with field name restrictions.

The `9_Create_Petersen_Networks_Struct.m` script stores the data into a struct for later use. The resulting file is stored as `Petersen_Networks.mat` . The following is an example of one of the network's, the Default Mode Network (DMN), indices.

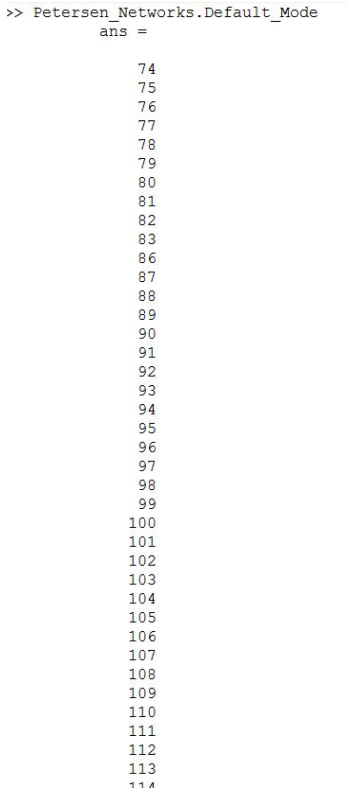


Figure 7: *Default Mode Network ROI Indices.* Example of what ROI indices correspond to the Default Mode Network as identified by Petersen in the cognitive networks meta-analysis study.

These indices will later be used to build a feature set of connectivity values to train a classifier that will be used to predict PMAT scores. Each network contains an array of parcel IDs in the network.

Feature Set Construction

There are multiple ways to build a feature set to train our model from the connectivity data. We can use solely SC data, solely FC data, or some combination of both, a single network, two networks,

some combination of networks, etc. Therefore, we need to define how any of our feature sets are constructed. (All of these scripts are implemented as functions in the Scripts folder, starting from `10_features_structural.m` to `18_features_sf_Interact.m`).

Our feature sets were indexed and manipulated as follows:

- Intra- vs. Inter-Network Connectivity
  - Within a single network (denoted as wX)
  - Bidirectionally across two networks (bXY)
  - Unidirectionally across two networks (uXY)

- SC vs. FC
  - Structural Only (s)
  - Functional Only (f)
  - Masking the functional data to only include the values that showed either the highest or lowest structural connectivity (f_hs, f_ls)
  - Masking the structural data to only include the values that showed either the highest or lowest functional connectivity (s_hf, s_lf)
  - Interaction by multiplying the structural and functional connectivity values (sxf)

Single Intra-Network Connectivity

When only considering a single network, we simply look up the averaged pairwise connectivity values between each pair of ROI indices within the network. This is essentially a handshake problem.

A set of ROI indices are first obtained by accessing our Petersen_Network struct to grab the corresponding array of ROI indices. Then, we grab the corresponding ROI to ROI pairwise connectivity values for all subjects that have the associated data. There will be N x (N-1) / 2 ROI pairs (closed form solution to handshake problem). And so our feature set, will be (N x (N-1) / 2) x (# of valid subjects) for any single-network-based feature set. A simplified version of the code is shown below.

```
% Retrieve network ROIs
roiList = Petersen_Networks.(net1);

%Initialize a matrix to hold pairwise connectivity values
sizeROI = length(roiList);
num_connections = (sizeROI*(sizeROI-1)/2); % Number of pairs of ROIs
feature_set = zeros(num_connections, length(subjs));

% For each subject
for s = 1:length(subjs)
    % Grab info for subject
    file_str = char(subjs(s));
```

```
    subjectID = file_str(6:end-4);

    % Get subject's data
    try
        load([structural_avg_path 'Subj_' subjectID '_avg.mat']);
    catch
        % Subject's data is partially missing. Skip.
        continue;
    end

    % Find connectivity values for each pair...
    % ...Within network
    n = 1;
    connectivities = zeros(num_connections, 1);
    for i = 1:(length(roiList)-1)
        for j = (i+1):length(roiList)
            connectivities(n) = mean_non_zero_avg(roiList(i),
roiList(j));
            n = n + 1;
        end
    end

    % Sort by descending order and set return value
    feature_set(:, s) = sortrows(connectivities, -1);
end
```

