# Neural Correspondence Mapping

Advisor: Bradley Voytek https://voyteklab.com/
Team Members: Adita Zeqollari, Arlens Zeqollari, Erik Hoye, Robert Reeves
MAS DSE Cohort 5, 2020 - Group 4
Github: https://github.com/voytek/NCM

## Abstract

The analysis and comparison of modern neuroscience data is prohibitively diverse, leading to challenges in aggregation, verification, and collaboration as a whole in the neuroscience research community. These challenges make hypothesis generation difficult since findings from one study are not easily ingested by other studies with different processing techniques, sampling, and brain atlases. Within this paper, we propose a method to aid neuroscience research discovery by facilitating semi-automatic hypothesis generation with the creation of a flexible and extendable open-source Python library that merges disparate neuroscience data into a common coordinate system. The package interfaces with the commonly used MNI-XYZ and MRI-voxel coordinate system and allows for mapping to any broader parcel scheme defined by an atlas provided in a Nifti file. Data points are processed and mapped to other coordinate systems using a custom weighting algorithm. The package is also designed to perform single or dual hemisphere analysis. The package can be utilized locally or deployed in a cloud computing service. We demonstrate herein usage of the package in AWS leveraging S3, AWS Sagemaker, and Papermill. To demonstrate the value proposition of the package, the Allen Brain Atlas dataset of gene expressions was processed and modeled using xgBoost with Bayesian hyperparameter optimization in AWS Sagemaker. The model results were visualized using nilearn and pysurfer to generate visualizations representing potential physical or functional gene expression associations between parcels. Ultimately, this package can serve as a common platform for analyzing diverse neuroscience data to support neuroscience data research and collaboration.

## Introduction and Question Formulation

Modern neuroscientists have been exploring the brain for decades performing countless studies and testing numerous hypotheses. George H. W. Bush even claimed that the 1990s was to be the "decade of the brain". This would lead one to believe that our understanding of the brain and the study of neuroscience itself is growing at an unprecedented rate. Unfortunately, this is not the case. Despite numerous publications, acquiring data utilized in these experiments to further investigate and learn more about the brain is highly difficult if not impossible. There is no single location where the data is stored and these unique datasets are often stashed away for private use or even lost over time. Even if these datasets can be acquired, comparing them is highly difficult due to their diversity and variances is sampling.

One big issue for comparison between various datasets is the coordinate systems utilized in each study.  Initial measurements are often taken on the brain in very precise methods such as MNI-XYZ or MRI-Voxel coordinates and then a broader parcel or sub-section measurement of the brain is derived from these more precise measurements. Unfortunately, instead of neuroscientists utilizing a common atlas, parcel scheme for dividing the brain into subsections, studies often involve the generation of their own atlas in order to best fit their data or utilize one of the numerous pre-existing ones.  This makes it incredibly difficult to compare different studies.

In order to allow for comparison between these heterogeneous datasets generated by countless studies, we aim to provide a platform that not only allows them to be compared but also preserves the accuracy of the initial measurements and studies. This is where the problem becomes more of a data science issue than a neuroscience one. We propose that we can aid neuroscience research discovery by facilitating semi-automatic hypothesis generation with the creation of a flexible and extendable open-source python library that merges disparate neuroscience data into a common coordinate system.

Although this approach seems like a necessity in the field, studies have only taken this approach for either smaller subsections of the brain or in a very specific manner for their datasets.  UCSD's own PHD student Richard Gao's investigation into Neuronal timescales performed initial investigations into such techniques and provided us with a starting point to explore and expand upon.  Some Python packages exist as well that perform some capabilities necessary for this endeavor such as nilearn and nibabel, but they miss most of the functionality in order to perform the task. Our goal is to put together these packages and techniques done by pre-existing studies, expand upon them, and create a single pipeline that performs the mapping automatically so that neuroscientists can focus on understanding the brain instead of learning the intricacies of these various packages and methods.

## Team Roles and Responsibilities

The project team's roles and responsibilities followed the guidelines provided, while adjusting throughout the course of the project to meet project requirements and provide the entire team exposure and experience to the various areas of expertise required.

The team met at least once a week to discuss all aspects of the project and all team members met with the advisor regularly. The team's project manager, **Robert Reeves**, took the lead on coordinating meetings with the team members and project advisor, ensuring all project requirements were met and submitting the reports and presentations on time. Robert also ensured that the project advisor had access to all the deliverables of the project by providing the advisor with the submitted materials and soliciting his feedback throughout the course of our project. Robert was also instrumental in creating a collaborative environment for the team.

**Arlens Zeqollari** took the lead on  implementing the package architecture while **Erik Hoye** took the lead on developing the Python code for the package with all team members contributing with preprocessing work, planning, and ensuring the package met all the requirements.  **Erik Hoye** also spent time investigating and implementing different methods for visualizing spatial data on the brain as both parcels and coordinate points.

The modeling aspect of the project was shared by **Arlens Zeqollari** (xgBoost) and **Adita Zeqollari** (Adaboost). The team experimented, discussed and considered various models and inputs to the model before arriving at the optimal solution. The best performing model (xgBoost) was then migrated to AWS Sagemaker to perform modeling on additional parcels. Migration to AWS involved refactoring to the Sagemaker xgBoost API (**Arlens Zeqollari**) and integration with S3 (**Erik Hoye**).

The team's efforts in AWS were facilitated by **Adita Zeqollari**'s and **Robert Reeves**' work on creating S3, Sagemakes, and EC instances in AWS. The team also kept track of the budget and was successfully able to complete the requirements of the project within the AWS budget provided by the DSE program.

## Brief Description of Data Pipeline

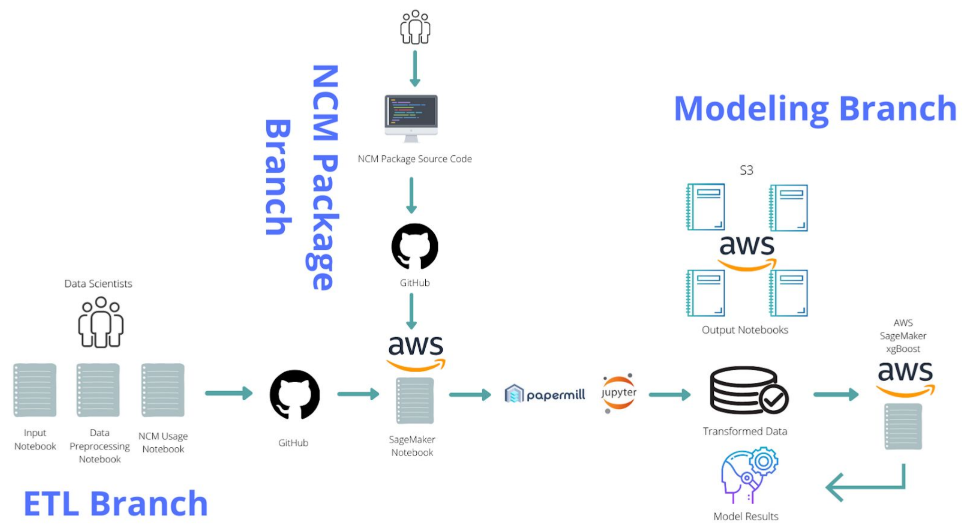


**Figure 1: Solution Pipeline**

Figure 1 above, shows the overall data pipeline for testing and demonstrating the entire project's capabilities. The pipeline takes us through the initial preprocessing of our desired datasets beginning at the far left, the transformations provided by the package visualized within the center, and then the final model and testing implementations visualized on the far right. The pipeline is intended to demonstrate our use of the classic ETL pipeline and that our process can be implemented both locally and in AWS for faster preprocessing and modeling on larger datasets. In the following sections we will breakdown each leg of our pipeline explaining the process and how it was implemented to satisfy the overall project design.

## Data Acquisition

For testing our overall pipeline and the package itself, we first gathered well known neuroscience datasets to be utilized to test the package. Since our overall goal is to provide generic methods for analyzing neuroscience data, instead of focusing on designing a specific section of our pipeline for pulling the datasets from the web, we merely provided links and steps to download them so that if one desired to utilize these datasets in their own investigations they would have no problem doing so. The three major datasets we utilized were:

- **Allen Brain Atlas (ABA)** - Gene expression data taken from the brains of 6 donors. Each individual's dataset provides multiple readings for over 21,000 genes, the significance over mean gene representation for each gene, and the probe's coordinate location in both mni-xyz and mri-voxel coordinates.
- **NeuroSynth (NS) Data** - Word correlation to specific regions of the brain (ex. Alzheimer). Data was initially gathered from NIH neuroscience articles utilizing natural language processing and then smoothed into single value correlations(tf-idf) for over 3,200 words for each article and coordinate location.
- **ECoG data** - Dataset generated by fooof 0.1.3, a Python library/model developed by the advisor's team to parameterize neural power spectra, characterizing both the aperiodic 'background' component, and periodic components as overlying peaks, reflecting putative oscillations.

