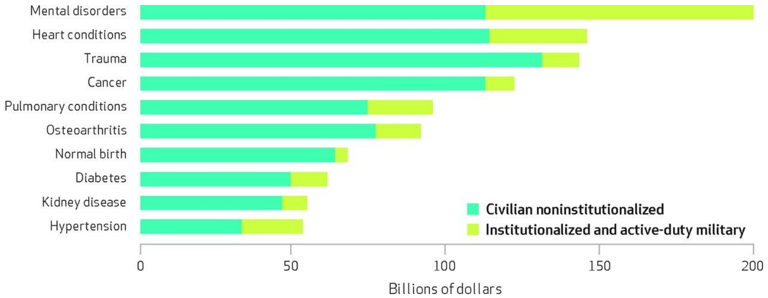
Decomposing fMRI Data with Latent Similarity
Research Meeting

Anton Orlichenko

September 21, 2022

Clinical Problem

- Schizophrenia, ADHD, depression, and other mental illnesses cost the U.S. \$201+ billion annually[1]
- Dementia and Alzheimer's cost the U.S. \$157+ billion annually[2]
- Diagnosis of these diseases may be unreliable until symptoms become severe, when treatment options are more limited



From Roerhig 2016.

[1] Roerhig 2016 https://doi.org/10.1377/hlthaff.2015.1659

[2] Hurd et al. 2013 doi:10.1056/NEJMsa1204629

fMRI and Mental Health

- fMRI can be used to predict disease status and (endo)phenotypes such as age, sex, and general fluid intelligence[3]
- Machine learning predictions of brain age have been correlated with future Alzheimer's diagnosis before clinical symptoms appear[4]
- fMRI has been used for pre-surgical planning, biofeedback, consumer preference identification, and lie detection[5]...
- ...but diagnoses of mental disorders are still made by psychiatrists or physicians based on cognitive tests[6]

---

[3] Qu et al. 2021 10.1109/TBME.2021.3077875

[4] Millar et al. 2022 10.1016/j.neuroimage.2022.119228

[5] Farah et al. https://doi.org/10.1038/nrn3665

[6] https://www.ndcn.ox.ac.uk/divisions/fmrib/what-is-fmri/how-is-fmri-used

fMRI Techniques

- We can monitor neural activity at a coarse level through neurovascular coupling and the BOLD signal
- Many studies measure the activation of specific regions in response to stimulus
- We can also measure the synchronization between different ROIs
  - Functional connectivity
  - Effective connectivity
  - Dynamic connectivity
- Other techniques like ReHo[7] exist, and may be better in some circumstances

[7] Zhang et al. 2020 https://doi.org/10.3389/fnhum.2020.00244

Technical Challenges

It's hard to find the signal.

Problem 1: Small Study Size

In 2017-2018, only 1% of fMRI studies had more than 100 subjects.[a]

[a] Szucs and Ioannidis 2020 10.1016/j.neuroimage.2020.117164

Problem 2: High Dimensionality and Noise

Functional connectivity may have tens of thousands of features. The best feature may have only a 4% correlation with the response variable.[a]

[a] Author's observations

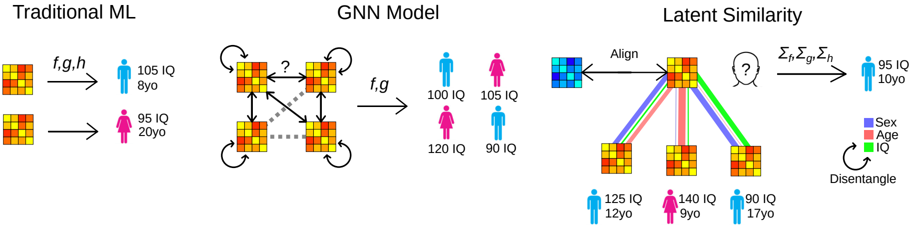
Project Overview

1. Use *Latent Similarity* to solve Problem 1
2. Use *Connectivity Decomposition* to solve Problem 2
3. Develop and share tools to encourage reproducibility
4. Create visualization software to identify important features

Part 1: Latent Similarity (LatSim)

LatSim Overview

The idea is to use the $\mathcal{O}(n^2)$ connections between the subjects rather than the features of the $\mathcal{O}(n)$ subjects themselves.