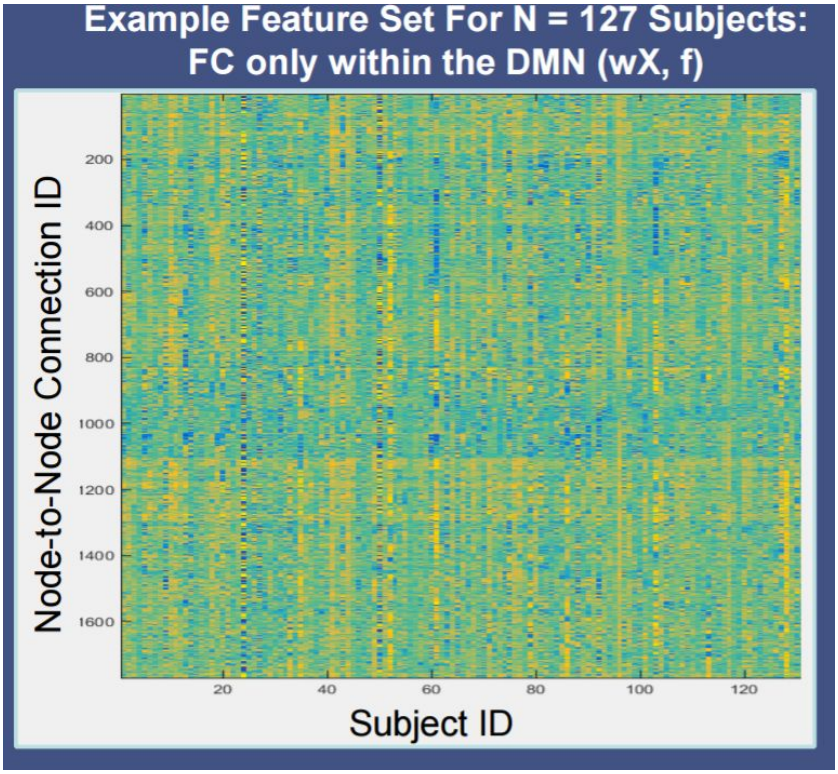


Figure 8: *Feature Set Visualization*. This is a visualization of an example feature set that will be used by our model in predicting PMAT scores. The redder a spot is, the higher the connectivity value.

Two Intra/Inter-Network Connectivity

When considering two networks, the complexity of our problem increases significantly. Firstly, we define "intra-network" connectivity as only examining the connections within a network. Secondly, we define "inter-network" connectivity as only examining the connections across two networks.

In this sense, when we have two networks X and Y, we can look at the following combination schemes:

- Within network X, within network Y
- Within network X, within network Y, across from X to Y
- Within network X, across from X to Y
- Within network Y, across from X to Y
- Across from X to Y

Also, we can either look at the averaged SC values or the specific unaveraged SC values, when examining inter-network connectivity.

We denote any of these schemes by labelling our results later with a string with the following format:

Network1_and_Network2_Scheme

An example label would be:

Default_Mode_and_Cingulo_Opercular_amXY_wX_wY

where amXY_wX_wY means:

- within Default Mode
- within Cingulo Opercular
- “across mutual” (averaged) between Default Mode and Cingulo Opercular

While another example label would be:

Default_Mode_and_Cingulo_Opercular_aoXY_wX

where aoXY_wX means:

- within Default Mode
- “across one-way” (unaveraged) between Default Mode and Cingulo Opercular

The number of ROI pairs for amXY or aoXY schemes are simply the product of the sizes of the two networks. There are N1 x N2 pairs when counting network to network ROI pairs.

This is added with the number of ROI pairs with wX or wY. wX will add (N1 * (N1-1) / 2) pairs to the total and wY will add (N2 * (N2-1) / 2) pairs.

So a feature set with the scheme: amXY_wX_wY will have (N1 * N2) + (N1 * (N1-1) / 2) + (N2 * (N2-1) / 2) pairs of ROI connections.

Structural and Functional Connectivity Interaction

On top of the aforementioned network combinations, we also have both SC matrices and FC matrices. These can also be combined in the following ways:

- Structural Only (s)
- Functional Only (f)
- Masking the functional data to only include the values that showed either the highest or lowest structural connectivity (f_hs, f_ls)
- Masking the structural data to only include the values that showed either the highest or lowest functional connectivity (s_hf, s_lf)
- Interaction by multiplying the structural and functional connectivity values (sxf)

Each of these schemes are implemented in different function scripts (as mentioned earlier, #10 to #18 in the Scripts folder).

The structural only and functional only schemes are straightforward. Structural only means only the SC matrices are used; functional only means only the FC matrices are used.