For more statistics about the preceding datasets please reference Figure 2.

| Raw Data Sources | Source Location | Features | Feature Size | Coordinate System |
|---|---|---|---|---|
| NeuroSynth Data | https://github.com/neurosynth/neurosynth-data | Neuroscience Terms | 3200 terms for 1430 articles | MNI-XYZ |
| Allen Brain Atlas | http://human.brain-map.org/static/do | Gene Expressions | Over 21,000 genes with multiple | MRI Voxel and MNI-XYZ |

| | | | | |
|---|---|---|---|---|
| Data | wnload | | readings for each probe for 6 human brains | |
| ECoG data | https://github.com/rdgao/field-echos/blob/master/data/df_human.csv | Neural power spectra features | 1723 electrode locations for time series data | MNI-XYZ |

**Figure 2 - Datasets Table**

## Data Preprocessing

Our next step and the initial section in the data pipeline was the data pre-processing phase. Although the specific preprocessing methods taken for each dataset are unique, it was included in the pipeline in order to demonstrate the necessity of performing this task. Our package does not solve any inconsistency within datasets, but instead takes clean preprocessed data and transforms it to a common spatial frame to compare with other clean preprocessed data. Working with all these datasets at first was daunting due to the fact that as data scientists cleaning neuroscience data is not only trickery but requires deep research in order to make sure it is done correctly. Luckily for us, only one dataset required a substantial amount of cleaning.

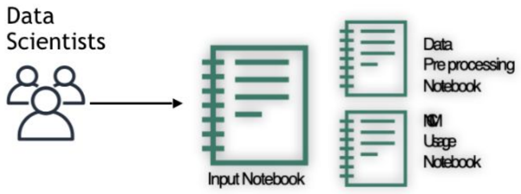


Figure 3 - Pipeline Data Preprocessing

**ECoG data** - We were able to acquire a clean preprocessed dataset from our advisor's PhD student, Richard Gao's open source Github page. In the ECoG dataset, each row represents one electrode reading, across all 110 patients.

To visualize the x, y, z coordinates for each row, a 3D plot was constructed.

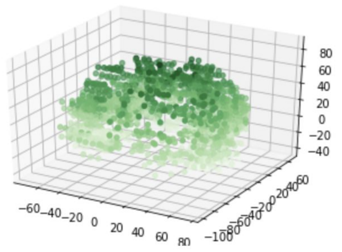


**Figure 4 - ECoG Data Visualization**

The only preprocessing necessary was to manipulate the dataset into the corresponding feature and coordinate numpy arrays to be utilized by the package.

**NeuroSynth data -** The latest NeuroSynth data file is stored in current_data.tar.gz. The current dataset is version 0.7, released July, 2018. The archive contains two files: database.txt and features.txt. The database.txt file contains activation data for 14,371 publications. The features.txt file contains feature information for 3,228 term-based features. Feature.txt contains term-based features derived only from the abstracts of articles in the Neurosynth database, and not from the full publication text. This data set is relatively clean as it has been processed to generate the two files we used for Input data.

| Data Set Name | Input data Sets | Processing Scripts | Final_Size*6 |
|---|---|---|---|
| NeuroSynth | • datatbase.csv<br>• feature.csv | • NSData_Pre-processing.ipynb<br>• Parcellation_Exploratio_NSData.ipynb | Final Dataframe 507,891 NMI locations × 3228 term values (tf-idf) |

**Figure 5 - NeuroSynth Preprocessing**

Below is an image of all points plotted and single terms points plotted using matplotlib.

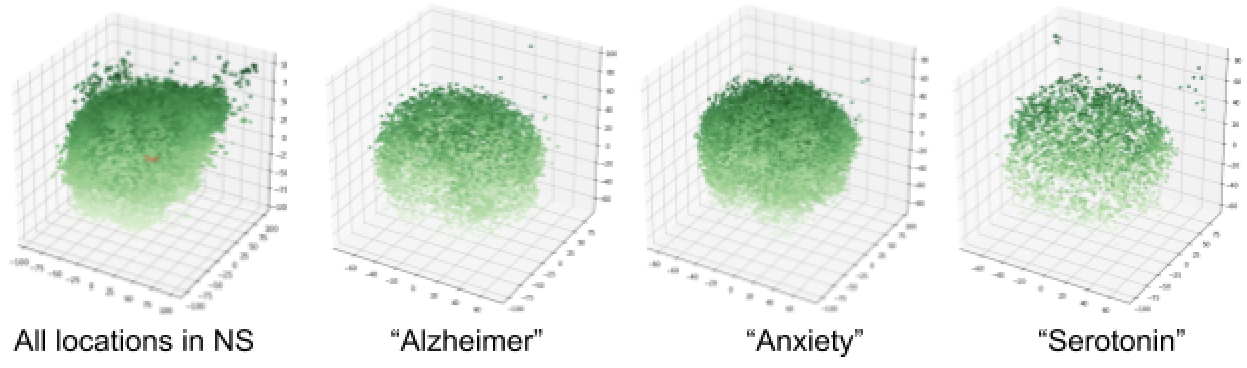


**Figure 6 - NeuroSynth Plots of Different Neuroscience Terms - Notice density difference**

One issue was that each publication could discuss multiple locations on the brain. Due to this the data set has 507,891 rows. Each row has corresponding 3228 term values. The data set created has over 1.6 billion cells. This large data frame is very cumbersome in python. In some cases it would error out due to lack of memory on both local and cloud environments. To deal with this the preprocessing was designed to work on only one term at a time. As a result the annotation file remains the same at 507,891 rows of x, y, z coordinates but the features data is reduced to a more manageable 507891 rows x single term column. This also seems to make sense as most research is likely done on a few terms. Also the preprocessing notebook was parameterized to take as an input a term variable. Then using python package Papermill the preprocessing notebook can be executed from a script for as many terms as a researcher requires. Below is an example of a papermill command for the term ‘serotonin’:

```
papermill NSData_Pre-processing.ipynb NSData_Pre-processing_output.ipynb -k
papermill-tutorial -p selected_feature 'serotonin'
```

The result of preprocessing is NS_Coordinates.csv and NS_Term_<my_term>.csv. The files are then used by a Notebook Example_NCM_package_NS.ipynb using the NCM package to map the data to a parcelization schema and plotted for analysis by the researcher.

**Allen Brain Atlas data** - The ABA data was the one dataset that required a substantial amount of cleaning and discussions with our advisors. The ABA dataset includes gene expression data taken from six brain donors for ~50,000 probes ranging from 900 to 400 mni-xyz locations on the brain. The data for each brain donor is located in five separate tables that require merging and dealing with several inconsistencies. These issues include: brain donor data containing only left side brain probes while others contain data from both sides of the brain, differences in probe location per brain donor, and significant differences in number of probes per gene. The following preprocessing steps were taken from *Hierarchy of transcriptomic specialization across human cortex captured by structural neuroimaging topography*. Through discussions with our advisors we decided to only implement 4 out of the 10 steps provided by the literature. Many seemed redundant and provided very little cleaning on the dataset.

| Data Set Name | Input data Sets * 6 donors | Processing Scripts | Final_Size*6 |
|---|---|---|---|
| ABA Data | ● MicroarrayExpression.csv<br>● Ontology.csv<br>● PACall.csv<br>● Probes.csv<br>● SampleAnnot.csv | ● ABA data pre-processing.ipynb<br>● ABA_data_pull.ipynb<br>● Parcellation_Exploration.ipynb | Final Dataframe 20787 genes × 449-900 probe locations |

1. Gene probes without a valid Entrez Gene ID were excluded. This was done due to genes without Entrez Gene ID not having genomes that have been completely sequenced, and they do not have an active research community to contribute gene-specific information.

2. Samples whose coordinates did not originate in the left hemisphere of the brain were then removed. This was done because research indicated there was not a significant difference in gene expression between the left and right side of the brain and because the last 4 donors did not have any probes in the right side of their brain.

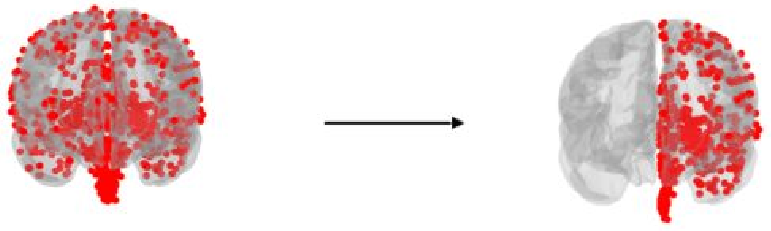


**Figure 7 - Removing probe samples from the right hemisphere**

3. Samples whose measured expression level was not well above background, as provided in the ABA dataset, were excluded. Samples remained if they (i) belonged to a probe whose mean signal was significantly different from the corresponding background, and (ii) had a background-subtracted signal which was at minimum 2.6 times greater than the standard deviation of the background.

4. Finally we performed one last transformation in order to reduce measurements to include only one measurement for each gene.

A. For those genes that contained more than one measurement in the same location which passed the preprocessing of phase 3, we took their mean expression
B. For those which included only one measurement in the same location which passed the preprocessing of phase 3 and others which didn't, we took the measurement which passed
C. Finally, for those which contained no measurement in the same location which passed the preprocessing of phase three, we labeled their gene expression as 0.0

After these four steps were applied to the six individual datasets for each brain donor, two final pandas dataframe were created, one containing gene feature data and one containing coordinate values in both MNI-XYZ and MRI-voxel.

## Package Implementation and Mapping

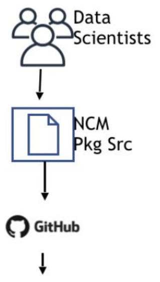


Figure 8 - NCM Package

Now that we have successfully explored the first phase of our data pipeline, let's take a closer look at phase two: package implementation and spatial mapping. Once again, the goal of the package is to take sparse coordinate data provided in either MNI-XYZ or MRI-voxel and map it to a broader more general parcel scheme so that disparate datasets can be compared to each other. An example can be visualized in Figure 9 below. The package itself converts around 700 lines of code and various functions into four simple lines of code that can easily be implemented by any neuroscientists as long as they have their data in the correct format. The package follows a class based architecture which means it first needs to be initialized and then various functions can be run on the stored data. This style of modularization allows for easy package addition and modifications in the future without breaking the entire package. Our goal was to make this package extremely easy to use, but also allow for it to be easily modified if desired in the future for advanced mapping and smoothing techniques. Below we will take a closer look at the data shape and format the package requires for use, how to utilize each specific function, the possible additions and modifications for specific functions, and the possible extensions that could be added to the package for improvement.