Metric Learning

A distance function (metric) satisfies the following conditions:

1. (Positivity) $d(x, y) > 0$
2. (Identity of indiscernibles) $d(x, y) = 0 \iff x = y$
3. (Symmetry) $d(x, y) = d(y, x)$
4. (Triangle inequality) $d(x, z) < d(x, y) + d(y, z)$

Metrics include Euclidean distance, Mahalanobis distance, and (the possibly learned) generalized Mahalanobis distance.

$$\begin{aligned} ||\mathbf{x}_i - \mathbf{x}_j||_2^2 &= (\mathbf{x}_i - \mathbf{x}_j)^T \mathbf{I} (\mathbf{x}_i - \mathbf{x}_j) \\ ||\mathbf{x}_i - \mathbf{x}_j||_{\mathbf{\Sigma}}^2 &= (\mathbf{x}_i - \mathbf{x}_j)^T \mathbf{\Sigma}^{-1} (\mathbf{x}_i - \mathbf{x}_j) \\ ||\mathbf{x}_i - \mathbf{x}_j||_{\mathbf{W}}^2 &= (\mathbf{x}_i - \mathbf{x}_j)^T \mathbf{W} (\mathbf{x}_i - \mathbf{x}_j) \end{aligned} \tag{1}$$

Extensions and Deep Metric Learning

Metric learning is popular in machine learning on images and is related to contrastive learning. Some examples of metric learning are[8]:

- Fisher Discriminant Analysis
- Fisher-HSIC Multi-view Metric Learning
- Adversarial Metric Learning
- Neighborhood Component Analysis
- Noisy Contrastive Estimation and Negative Sampling
- Siamese Networks and Triplet Loss
- Label Propagation

[8] Ghojogh et al. 2022 https://doi.org/10.48550/arXiv.2201.09267

Similarity Kernel

We learn a metric or similarity score between pairs of subjects.

$$\begin{aligned} sim(a, b) &= \langle \phi(\mathbf{x}_a), \phi(\mathbf{x}_b) \rangle \\ sim(a, b) &= \mathbf{x}_a \mathbf{A} \mathbf{A}^T \mathbf{x}_b^T, \end{aligned} \tag{2}$$

$\langle \cdot, \cdot \rangle$ is the inner product
$\mathbf{x}_a, \mathbf{x}_b \in \mathbb{R}^d$ are feature vectors for subjects a and b, respectively
$\phi(\mathbf{x}_a)$ is a low-dimensional projection
$\mathbf{A} \in \mathbb{R}^{d \times d'}$ is the learned kernel matrix implementing the low-dimensional projection

Population Graph

We then ensure the sum of each subject's similarity to other subjects equals 1 using the softmax function.

$$\begin{aligned} \mathbf{M} &= diag(\infty), \\ \mathbf{E} &= S_{Row}((\mathbf{1} - \mathbf{M}) \odot \mathbf{XAA}^T\mathbf{X}^T), \\ S(\mathbf{z})_i &= \frac{e^{z_i/\tau}}{\sum_{j=0}^{N} e^{z_j/\tau}}, \end{aligned} \tag{3}$$

$\mathbf{E} \in \mathbb{R}^{N \times N}$ is the final similarity matrix
$\mathbf{M} \in \mathbb{R}^{N \times N}$ is a mask to remove self-loops in predictions
$\mathbf{X} \in \mathbb{R}^{N \times d}$ is the feature matrix
$\mathbf{A} \in \mathbb{R}^{d \times d'}$ is the learned kernel taking connectivity features to a lower latent dimension
$S(\mathbf{z})_i$ is the softmax function with temperature τ

Estimation and Training

The response variable estimate is found by multiplying the training set response by the similarity matrix.

$$\hat{\mathbf{y}} = \mathbf{E}\mathbf{y}_{train} \tag{4}$$

Training is performed via gradient descent, with parameters to control sparsity, disentanglement between tasks, and alignment between modalities.

Feature Selection

- We utilized a greedy feature selection algorithm, made possible by the high computational efficiency of LatSim.
- The algorithm selects connections (features) one at a time by ranking their ability to separate dissimilar subjects, i.e., their ability to minimize similarity between subjects that are "far apart" with regards to the current residual.