For the masking schemes, either SC or FC ROI connections are sorted from highest to lowest and a 50% cutoff is used to select the ROI pairs with the highest or lowest connectivity values to mask on the

opposing data. “f_hs” means that the top 50% of structural connections are used to select the functional data corresponding to those ROI pairs.

For the interaction by multiplying scheme, SC and FC values are multiplied together and treated as normal afterwards.

Cross-Validation

Now that we have our feature sets ready, we can now begin to train our model. We use a cross-validation leave-one-out procedure to select training data and testing data sets. This means that we choose one subject out of the total possible subjects to use for testing, and the rest for training. We build a model on n-1 subjects’ feature sets and predict the n^{th} subject’s PMAT score, and repeat n times, changing the n^{th} subject each time.

Model Selection

We use three different regression techniques: Ridge Regression, Support-Vector Regression (SVR), and LASSO Regression. Each were assessed for highest accuracy of prediction overall for all subjects. We loop through each feature set to assess predictive power of each model (we discuss how we do that in the Results section). Three different scripts: `19_NIQ_full_Ridge.m`, `20_NIQ_full_SVR.m`, and `21_NIQ_full_Lasso.m` were used. These scripts were controlled using a master script `22_Workspace_Running.m`

A sample of the code from `NIQ_full_SVR.m` is below:

```
function NIQ_full_SVR(behavioral_var,conn_type,Network_1,Network_2)

% Loop over types of analyses
types = {'s', 'smof_upper', 'smof_lower', 'fmos_upper', 'fmos_lower', 'Interact', 'f'};
val_types = {'M', 'V'};
percent = .5;
for t = 1:size(types, 2)
    for tt = 1:size(val_types, 2)
        % Set variables of interest
        switch nargin
            case 3
                fprintf('Trying intranetwork...\n');
                fprintf('%s...\n', types{t});
                if t == 1
                    [subjs_used,classification_patterns] = features_structural(conn_type,
val_types{tt}, Network_1);
                elseif t == 2
                    [subjs_used,classification_patterns] = features_smof_upper(conn_type,
val_types{tt}, percent, Network_1);
                elseif t == 3
                    [subjs_used,classification_patterns] = features_smof_lower(conn_type,
val_types{tt}, percent, Network_1);
                elseif t == 4
```

```
                    [subjs_used,classification_patterns] = features_fmos_upper(conn_type,
val_types{tt}, percent, Network_1);
                elseif t == 5
                    [subjs_used,classification_patterns] = features_fmos_lower(conn_type,
val_types{tt}, percent, Network_1);
                elseif t == 6
                    [subjs_used,classification_patterns] = features_sf_Interact(conn_type,
val_types{tt}, Network_1);
                elseif t == 7 && tt ~= 2
                    [subjs_used,classification_patterns] = features_functional(conn_type,
Network_1);
                end

                if t == 7 && tt == 2
                    continue
                end

                fprintf('Retrieved feature set.\n');
            case 4
                fprintf('Trying internetwork...\n');
                fprintf('%s...\n', types{t});
                if t == 1
                    [subjs_used,classification_patterns] = features_structural(conn_type,
val_types{tt}, Network_1, Network_2);
                elseif t == 2
                    [subjs_used,classification_patterns] = features_smof_upper(conn_type,
val_types{tt}, percent, Network_1, Network_2);
                elseif t == 3
                    [subjs_used,classification_patterns] = features_smof_lower(conn_type,
val_types{tt}, percent, Network_1, Network_2);
                elseif t == 4
                    [subjs_used,classification_patterns] = features_fmos_upper(conn_type,
val_types{tt}, percent, Network_1, Network_2);
                elseif t == 5
                    [subjs_used,classification_patterns] = features_fmos_lower(conn_type,
val_types{tt}, percent, Network_1, Network_2);
                elseif t == 6
                    [subjs_used,classification_patterns] = features_sf_Interact(conn_type,
val_types{tt}, Network_1, Network_2);
                elseif t == 7 && tt ~= 2
                    [subjs_used,classification_patterns] = features_functional(conn_type,
Network_1, Network_2);
                end

                if t == 7 && tt == 2
                    continue
                end

                fprintf('Retrieved feature set.\n');
        end

        %% Behavioral Data

        % Initialize paths
        behavioral_dir = '/space/raid6/data/rissman/Nicco/NIQ/Behavioral/';