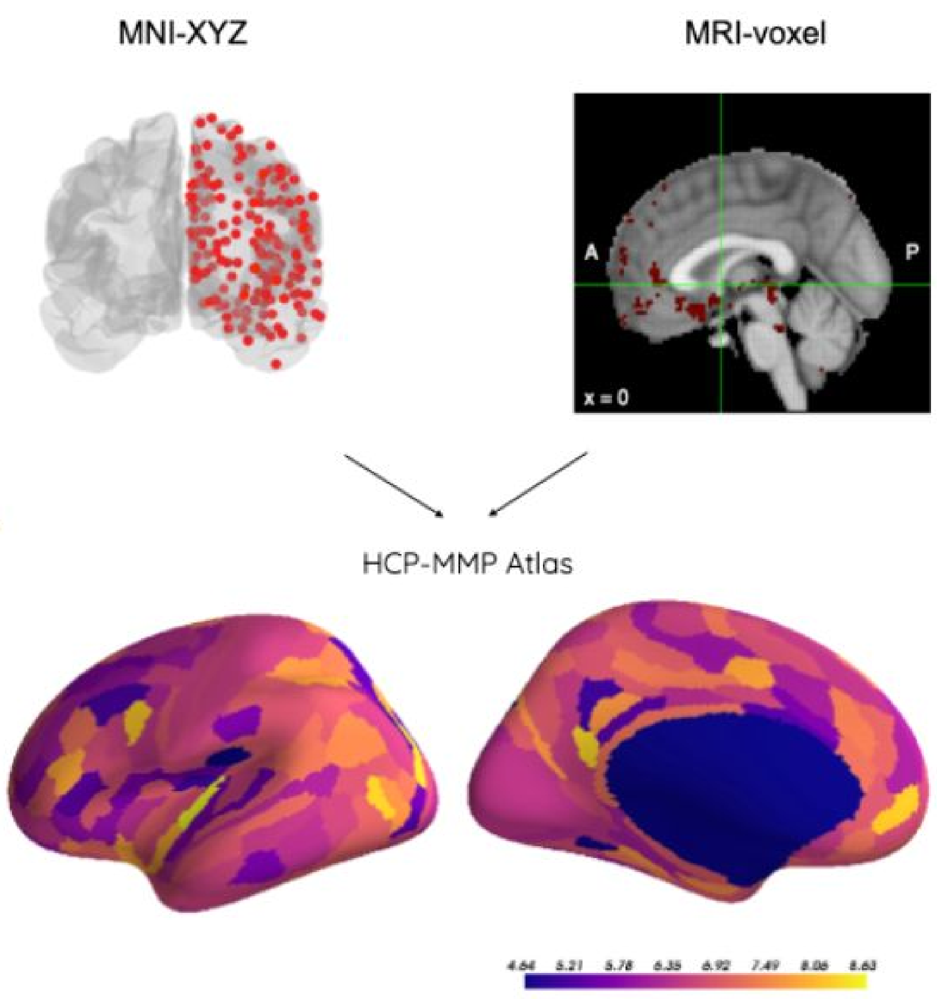


**Figure 9 - Mapping from Precise to Broad**

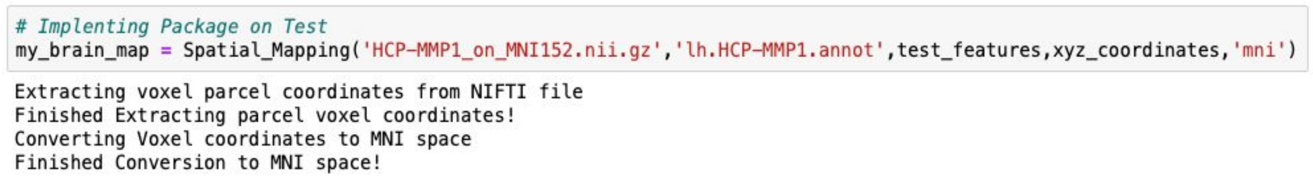
**Step 1 Package Initialization**

In order to utilize the package, you must first provide information about both the atlas you wish to map your data to, and the data itself. The first two inputs for initialization require the file locations for both a nifti file and an annotation file. A nifti file format dictates a file followed by either .nii or .nii.gz and it is a very common format utilized by neuroscientists to store images that provide parcel identifications within voxel space.  It also provides details on translating voxel coordinates to MNI space.  One may also provide an annotation file which is utilized to provide parcel identifiers, but the package will still work if instead of providing the file you specify "None".  Both these files must be stored in the specified nifti file folder within the NCM package location to work properly.

The next three input files required are your feature data, which is the experiment or study data that you want to map, the coordinate data file, which provides the coordinates or locations where your measurements were taken, and the source coordinate type in either MNI-XYZ or MRI-voxel.  The package requires that one utilizes these two coordinate measurements because they are the most accurate, and preserving accuracy is one of the main goals of the package. Implementing transformations from a broad parcel definition to another broad parcel definition is not recommended because you would likely need to identify parcel values by averaging shared surface areas with the other parcel definitions which would have already been averaged themselves.  Taking the average of averages is a good way to lose accuracy.

Lastly you must provide the axis in which the feature data and coordinate data correlate. For example if your feature data is stored in numpy array formatted (features, location_identifiers) and your coordinate date is formatted (location_identifiers, coordinates)  then you would provide axis = 0.  On the other hand, If your feature data was transposed, (location_identifiers, features), then you would provide axis = 1.  Axis is defined as 0 unless otherwise specified.

An example of the initialization function is provided below where NCM is the package, and spatial mapping is the class utilized for mapping sparse coordinate data to parcels.
Initialization Function:



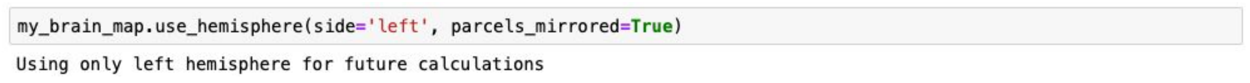
```
# Implenting Package on Test
my_brain_map = Spatial_Mapping('HCP-MMP1_on_MNI152.nii.gz','lh.HCP-MMP1.annot',test_features,xyz_coordinates,'mni')

Extracting voxel parcel coordinates from NIFTI file
Finished Extracting parcel voxel coordinates!
Converting Voxel coordinates to MNI space
Finished Conversion to MNI space!
```

**Figure 10 - Initialization Function**

Once the package has been initialized, it first extracts the image from the nifti file and identifies all parcels within the image. Then it determines whether or not converting parcel voxel coordinates to mni space for mapping is necessary by checking the coordinate type input, and then finally it checks to make sure data is in correct shape for transformations.

## Step 2 Hemisphere and mirror input

The next step in mapping the data from precise coordinate data to broader parcel data is to identify what type of analysis you want to perform. The package allows the user to identify whether they want the analysis to be dual hemisphere, or single hemisphere defined by left or right. Allowing the user to choose is especially significant for datasets such as the ABA dataset because all probes utilized in the last 4 brain donors are only on the left side of the brain. Analyzing both sides of the brain would likely skew results allowing for misinterpretation of correlations between datasets. Below is an example of using left-hemisphere:



```
my_brain_map.use_hemisphere(side='left', parcels_mirrored=True)
```

```
Using only left hemisphere for future calculations
```

**Figure 11 - Use Hemisphere Function**

Although choosing which side to perform your analysis on is important, even more significant is identifying whether or not the atlas provided is mirrored or not. Although not a common neuroscience term, we utilize this term to identify whether every parcel defined by the atlas has been "mirrored" from one side to the other. We identified this difference while observing parcel structures of the HCP-MMP atlas and the Talairach atlas, and it is extremely significant when performing mapping on either style atlas when utilizing parcel centers for calculations. Below in Figure 12 and 13 you can observe the difference between mirrored parcel schemes and non-mirrored.

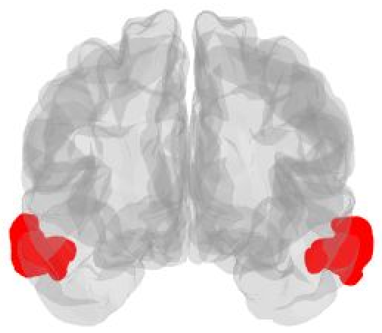


**Figure 12 - Mirrored Parcel Figure**

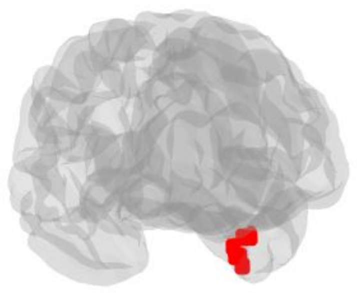
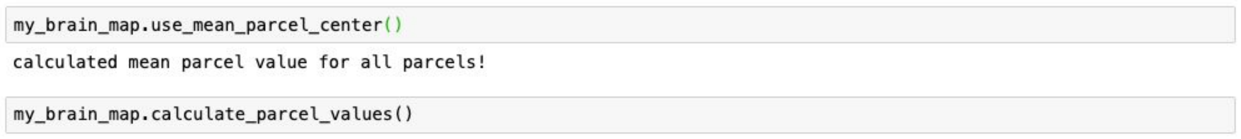


**Figure 13 - Un-mirrored Parcel Figure**

## Step 3 Mapping Method



```
my_brain_map.use_mean_parcel_center()
calculated mean parcel value for all parcels!

my_brain_map.calculate_parcel_values()
```

**Figure 14 - identifying mapping method**

After choosing the hemispheres we plan on using, our next step is to decide how we want to calculate parcel values. In order to calculate the feature expression in each parcel there are several different methods we can take. One method would entail mapping each measurement point for your inputted feature data to the closest individual parcel and then calculating the average expression in each parcel if multiple measurements have been mapped to it. One drawback for this method is that some parcels might not have any measurements mapped to them. This can be remedied by calculating the surface area shared by these parcels lacking any measurements and those of surrounding parcels, and then taking a weighting of their gene expression based on the shared surface area. The drawbacks for this method is over-generalization for parcels lacking data and that it would take an extremely long time to perform. We initially explored this method and realized it would take too long if one wanted to run the analysis locally.