Greedy Feature Selection Algorithm

$$\begin{aligned}
\mathbf{r}^{(i)} &= LatSim(\mathbf{X}_{F_{i-1}}, \mathbf{y}) - \mathbf{y}, \\
D_{ab} &= (r_a^{(i)} - r_b^{(i)})^2, \\
\mathbf{D} &= \mathbf{D} - \frac{1}{N^2}\Sigma_{ab} D_{ab}, \\
F_i &= F_{i-1} \cup \{\operatorname*{argmin}_j \Sigma_{ab}(D_{ab} X_{aj} X_{bj})\},
\end{aligned} \tag{5}$$

$LatSim : \mathbb{R}^{N \times d+1} \to \mathbb{R}$ is the predictive model
$r_a^{(i)}$ is the residual at iteration i for subject a
$\mathbf{D} \in \mathbb{R}^{N \times N}$ is a centered matrix of differences between residuals
$F_i = \{0 \ldots i\}$ is the set of selected connections at iteration i
$\mathbf{X} \in \mathbb{R}^{N \times d}$ is the vectorized matrix of connections for all subjects
$\mathbf{y} \in \mathbb{R}^N$ is the response variable

Post-Hoc Feature Importance

We also estimated feature importance from model weights using a post-hoc algorithm.

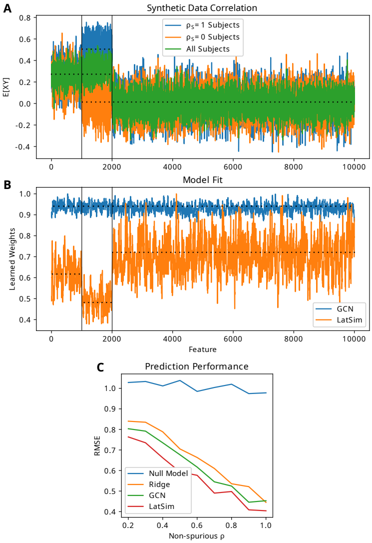
$$F = \underset{j}{\text{argsort}}\ \Sigma_{abd}\ (D_{ab} W_{dj}^2 X_{aj} X_{bj}), \tag{6}$$

Here the residual is set to the response variable,
D is calculated as before
$\mathbf{W} \in \mathbb{R}^{d \times d'}$ is the set of model weights, i.e., the kernel **A**
F is the resulting set of ranked features

Simulation Study

- We performed a simulation on a synthetic dataset, with $d = 10,000$ features, $N_{train} = 40$ training subjects, and $N_{test} = 120$ test subjects.
- The first $1,000$ features were correlated with the response variable with $\rho = 0.5$.
- The second $1,000$ features were correlated in only half of subjects at a variable $\rho_S \in [0.2, 1]$.
- $X_{ij} \sim \mathcal{N}(0, 2)$
- $y_i \sim \mathcal{N}(0, 1)$

Simulation Results

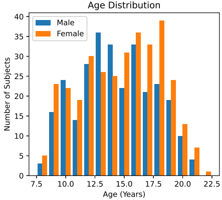


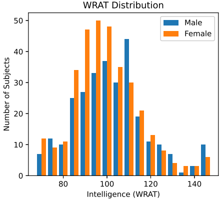
Simulation Conclusions

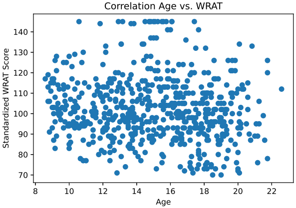
1. LatSim has a small predictive advantage over GCN, which has a small predictive advantage over Ridge Regression.
2. LatSim can identify all three classes of features (correlated, partially correlated, and non-correlated), while the GCN can't.

Brain Development Study

We used the fMRI scans of 620 subjects from the Philadelphia Neurodevelopmental Cohort (PNC) dataset to predict subjects' age, sex, and general intelligence over 10 CV splits.