        % Grab subjects' behavioral data
```

```
        fprintf('Grabbing behavioral data...\n');
        cd(behavioral_dir);
        load('all_behave.mat');

        % Initialize a vector for behavioral values for each subject
        temp_behav = zeros(1, length(subjs_used));

        % Make sure non-empty data
        init_subjs_used_size = length(subjs_used);
        s = 1;
        while (s <= init_subjs_used_size)

            % Grab info for subject
            subjectID = subjs_used(s);

            % Grab subject's behavioral value
            temp_behav(s) = all_behave.(['Subject' num2str(subjectID)]).(behavioral_var);

            % If NaN, remove subject from subjs_used and corresponding features
            if (isnan(temp_behav(s)))
                init_subjs_used_size = init_subjs_used_size - 1;
                if (s == 1)
                    subjs_used = subjs_used(2:end);
                    temp_behav = temp_behav(2:end);
                    classification_patterns = classification_patterns(:, 2:end);
                else
                    subjs_used = [subjs_used(1:s-1) subjs_used(s+1:end)];
                    temp_behav = [temp_behav(1:s-1) temp_behav(s+1:end)];
                    classification_patterns = [classification_patterns(:, 1:s-1)
classification_patterns(:, s+1:end)];
                end
            else
                s = s + 1;
            end
        end

        % Cross Validation
        fprintf('Finding selectors...\n');
        behav_vector = temp_behav;
        condensed_regs_of_interest = 1:length(behav_vector);
        [selectors] = enforce_forced_choice(condensed_regs_of_interest);

        %% Classification
        fprintf('Beginning Classification...\n');
        svr_acts = zeros(1, size(selectors, 2));

        for n=1:size(selectors,2)
            current_selector = selectors{n};
            train_idx = find(current_selector == 1);
            test_idx  = find(current_selector == 2);

            train_labels = behav_vector(:,train_idx);
            test_labels = behav_vector(:,test_idx);

            train_pats = classification_patterns(:,train_idx);
            test_pats = classification_patterns(:,test_idx);
```

```
            fprintf('SVR for n = %d \n', n);

            % SVR
            [model]=svmtrain_NR(train_labels', train_pats','-s 4 -t 0 -c 1 -q');
            [svr_acts(n)] = svmpredict_NR(test_labels',test_pats',model,'-q');
        end

        fprintf('Finishing Classification...\n');

        SVR = corr(svr_acts', behav_vector');

        % Output results
        fprintf('Saving results...\n\n');
        switch nargin
            case 3
                if tt == 1
                    save_file = ['/space/raid6/data/rissman/Nicco/NIQ/Results/EXPANSION/'
Network_1 '_' conn_type '_' behavioral_var '_n' num2str(length(subjs_used)) '_' types{t}
'_SVR_Mean.txt'];
                elseif tt == 2
                    save_file = ['/space/raid6/data/rissman/Nicco/NIQ/Results/EXPANSION/'
Network_1 '_' conn_type '_' behavioral_var '_n' num2str(length(subjs_used)) '_' types{t}
'_SVR_Volume.txt'];
                end
            case 4
                if tt == 1
                    save_file = ['/space/raid6/data/rissman/Nicco/NIQ/Results/EXPANSION/'
Network_1 '_and_' Network_2 '_' conn_type '_' behavioral_var '_n' num2str(length(subjs_used))
'_' types{t} '_SVR_Mean.txt'];
                elseif tt == 2
                    save_file = ['/space/raid6/data/rissman/Nicco/NIQ/Results/EXPANSION/'
Network_1 '_and_' Network_2 '_' conn_type '_' behavioral_var '_n' num2str(length(subjs_used))
'_' types{t} '_SVR_Volume.txt'];
                end
        end

        header = {'SVR'};
        data = SVR;

        save_data_with_headers(header,data,save_file);
    end
end
```

Results

To obtain a metric of model accuracy we correlated the array of predicted PMAT scores with the matched array of subjects' actual PMAT scores. This results in an R^2 value, the percent of the variance in PMAT score (and indirectly fluid intelligence) that our model was about to account for.

All of these results were not significant for the Ridge method, and LASSO would not have finished running in the duration of the quarter. SVR had the highest accuracy out of Ridge and SVR. Output for

each scheme can be found in the Results/EXPANSION folder as text files labelled with the respective networks and feature set scheme used. Also, by varying the Ridge penalty parameter and SVR cost parameter, we were able to maximize the effectiveness of each model (check in `Other Useful Scripts/NIQ_full_Ridge_Pen_Varied.m` and `NIQ_full_SVR_Varied.m`).

The following graphs were created using the maximal R^2 values from each scheme using SVR.