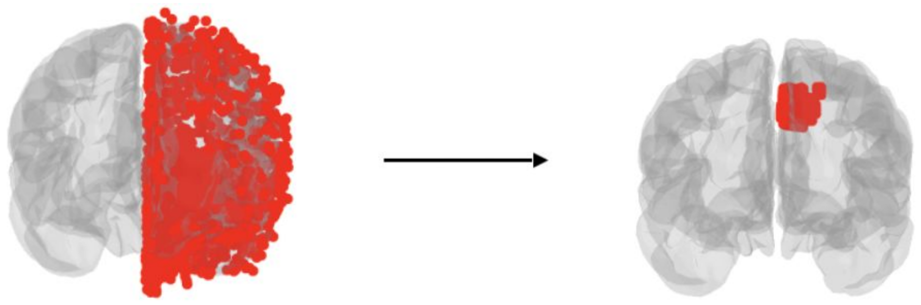


**Figure 15 - Mapping Points to Closest Parcel**

Another option is to instead calculate the centroid of each parcel, and then calculate the likely parcel expression by weighting each sample feature data based on the distance to the center of the parcel, or perform gaussian smoothing. This also has a downside regarding the correct way to determine the weighted distribution for each parcel and the awkward shape of certain parcels which can lead to some generalization. The upside however is that it is quite a bit faster.

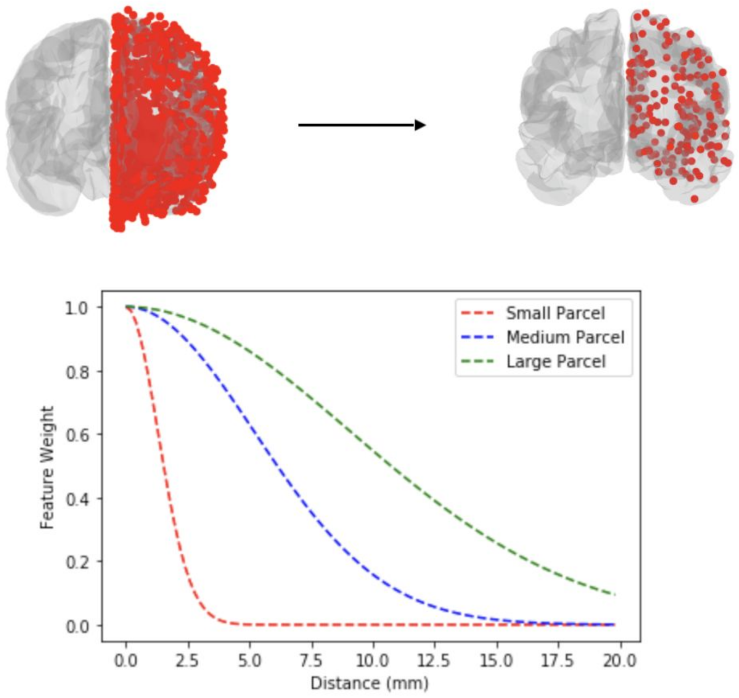


**Figure 16 - Calculating Parcel Value by Weighting Samples based on Distance**

The numerous possibilities for defining the generalized parcels values was one of the major reasons why we wanted to design an open-modularized package. This allows any future developers to implement and add to the package any different techniques they plan on implementing.  We ended up implementing method two due to our desire for the package to be capable of running either locally or in AWS at an acceptable speed.  We tested the package on all three of our datasets and here are some examples mapping from precise measurements to broad parcel definition in the HCP-MMP atlas utilizing the package pysurfer and nilearn for visualization.

## Visualizing Package Transformation

Observing the visualizations below we can immediately identify some significant variances between our three datasets.  Gene expression for gene 729 from the ABA dataset in Figure 17 contains a small spread ranging from 4.6 to 8.4, while the knee_freq extracted from the ECoG dataset in Figure 18 ranges from 2.6 all the way to 36 a massive difference between parcels.  On the other hand, Figure 19 shows data from the neurosynth dataset visualizing that most parcels have no correlation to the investigated terms and only just a few parcels are significant.

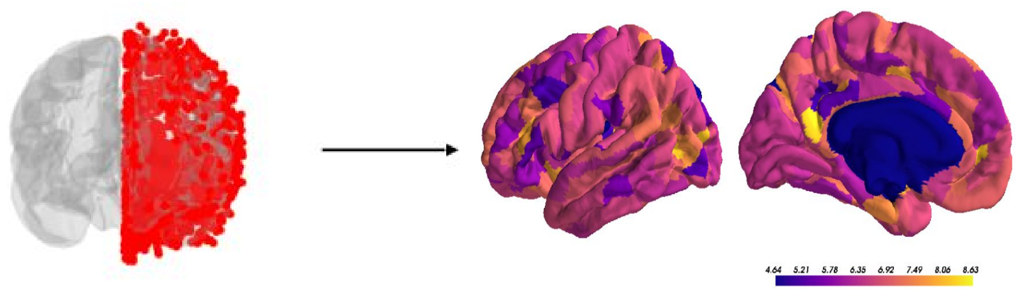


Figure 17 - Gene 729 Expression for Left-Hemisphere of HCP-MMP Atlas

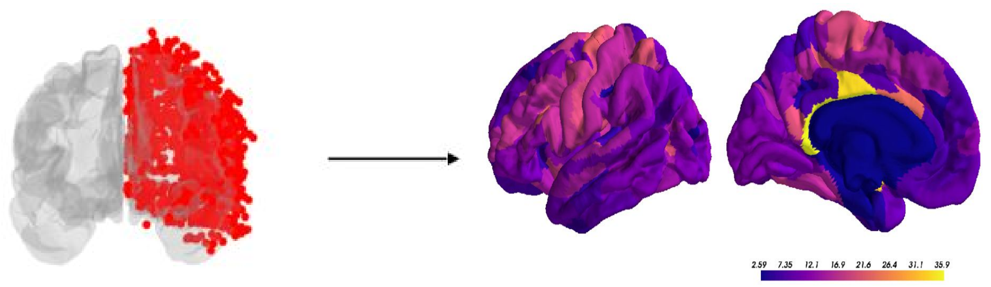


Figure 18 - Knee_freq from ECoG data for Left-Hemisphere of HCP-MMP Atlas

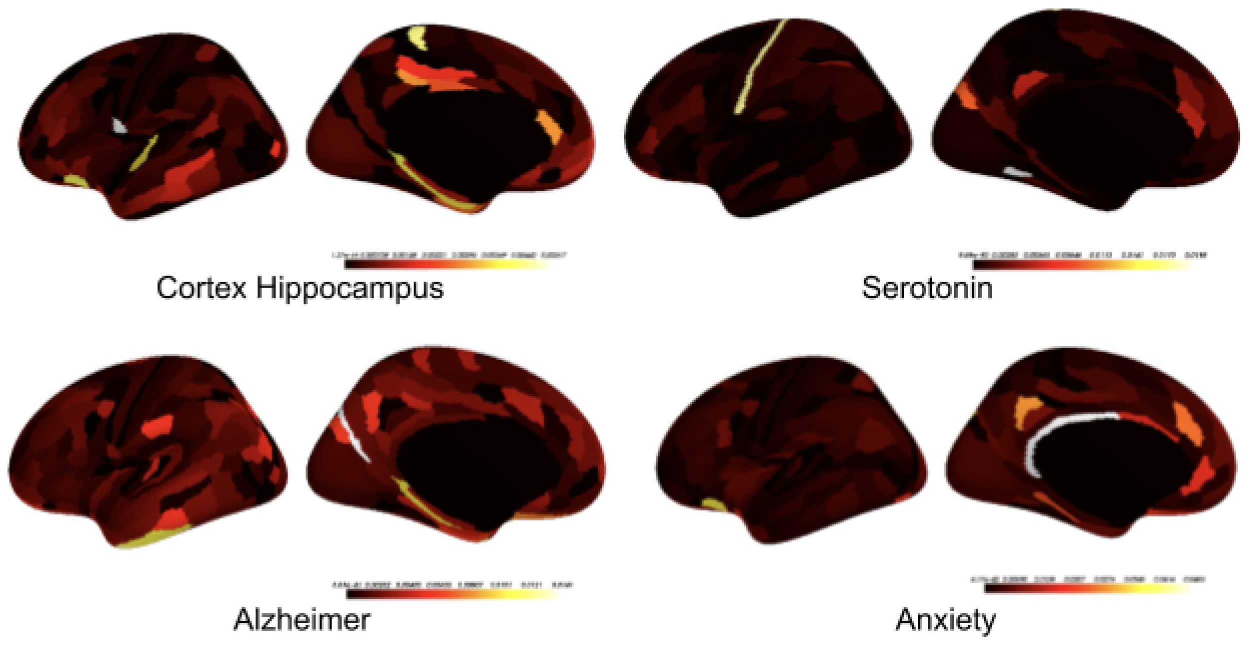


Figure 19 - NeuroSynth Terms for Left-Hemisphere using HCP-MMP Atlas

## Further Investigations into Side Cases

After successfully building the initial package, the team spent a substantial amount of time attempting to deal with the numerous side cases that the package would face. In no way have we identified all of them, but we took our best shot in identifying as many as we could with the various datasets and atlases we investigated. This is another reason why creating an open-source package allowing for updates on specific use cases was significant to our project. Below are specific cases we investigated and how we dealt with each.