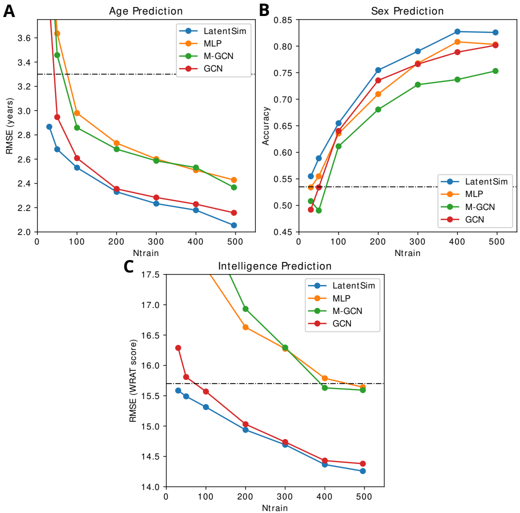




	Number of Subjects
Males	286
Females	334
Total	620

	Min	Mean	Max
Age (months)	103	180$\pm$39	271
Age (years)	8.6	15$\pm$3.3	22.6
WRAT score	70	102$\pm$15.7	145

Prediction Results



Dashed line represents the null model.

Prediction Accuracy as Function of Cohort Size

	Age (RMSE, years)		Sex (Accuracy)		Intelligence (RMSE, WRAT score)	
Model	N=30	N=496	N=30	N=496	N=30	N=496
Null	3.3		0.54		15.7	
M-GCN	4.47	2.37	0.51	0.75	23.27	15.59
MLP	4.52	2.43	0.53	0.8	21.17	15.64
GCN	3.89	2.16	0.49	0.8	16.29	14.38
LatSim	2.86	2.05	0.55	0.82	15.59	14.26
p-value	**2.2e-6**	**5e-3**	0.32	0.11	**0.02**	0.30

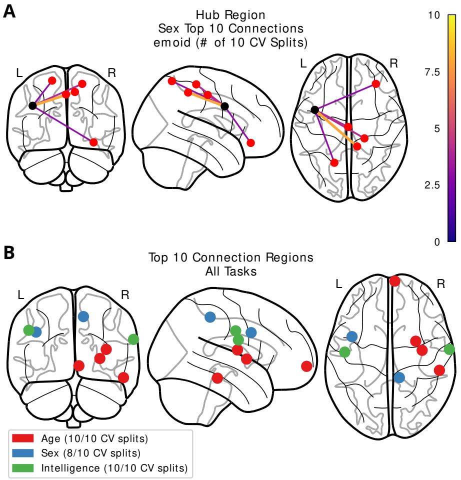
- LatSim is much better than GCN at up to 50 training subjects for age and intelligence prediction, equal after that.
- All models perform close to the same for sex prediction.

Computational Efficiency

Model	LatSim	GCN	MLP	M-GCN
Epochs	200	1e4	1e4	5e3
Training Time	**4.3s**	406s	364s	5912s

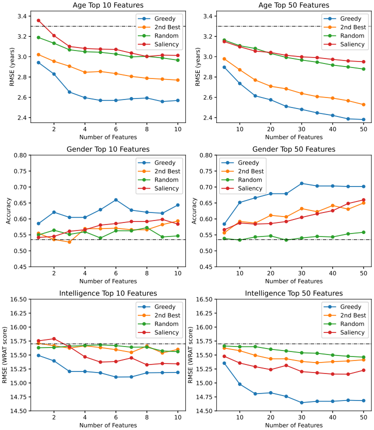
LatSim is almost as fast as linear methods, and almost 2 orders of magnitude faster than other deep models.

Key Connections



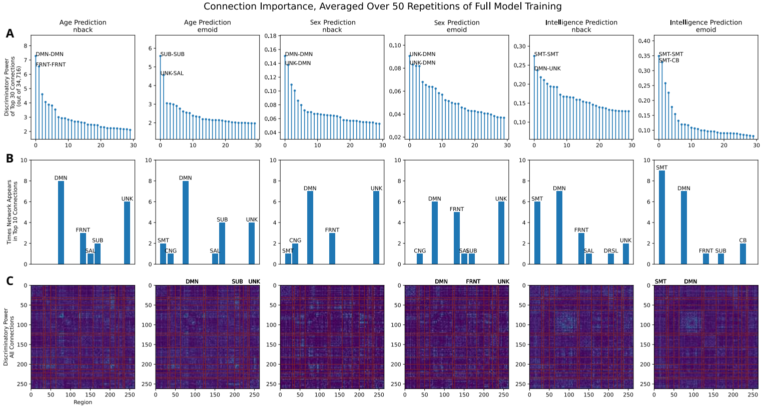
Using the greedy algorithm, we identified several connections appearing in the majority (sometimes 100%) of CV splits for the top 10 connections.