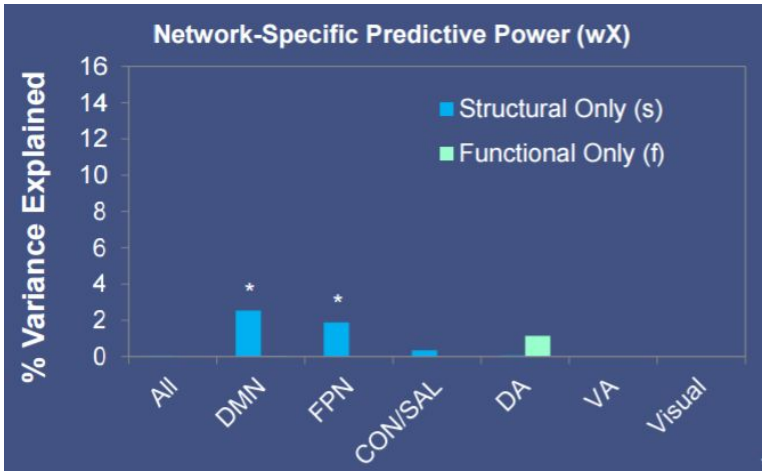


Figure 9: *Network-Specific Predictive Power*. The highest predictive power comes from the Default Mode Network which we hypothesized. * means that the p-value was less than 0.05.

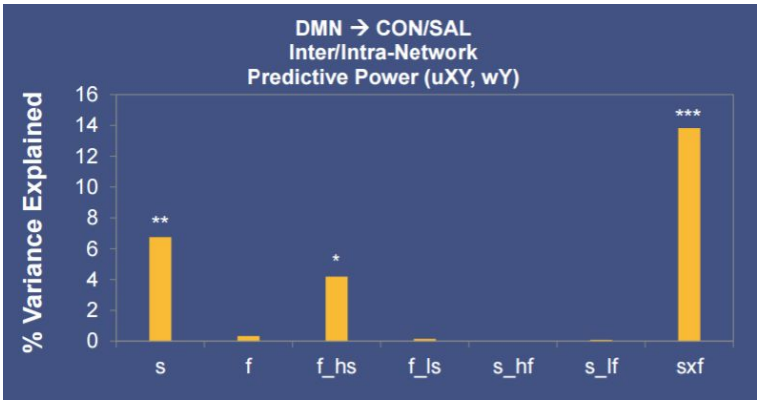


Figure 10: *Default Mode Network with Cingulo-Opercular/Salience Networks*. With the aoXY_wY scheme, achieved the highest R^2 value of around 14%. * means that the p-value was less than 0.05, ** 0.005, *** 0.00005.

When SC and FC were considered independently, only SC within the DMN and Fronto-Parietal Network (FPN) was predictive of fluid intelligence.

Given our hypotheses about the importance of across-network communication between the DMN and CON (Salience network was combined due to literature that it has similar properties to CON), we examined whether including the connectivity between nodes of these networks would improve prediction

of fluid intelligence. Interestingly, when FC data were limited to node pairs with high SC values, FC became predictive of fluid intelligence. Even more predictive was the interaction between SC and FC.

Conclusion

These results demonstrate that individual differences in the connectivity within and between brain networks can explain variance in fluid intelligence abilities (particularly with the PMAT). When examining intra-network effects, we found that structural connectivity was a better predictor of fluid intelligence than functional connectivity, with reliable effects seen in the DMN and FPN. When examining inter-network effects, we found that structural connectivity between nodes of the DMN and CON/SAL networks was particularly informative. Functional connectivity was also informative, but only when constrained by high structural connectivity values. Most interestingly, the interaction feature set of both functional and structural values provided the highest predictive power for our model. Using both types of connectivity was notably more informative than just using either structural or functional connectivity alone.

The CON/SAL has been shown to be a domain-general, task-control network, while the DMN has shown to be the opposite, whose decreased activity associates with task control. Perhaps, what our model has extracted is the efficiency of the CON/SAL's ability to inhibit the DMN, so that a subject is not distracted by any internally generated narrative that could distract him/her from engaging in a complex cognitive task.

Also, given a longer time period to work on this project, we would like to run the same analyses with the LASSO regression model, as that might prove to be even better than our results that we found here.

Finally, other studies have shown that brain connectivity (both structural and functional) actually change over time, possibly due to the dynamic process of myelination and cell birth/death. Abnormal changes in connectivity have also been shown to lead to diseases like Alzheimer's and dementia. Brain connectivity is a very complex subject and these results only scratch the surface of what can be discovered by it.