Dual Hemisphere analysis on Mirrored Atlas - Performing a mapping by Identifying parcel center when the parcel is defined on both the left and right side leads to calculation errors. Meaning the parcel center is calculated to be at the middle of the brain regarding the x-axis. In order to deal with this, when performing a dual analysis on a mirrored atlas, parcels are divided in half and labelled according to their parcel identity and hemisphere. Then calculations begin regarding parcel center and feature representation for parcel.

Ex: parcel V1_ROI from HCP-MMP atlas split into two parts labelled left_V1_ROI and right_V1_ROI indicating the specific hemispheres they are located on.

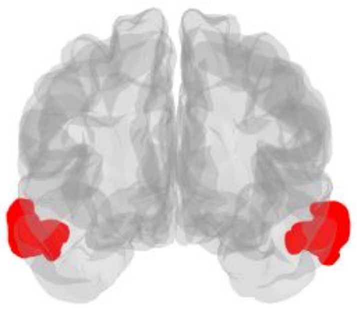


**Figure 20 - Single parcel labelled above converted to two unique parcels**

Single Hemisphere analysis on Non-Mirrored Atlas - Performing single side mapping for a mirrored atlas is easy because you simply remove each parcel definition coordinates that are on the opposite hemisphere. For un-mirrored atlases we can't take such a simple approach since any parcel defined along the center of the brain is unique and cannot simply be divided in two. In order to deal with this, when performing any single side analysis on a non-mirrored atlas we remove parcels which contain no values on the interested side but maintain the entire parcel definition if some portion is on the hemisphere we are interested in. Most of the parcels appear to remain on their respective sides, but in some cases they do not.

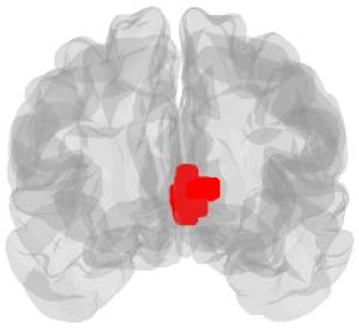


**Figure 21 - Entire parcel kept for centroid calculations despite partially being in wrong hemisphere**

**Smoothing calculations** - One of the most significant investigations of this project involved variations in the smoothing calculations. After calculating the parcel center and identifying the distance from each point to the parcel center, one must then determine how to weigh the feature data based on that distance. This was challenging as there is not necessarily any "right" way to approach the problem. The package currently applies a distance-based gaussian weighting function which utilizes the size of each parcel. This means that the gaussian function utilized is modified for each parcel for parcel feature representation calculations. Although this was effective for ABA MNI-XYZ data mapping to the HCP-MMP atlas, finding one method that will work for all data or atlases is unlikely. For sparse data which requires more generalization, or robust data, which requires more precision in parcel definitions, it is likely that the user would prefer to modify this function or even add their own. We allow for modifications allowing for constant gaussian smoothing regardless of parcel size, but attempting to create a robust function was not feasible. Once again, the class-based package developed by the team should allow for users to apply modifications or create their own functions they wish to apply to their own data. Figure 22 illustrates the possible weighting that could be applied to features based on various distances to parcel centers.

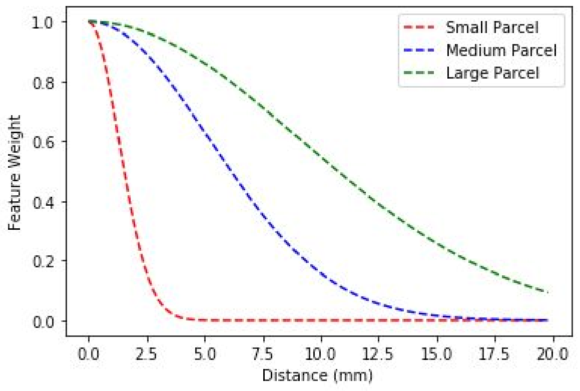


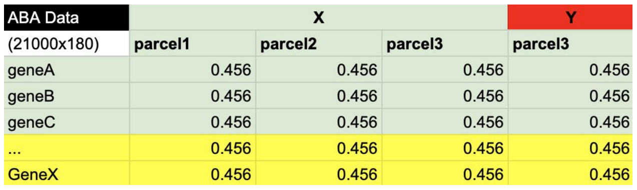
**Figure 22 - Gaussian based weighted distance function**

Some of the more minor investigations and implementations added to the package involved allowing for a lack of annotation files, variances between single hemisphere analysis regarding MRI-voxel vs MNI-XYZ, exclusion of parcels in output when they were not in the correct hemisphere, and replacement of NaN values to 0.0 for parcels where all precise measurements are too far away for proper generalization.

## Modeling

Now that we have fully walked through package implementation, we can examine how modeling can be applied to the results.  Although our project and data sources did not lend themselves well to modeling and extensive modeling efforts did not align with our project's overall goal to build a package for merging disparate neuroscience data into a common coordinate system, we saw modeling as a method for utilizing the results and capabilities that the package provides.   After discussions with our advisor, the team was able to formulate interesting modeling questions and identify the ABA dataset as input to answer the question: "Given parcel gene expressions for all other parcels, can we predict all gene expressions for a specific parcel?"

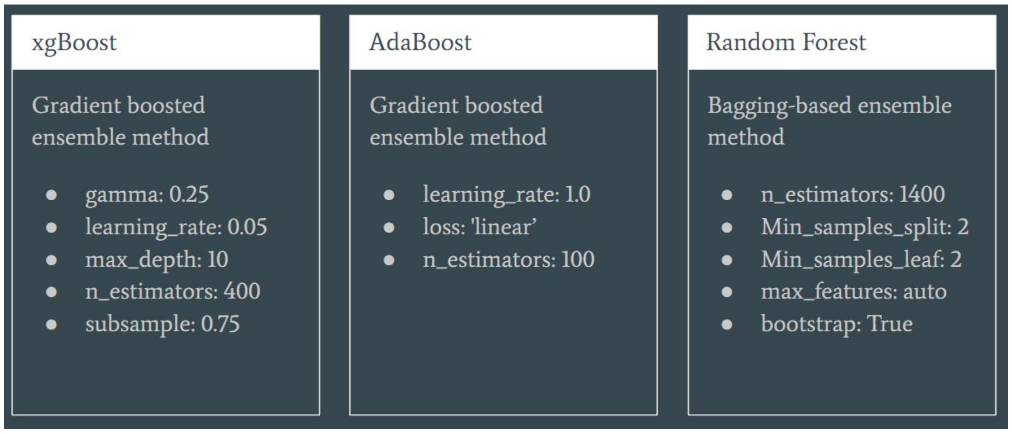
Figure 23 uses parcel gene expressions from the ABA data as features and the genes as samples. The values are the weighted averages from the six brains in the dataset.



| ABA Data | X | | | Y |
|---|---|---|---|---|
| (21000x180) | parcel1 | parcel2 | parcel3 | parcel3 |
| geneA | 0.456 | 0.456 | 0.456 | 0.456 |
| geneB | 0.456 | 0.456 | 0.456 | 0.456 |
| geneC | 0.456 | 0.456 | 0.456 | 0.456 |
| ... | 0.456 | 0.456 | 0.456 | 0.456 |
| GeneX | 0.456 | 0.456 | 0.456 | 0.456 |

**Figure 23 - ABA Dataset for Modeling**

The team explored the ensemble methods outlined below.



xgBoost

Gradient boosted ensemble method

- gamma: 0.25
- learning_rate: 0.05
- max_depth: 10
- n_estimators: 400
- subsample: 0.75

AdaBoost

Gradient boosted ensemble method

- learning_rate: 1.0
- loss: 'linear'
- n_estimators: 100

Random Forest

Bagging-based ensemble method

- n_estimators: 1400
- Min_samples_split: 2
- Min_samples_leaf: 2
- max_features: auto
- bootstrap: True

**Figure 24 - Models evaluated for Gene Expression prediction**

For the **Random Forest** model, we achieved an MSE of .019 with a 80/20 train test split.
For the **Adaboost** model, we achieved an MSE of .07 and RMSE of .27.
For the **xgBoost** model, we achieved an MSE of .02.

The xgBoost model performed the best and was the fastest to train, as the library is able to make use of parallelization locally and the usage of GPUs locally as well. This allowed for better optimization of hyperparameters. We suspect that the regularization also helped achieve better performance and generalizability. Furthermore, AWS SageMaker support for xgBoost, makes the choice of models easy when considering local and cloud environments.