Greedy Selection is Superior to Other Interpretability Methods



Most predictive information is found in 1-5 connections, and adding more features only slowly improves prediction accuracy.

Default Mode and Uncertain Network Regions



ROIs from the DMN and UNK functional networks are over-represented in connections important for age, sex, and intelligence prediction.

Part 2: Decomposition of Brain Connectivity

Useful Information

- It is possible to identify around 14 functional networks from functional connectivity.
- A 264-ROI template gives rise to 34,716 unique connections.
- 1-5 connections give most of the useful information for a predictive task, but the actual connections vary from task to task.

The Autoencoder Problem

The Autoencoder Problem

Is it possible to summarize connectivity data in a small number of variables, independent of predictive task?

Dictionary Learning

- The idea is to create a codebook, in the spirit of dictionary learning, and use it for subsequent tasks.
- Previous works[9] used a codebook of $K = 8$ rank-1 matrices[10], tied to a specific predictive task.

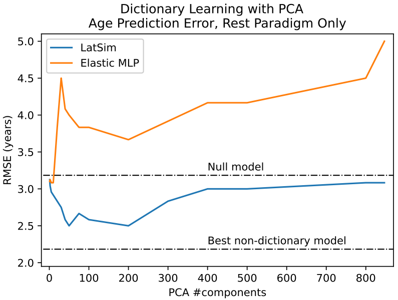
$$\begin{aligned}
\mathcal{D} &= \sum_n (||\mathbf{\Gamma}_n - \mathbf{X} diag(\mathbf{c}_n)\mathbf{X}^T||_F^2 + \gamma_2||\mathbf{c}_n||_2^2) + \gamma_1||\mathbf{X}||_1 \\
\hat{y}_n &= MLP_\theta(\mathbf{c}_n) \\
\mathcal{L} &= \lambda \sum_n ||y_n - \hat{y}_n||^2
\end{aligned} \tag{7}$$

[9] D'Souza et al. 2019 https://doi.org/10.1007/978-3-030-32248-9_79

[10] The authors stated this was the "knee" of the eigenspectrum of $\mathbf{\Gamma}$

Missing the Manifold

Why are these codebooks based on downstream tasks (i.e., learned in a supervised manner rather than inferred from the data)?



Unsupervised dictionary learning does not seem to capture the structure of the manifold.

Connectivity Decomposition

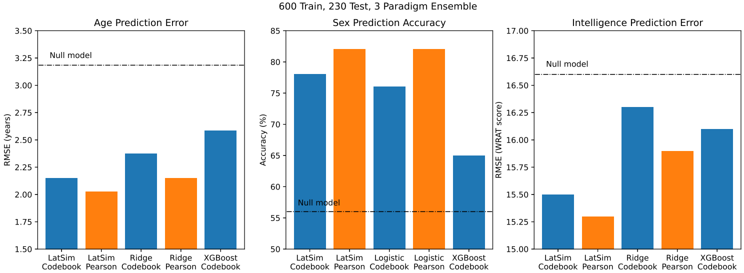
- We believe that not enough codes are being used (the previous graph suggests there is an optimum number greater than 100).
- Rank-1 matrices may not capture meaningful information about functional connectivity.

Mixed-Rank Codebook

Our idea is to construct a codebook from mixed ranks.

$$\begin{aligned}\mathcal{B} &= \{\mathbf{A}\mathbf{A}^T \mid \mathbf{A} \in \mathbb{R}^{d \times r_i},\ r = \{r_1, r_2, \ldots, r_M\},\ r_i < d\} \\ \hat{\mathbf{X}}_n &= \sum_i w_{in} \mathcal{B}_i \\ \mathcal{L} &= \sum_n ||\mathbf{X}_n - \hat{\mathbf{X}}_n||_F^2\end{aligned} \tag{8}$$

Codebook Preliminary Results



- 300 entry codebook of rank-120 matrices
- This *task-agnostic* autoencoder reduces data dimensionality by 2 orders of magnitude, from $d = 34,716$ to $d' = 300$, while maintaining predictive accuracy.