We suspect that the regularization also helped achieve better performance and generalizability.

| Model (Type) | Best Model Parameters | Cross-Validation | Train / Val / Test (%) | GridsearchCV Parameter Space |
|---|---|---|---|---|
| **xgBoost Regressor** (Gradient Boosted Ensemble Method) | **'gamma'**: 0.25<br>**'learning_rate'**: 0.05<br>**'max_depth'**: 10<br>**'n_estimators'**: 400<br>**'subsample'**: 0.75 | 3-fold CV | 56/24/20 | **'N_estimators'**:[50, 100, 400],<br>**'Max_depth'**:[3, 5, 10],<br>**'Learning_rate'**:[0.05, 0.1, 0.5],<br>**'Subsample'**:[0.5,.75, 1],<br>**'gamma'**: [0.25, 0.5, 1, 3], |

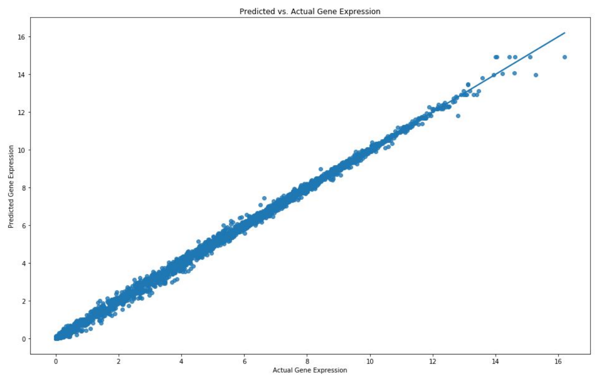
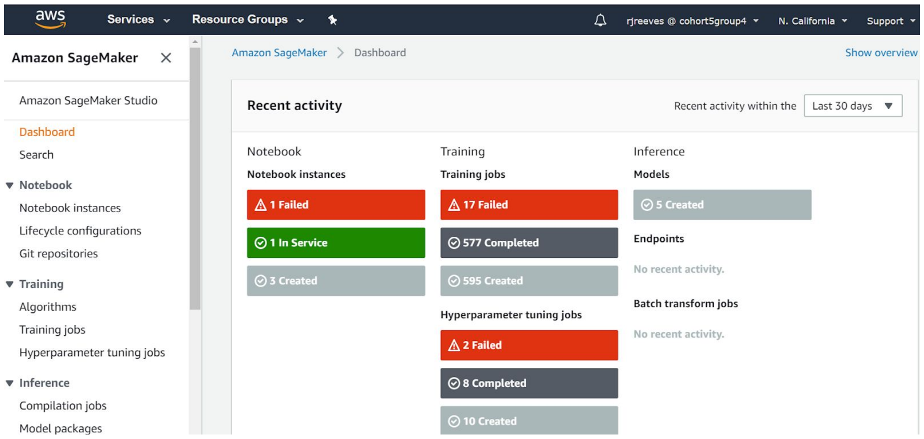


**Figure 25: Predicted vs Actual for xgBoost on ABA data**

## AWS Pipeline implementation

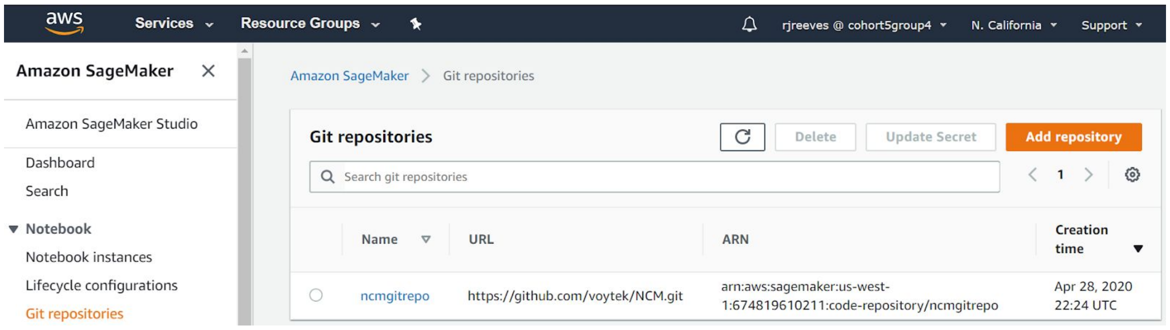
This project used the AWS Sagemaker Studio for cloud services. AWS Sagemaker Studio is an integrated machine learning environment where you can build, train, deploy, and analyze your models and Notebooks, all in the same AWS cloud application. Sagemaker provides experiment management and tracking. Users can organize their experiments and artifacts in a centralized location using a structured organization scheme. The Sagemaker environment provides full visibility into the model training process by enabling the inspection of all the training parameters and data throughout the training process, all with in-depth logging at every step. Sagemaker also creates a domain for the team. The domain includes an Amazon Elastic File System (Amazon EFS) volume with home directories for each user. Notebook files and data files are stored in these directories. Sagemaker also supports Git repository integration. The project intrated GitHub allowing for seamless pulling of project resources. Overall, the Sagemaker Studio is a good solution to enable ETL, Python package, and model development. The solution's automation and scalability are very straight forward with quick deployment, providing a complete automated pipeline with monitoring and debugging capabilities at any scale.



| Notebook | Training | Inference |
|---|---|---|
| **Notebook instances** | **Training jobs** | **Models** |
| 1 Failed | 17 Failed | 5 Created |
| 1 In Service | 577 Completed | **Endpoints** |
| 3 Created | 595 Created | No recent activity. |
| | **Hyperparameter tuning jobs** | **Batch transform jobs** |
| | 2 Failed | No recent activity. |
| | 8 Completed | |
| | 10 Created | |

**Figure 26 - AWS Sagemaker Dashboard**

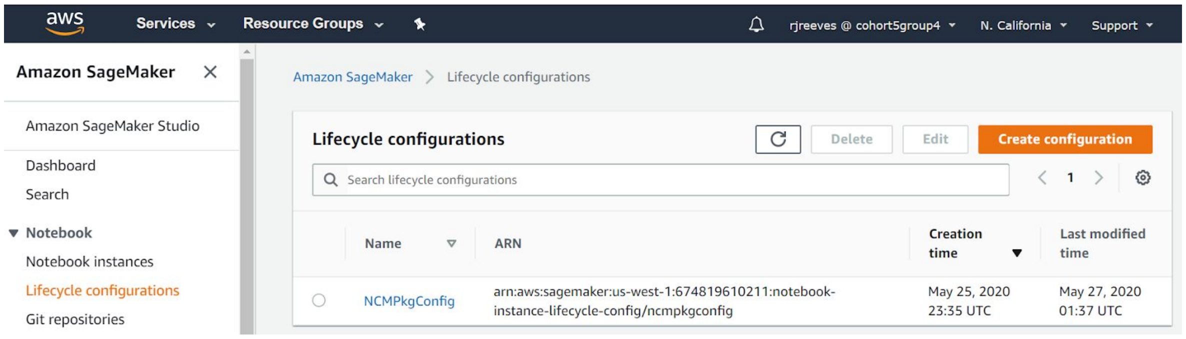
The Sagemaker Dashboard shown in the figure above illustrates some of the resources used for this project. The Notebook Instance, Training jobs and Hyperparamter tuning jobs.



| Name | URL | ARN | Creation time |
|---|---|---|---|
| ncmgitrepo | https://github.com/voytek/NCM.git | arn:aws:sagemaker:us-west-1:674819610211:code-repository/ncmgitrepo | Apr 28, 2020 22:24 UTC |

**Figure 27 - AWS Sagemaker - Git repositories**

Not shown in the Dashboard but also used by this project was integrated Git repositories shown in the figure 27 above. This integration shared a public key with GitHub and the private key kept in Sagemaker so no login required when pulling and committing code to and from Sagemaker notebooks. All three Notebooks used this integration to pull and push changes.



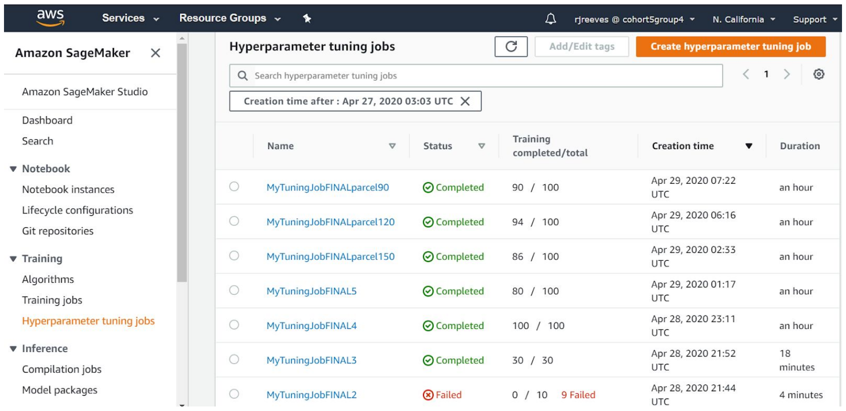
| | Name | ARN | Creation time | Last modified time |
|---|---|---|---|---|
| ○ | NCMPkgConfig | arn:aws:sagemaker:us-west-1:674819610211:notebook-instance-lifecycle-config/ncmpkgconfig | May 25, 2020 23:35 UTC | May 27, 2020 01:37 UTC |

**Figure 28 - AWS Sagemaker - Lifecycle configurations**

Another feature used but not shown in the Dashboard was the Notebook Lifecycle configuration seen above in figure 28. The Lifecycle configuration was used for both the test and final notebooks. The main function of Lifecycle configurations is to provide a shell script that runs when the notebook instance is started. The script we used configured the Notebook environment including the kernel, pulling from GitHub and running a notebook using Papermill. This allowed for automation in testing changes to the NCM package. The complete details of the script used are documented in the GitHub readme.md file.

## AWS Sagemaker Modeling

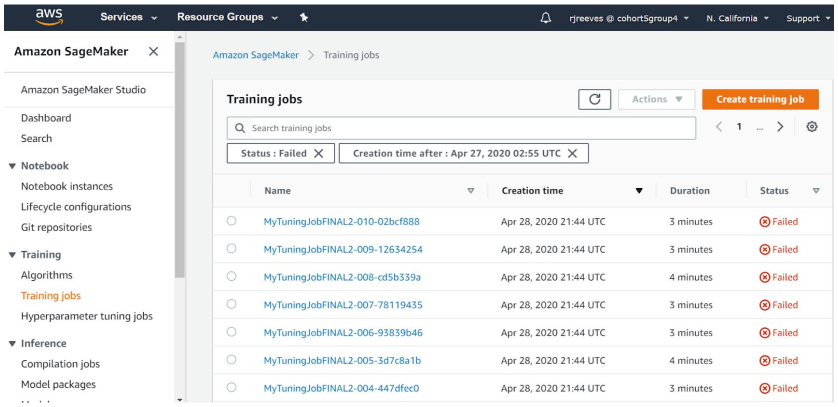
After initial investigations into modeling on local machines, we decided to investigate further into the capabilities of deploying models in AWS as well to test integration with larger datasets. The implementation of xgBoost algorithms using AWS's Sagemaker allowed for comprehensive machine learning training and hyperparameter optimization. The models were refactored into AWS's custom xgBoost library, which is designed to quickly build, train, and deploy machine learning models. The Sagemaker user interface provides helpful diagnostics and progress reports showing the results of each job with respect to the objective metric as the jobs are being trained (see figure 29 below).



| | Name | Status | Training completed/total | Creation time | Duration |
|---|---|---|---|---|---|
| ○ | MyTuningJobFINALparcel90 | Completed | 90 / 100 | Apr 29, 2020 07:22 UTC | an hour |
| ○ | MyTuningJobFINALparcel120 | Completed | 94 / 100 | Apr 29, 2020 06:16 UTC | an hour |
| ○ | MyTuningJobFINALparcel150 | Completed | 86 / 100 | Apr 29, 2020 02:33 UTC | an hour |
| ○ | MyTuningJobFINAL5 | Completed | 80 / 100 | Apr 29, 2020 01:17 UTC | an hour |
| ○ | MyTuningJobFINAL4 | Completed | 100 / 100 | Apr 28, 2020 23:11 UTC | an hour |
| ○ | MyTuningJobFINAL3 | Completed | 30 / 30 | Apr 28, 2020 21:52 UTC | 18 minutes |
| ○ | MyTuningJobFINAL2 | Failed | 0 / 10 9 Failed | Apr 28, 2020 21:44 UTC | 4 minutes |

**Figure 29 - Hyperparameter tuning jobs**

The training jobs feature of Sagemaker was also used.  Our failed training jobs can be seen in the image below.



Training jobs

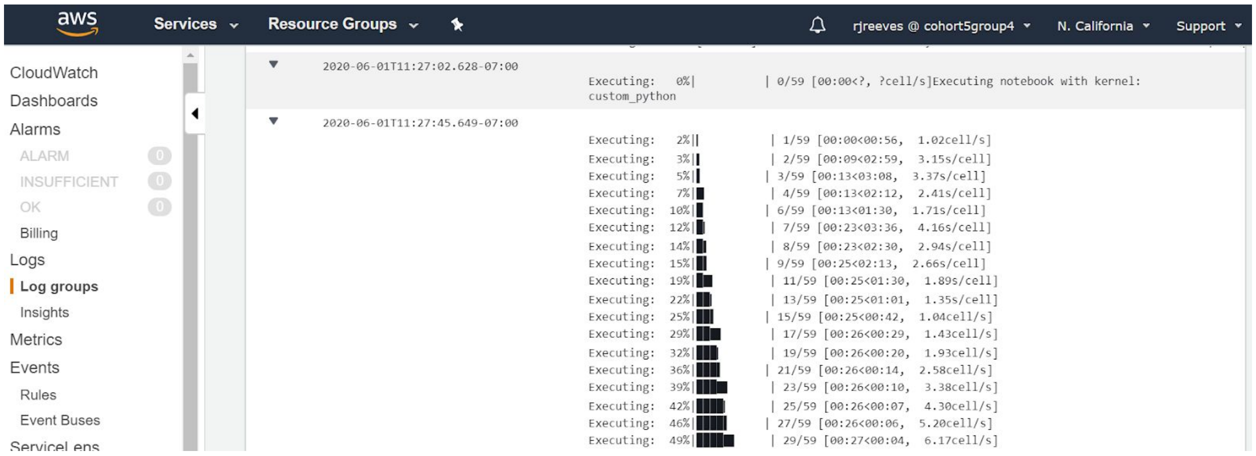
Status : Failed | Creation time after : Apr 27, 2020 02:55 UTC

| Name | Creation time | Duration | Status |
|---|---|---|---|
| MyTuningJobFINAL2-010-02bcf888 | Apr 28, 2020 21:44 UTC | 3 minutes | Failed |
| MyTuningJobFINAL2-009-12634254 | Apr 28, 2020 21:44 UTC | 3 minutes | Failed |
| MyTuningJobFINAL2-008-cd5b339a | Apr 28, 2020 21:44 UTC | 4 minutes | Failed |
| MyTuningJobFINAL2-007-78119435 | Apr 28, 2020 21:44 UTC | 3 minutes | Failed |
| MyTuningJobFINAL2-006-93839b46 | Apr 28, 2020 21:44 UTC | 3 minutes | Failed |
| MyTuningJobFINAL2-005-3d7c8a1b | Apr 28, 2020 21:44 UTC | 4 minutes | Failed |
| MyTuningJobFINAL2-004-447dfec0 | Apr 28, 2020 21:44 UTC | 3 minutes | Failed |

**Figure 30 - Training Jobs with AWS Sagemaker**

Sagemaker training jobs are an iterative process used to  teach our model to make predictions by presenting examples from our Gene expression training data.  The training algorithm writes metrics from jobs to logs, which Amazon Sagemaker monitors and sends to Amazon CloudWatch.  The logs allow the developer to pick the best results from their experiments.  Below in Figure 31 is an example of a CloudWatch log for notebook execution.

**AWS CloudWatch** allowed developers to debug execution of the Sagemaker notebook instances, lifecycle configurations, and model execution. These interactive outputs provide a comprehensive view of the environment, setup, and step-by-step execution of all AWS components involved in this project. This is essential for troubleshooting and optimizing scaled computing resources. An example output of the logs that are generated by executing Papermill in a lifecycle configuration at the startup of a notebook instance is shown below:



```
2020-06-01T11:27:02.628-07:00
        Executing:   0%|          | 0/59 [00:00<?, ?cell/s]Executing notebook with kernel:
        custom_python

2020-06-01T11:27:45.649-07:00
        Executing:   2%|▏         | 1/59 [00:00<00:56,  1.02cell/s]
        Executing:   3%|▎         | 2/59 [00:09<02:59,  3.15s/cell]
        Executing:   5%|▌         | 3/59 [00:13<03:08,  3.37s/cell]
        Executing:   7%|▋         | 4/59 [00:13<02:12,  2.41s/cell]
        Executing:  10%|█         | 6/59 [00:13<01:30,  1.71s/cell]
        Executing:  12%|█▏        | 7/59 [00:23<03:36,  4.16s/cell]
        Executing:  14%|█▎        | 8/59 [00:23<02:30,  2.94s/cell]
        Executing:  15%|█▌        | 9/59 [00:25<02:13,  2.66s/cell]
        Executing:  19%|█▊        | 11/59 [00:25<01:30,  1.89s/cell]
        Executing:  22%|██▏       | 13/59 [00:25<01:01,  1.35s/cell]
        Executing:  25%|██▌       | 15/59 [00:25<00:42,  1.04cell/s]
        Executing:  29%|██▉       | 17/59 [00:26<00:29,  1.43cell/s]
        Executing:  32%|███▏      | 19/59 [00:26<00:20,  1.93cell/s]
        Executing:  36%|███▌      | 21/59 [00:26<00:14,  2.58cell/s]
        Executing:  39%|███▉      | 23/59 [00:26<00:10,  3.38cell/s]
        Executing:  42%|████▏     | 25/59 [00:26<00:07,  4.30cell/s]
        Executing:  46%|████▌     | 27/59 [00:26<00:06,  5.20cell/s]
        Executing:  49%|████▉     | 29/59 [00:27<00:04,  6.17cell/s]
```

**Figure 31 - AWS CloudWatch Execution Log**

From Figure 31 above, one can view the cell-by-cell execution of the Jupyter notebook, which includes any errors, execution time, and progress. CloudWatch also allows queries and filtering to allow efficient methods enabling scalability.

## Findings & Reporting

Now that we have fully investigated the individual sections of our overall pipeline, let's take a closer look at our findings and tools utilized for reporting. Overall, our project demonstrates that it is possible to develop a platform that allows disparate neuroscience data to be compared by mapping data to the same spatial frame. The modularized NCM package allows for ingesting spatial data in both MNI-XYZ and MRI-voxel and has been tested to map data to both the HCP-MMP atlas and Talairach atlas. We discovered that mapping from MRI-voxel and MNI-XYZ space are the only mapping schemes where accuracy can be preserved through testing. Further inquiries into atlas intricacies also allowed our package to perform mapping on either the full brain or on just one hemisphere. Investigations also led to the discovery of how significant the feature weighting based on distance to parcel center was for parcel representation, especially with large variances in atlas parcel definitions and dataset sizes. In order to test package outputs, we were also able to develop several xgBoost models which helped demonstrate the potential physical or functional associations between different parcels, and can be valuable for hypothesis generation. For visualizations and reporting, we utilized both Nilearn and Pysurfer. Nilearn plotting allowed us to easily visualize individual datasets and parcel center calculations for verification of our results. Pysurfer provided us with a platform that allowed us to demonstrate the results of the mapping performed by the package. Finally, for additional reporting we utilized tools in AWS to automate the NCM package testing, notebook execution with scalability, and debug logging. Altogether, the package and work herein demonstrates that an extendable package can be developed to support neuroscience data analysis and hypothesis generation.