Effective Connectivity

We want to apply the codebook idea to effective connectivity.
There are several popular effective connectivity frameworks:

- Granger causality[11]
- Spectral dynamic causal modeling[12][13]
- Transfer entropy[14]

Problem: Granger causality may give poor results and is computationally expensive, but there may be opportunities for optimization. Many effective connectivity methods require small numbers of ROIs/signals[15].

11 Kassani et al. 10.1109/TMI.2020.2990371
12 Park et al. 2018 https://doi.org/10.1016/j.neuroimage.2017.11.033
13 Zhargami and Friston 2020 https://doi.org/10.1016/j.neuroimage.2019.116453
14 Ursino et al 2020 https://doi.org/10.3389/fncom.2020.00045
15 Hidalgo-Lopez et al. 2021 https://doi.org/10.1038/s42003-021-02447-w

Dynamic Connectivity

We want to apply the codebook idea to dynamic functional and effective connectivity.

- The time-varying graphical LASSO (TVGL) method has been used to estimate dynamic FC[16], but has not been shown to be superior in downstream tasks.
- Dynamic effective connectivity has been proposed by Friston, but is based on computationally inefficient methods, limiting its scope.

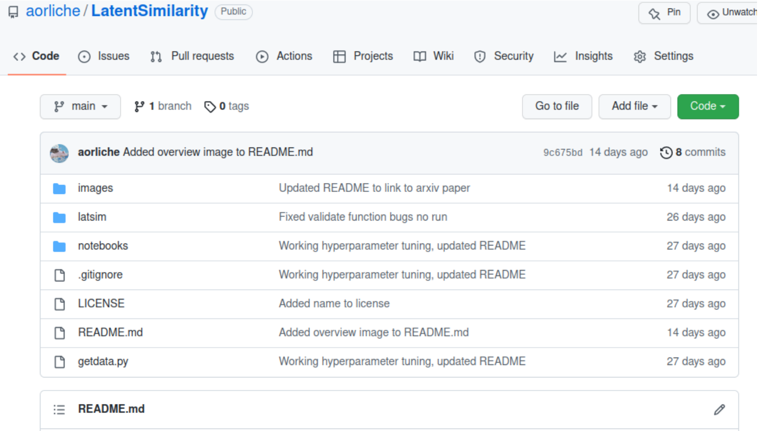
Objective

Our goal is to use a large, *empirically validated* codebook to track changes in connectivity while the subject undergoes scanner tasks.

[16] Cai et al. 10.1109/TBME.2018.2880428

Part 3: Tools and Reproducibility

LatSim Python Package



Currently available on GitHub at
https://github.com/aorliche/LatentSimilarity/

Downloading and Using LatSim

```
[ ] !git clone https://github.com/aorliche/LatentSimilarity

Cloning into 'LatentSimilarity'...
remote: Enumerating objects: 38, done.
remote: Counting objects: 100% (38/38), done.
remote: Compressing objects: 100% (25/25), done.
remote: Total 38 (delta 15), reused 30 (delta 9), pack-reused 0
Unpacking objects: 100% (38/38), done.
```

```python
import sys

sys.path.append('/content/LatentSimilarity')

from latsim import LatSim
from latsim.util import validate

print('Complete')
```

```
Complete
```

We are working on a Pip package, but the GitHub code is very easy to download and use.

Hyperparameter Tuning

```
In [7]:
# Tune hyperparameters
# Look for get_default_distributions() in train.py to see the range of hyperparameters being tuned
# Hyperparameter tuning only supported for single-task models (multi-modal allowed)
# Use the LatSim class (in latsim.py) directly for multi-task models

import sys

sys.path.append('..')

from latsim.train import tune

# Make splits (regular cv may be unreliable)
splits = []
for _ in range(40):
    idcs = np.arange(66)
    np.random.shuffle(idcs)
    splits.append((idcs[:50],idcs[50:]))

best = tune(X, y, 'class', n_iter=50, cv=splits)

print('Complete')
```

```
Complete
```

- Example Jupyter notebooks with test datasets are included in the GitHub repository.
- We include a scikit-learn interface with a function for hyperparameter tuning.