## Solution Architecture

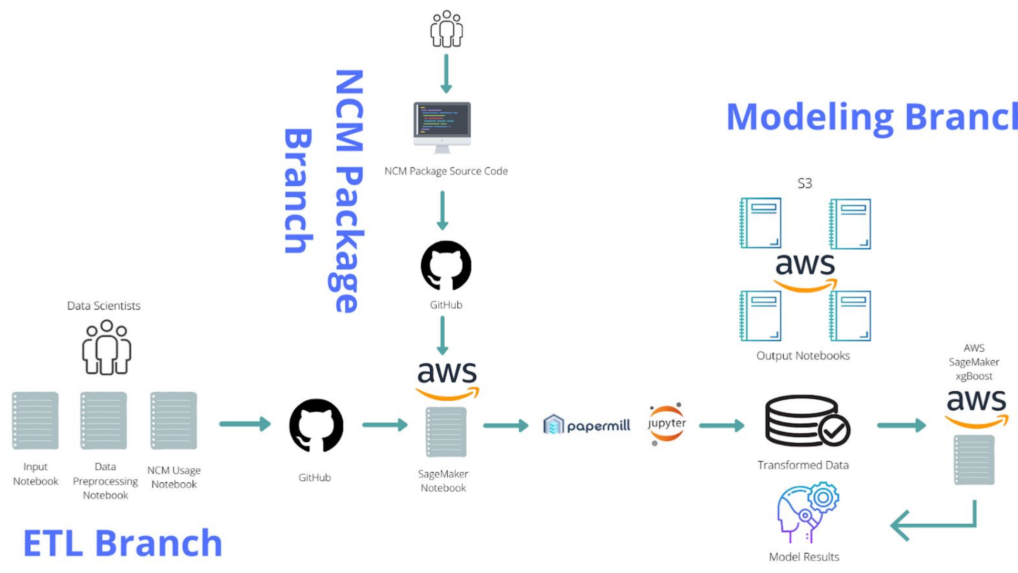


**Figure 32 - Solution Architecture**

As initially demonstrated in the beginning of this paper, the overall solution and pipeline for this project is shown again in figure 32. This structure takes us through initial preprocessing, package implementation for data mapping, and eventual result investigations for hypothesis generation on neuroscience data. The architecture works in both cloud and local environments. For the local environment the AWS pieces of the above solution are removed and S3 is replaced by local storage. The use of Jupyter notebooks allows for easy development and portability and the addition of papermill allows for automation and scalability. A great feature of papermill is when the notebooks are remotely executed an output notebook is stored as a log or record of the notebook execution. This can serve both as debug information as well as reports without manually running the notebook. The architecture is simple and powerful that allows for easy expandability in the future.

## Performance & Evaluation

The use of the operational models of cloud and local has proven to be a great model allowing speed, flexibility, simplicity, scalability, and maintainability of the development of the NCM project. If utilized within a scientific or business domain, one would only need to run the unit test notebook to make sure that the package is working correctly before proceeding with implementation on their own datasets. Since this project was not severely compute-intensive, the AWS budget was a negligible concern; the team spent approximately 25% of the allocated budget.

The models were accurate and demonstrate that gene expression values in other parcels of the brain can be used to accurately predict the gene expression in a specific parcel of interest. The team demonstrated that the technique generalizes gene expressions well to other parcels of the brain with similar accuracy.

The target of this package is not a prediction, but rather another dataset/visualization that is able to elucidate further analysis by neuroscientists, data scientists and researchers. These can also include aggregations/summaries of the datasets, akin to profiling of the dataset where highly expressed genes, number of parcels, density of data, etc. are summarized and can be exported.

## Conclusion

This project demonstrated that a package to transform neuroscience data could be developed, scaled, and lead to actionable hypothesis generation and exploratory data analysis for neuroscience data specialists. We hope that our findings can be expanded on and utilized in the neuroscience field in order to generate hypotheses about physical or functional interactions. While the package is open to all potential collaborators, the primary audience for this project involves the community of neuroscience data specialists and developers. These include researchers and software developers that are interested in developing scalable pipelines or methods for analyzing neuroscience data. The project demonstrates the transformation of several popular neuroscience data formats and parcellation schemes, which is a foundational technique in this field. The creation of the package repository, including the documentation, package setup, and ability to share with others via GitHub allows the audience to organically expand if interest is generated.  The team focused not only on the depth required to perform one specific task or functionality, but also the breadth in how such functionality can be integrated into scalable cloud solutions and extended with tools and techniques such as Papermill, CloudWatch, logging, cloud machine learning/training, parallelization, and deployment. In a modern implementation and usage of this package, the team theorizes that such functionality will be crucial to long-term sustainable development of the package.

## Appendices

### A. DSE MAS Knowledge Applied to the Project

The knowledge acquired in the DSE program was instrumental in the design and implementation of the project.

**Python for Data Analysis** taught us the skills required for preprocessing and exploratory analysis of our 3 data sets.

**Machine Learning** was a great foundation for the team to build and evaluate machine learning models and tune hyperparameters.

**Case Studies in Data Science** was very beneficial for our capstone because Bradley Voytek's presentation to the cohort made us aware of the work he was doing and served as our original introduction. In general, this course was beneficial because it exposed us to the data science work happening in a variety of fields and industries different from the industries we currently work in. The presentations also served as a model for how to best present to large groups and create content suitable for different audience types.

**Probability and Statistics with Python** gave us an understanding of how to evaluate the significance of our results and how to apply sound statistical methods to assess our models.

**Data Analysis Using Hadoop & Spark** served as the foundation for implementing our solution as scale.

**Data Visualization** helped inform our approach on how to best visualize data, create effective and captivating visualizations tailored for the intended audience, and the various tools available. It also made us aware of considerations such as color blindness and color theory.

### B. Data and Software Archive for Reproducibility

Github: https://github.com/voytek/NCM

UCSD Libraries:

> Zeqollari, Adita; Zeqollari, Arlens; Hoye, Erik; Reeves, Robert; Voytek, Bradley (2020). Neural Correspondence Mapping. In Data Science & Engineering Master of Advanced Study (DSE MAS) Capstone Projects. UC San Diego Library Digital Collections.https://doi.org/10.6075/J06M35B6

## C. Related Publications

1. Burt, J.B., Demirtaş, M., Eckner, W.J. et al. Hierarchy of transcriptomic specialization across human cortex captured by structural neuroimaging topography. Nat Neurosci 21, 1251–1259 (2018). https://doi.org/10.1038/s41593-018-0195-0

2. Adolescent consolidation of human connectome hubs

   Kirstie J. Whitaker, Petra E. Vértes, Rafael Romero-Garcia, František Váša, Michael Moutoussis, Gita Prabhu, Nikolaus Weiskopf, Martina F. Callaghan, Konrad Wagstyl, Timothy Rittman, Roger Tait, Cinly Ooi, John Suckling, Becky Inkster, Peter Fonagy, Raymond J. Dolan, Peter B. Jones, Ian M. Goodyer, the NSPN Consortium, Edward T. Bullmore

   Proceedings of the National Academy of Sciences Aug 2016, 113 (32) 9105-9110; DOI: 10.1073/pnas.1601745113

3. Yarkoni T, Poldrack RA, Nichols TE, Van Essen DC, Wager TD. Large-scale automated synthesis of human functional neuroimaging data. Nat Methods. 2011;8(8):665-670. Published 2011 Jun 26. doi:10.1038/nmeth.1635

4. Spatial analysis and high resolution mapping of the human whole-brain transcriptome for integrative analysis in neuroimaging

   Gregor Gryglewski, René Seiger, Gregory Miles James, Godber Mathis Godbersen, Arkadiusz Komorowski, Jakob Unterholzner, Paul Michenthaler, Andreas Hahn, Wolfgang Wadsak, Markus Mitterhauser. Siegfried Kasper, Rupert Lanzenberger https://doi.org/10.1016/j.neuroimage.2018.04.068

5. Voytek JB, Voytek B. Automated cognome construction and semi-automated hypothesis generation. J Neurosci Methods. 2012;208(1):92-100. doi:10.1016/j.jneumeth.2012.04.019

6. Cole S, Voytek B. Cycle-by-cycle analysis of neural oscillations. J Neurophysiol. 2019;122(2):849-861. doi:10.1152/jn.00273.2019

7. Parameterizing neural power spectra
   Matar Haller, Thomas Donoghue, Erik Peterson, Paroma Varma, Priyadarshini Sebastian, Richard Gao, Torben Noto, Robert T. Knight, Avgusta Shestyuk, Bradley Voytek
   bioRxiv 299859; doi: https://doi.org/10.1101/299859

8. Neuronal timescales are functionally dynamic and shaped by cortical microarchitecture
   Richard Gao, Ruud L. van den Brink, Thomas Pfeffer, Bradley Voytek
   bioRxiv 2020.05.25.115378; doi: https://doi.org/10.1101/2020.05.25.115378

## D. Acknowledgements

We are immensely thankful for the support and guidance from our advisor, Bradley Voytek and his colleague Richard Gao. They provided excellent feedback and guidance despite Richard having to worry about his own dissertation. We would also like to thank the entire DSE staff for excellent mentorship and classes over the past two years. We have learned an incredible amount and are excited to leverage this knowledge in the real world. We are also grateful for Professor Ilkay Altintas and her support throughout the capstone project.