Part 4: Data Visualization

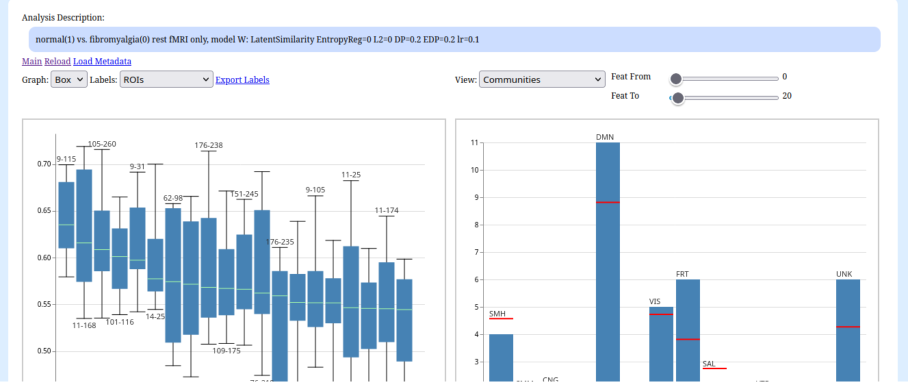
ImageNomeR

- Performing data exploration may require lengthy and repetitive code editing.
- ImageNomeR (**Image** ge**Nome** explore**R**) displays some commonly useful graphs.
- Currently available on GitHub at https://github.com/aorliche/ImageNomeR/

LatSim/ImageNomeR Demo

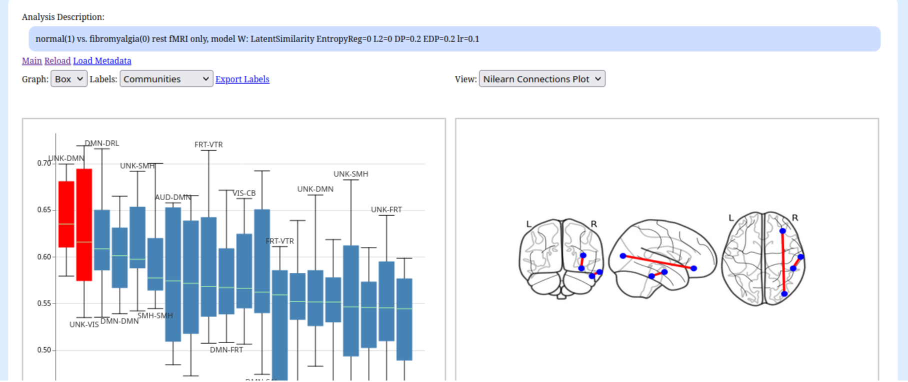
- An interactive demo is running on a Linode cloud instance.
- Go to https://aorliche.github.io/LatSim/ and click on the demo link.

Multiple Useful Graphs



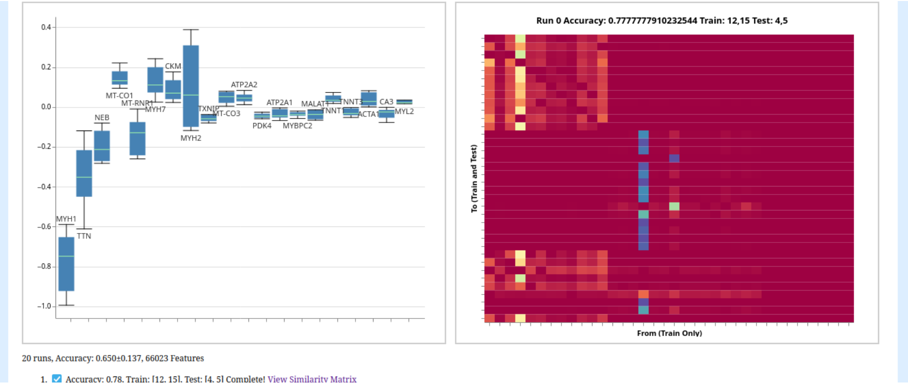
ImageNomeR includes bar graphs and box plots of top features, as well as functional network and connection summary graphs.

Nilearn Integration



You can visualize significant connections or regions via a point and click interface.

Interactive Population Similarity



You can view the population-level similarity matrix. Clicking on a matrix element gives a subject-level breakdown.

Thank you!
Any questions?