NeuroData SIMPLEX Report: April 2017

The following report documents the progress made by the labs of PI Joshua T. Vogelstein and Co-PIs Randal Burns and Carey Priebe at Johns Hopkins University towards goals set by the DARPA SIMPLEX grant.

Contents

1 Bibliography

Manuscripts

[1] D. Mhembere, D. Zheng, C. E. Priebe, J. T. Vogelstein, and R. Burns, "knor: A NUMA-optimized In-memory, Distributed and Semi-external-memory k-means Library," 2017.

[2] C. Shen, C. E. Priebe, M. Maggioni, and J. T. Vogelstein, "Discovering Relationships Across Disparate Data Modalities," 2017.

[3] G. Kiar, K. J. Gorgolewski, D. Kleissas, W. Gray Roncal, B. Litt, B. Wandell, R. A. Poldrack, M. Wiener, R. Vogelstein, R. Burns, and J. T. Vogelstein, "Science In the Cloud (SIC): A use case in MRI Connectomics," GigaScience, vol. gix013, mar 2017.

[4] Y. Lee, C. Shen, and J. T. Vogelstein, "Network Dependence Testing via Diffusion Maps and Distance-Based Correlations," Journal of the American Statistical Association, 2017.

Invited Talks

[1] D. Zheng, "FlashR: Parallelize and Scale R Machine Learning Libraries with Extreme Efficiency for Big Data," rstudio conference 2017, Jan 2017, Invited talk.

2 Statistical Theory and Methods

2.1 LOL @jovo

This quarter we generate simulation results supporting those theoretical claims (see Figure 1).

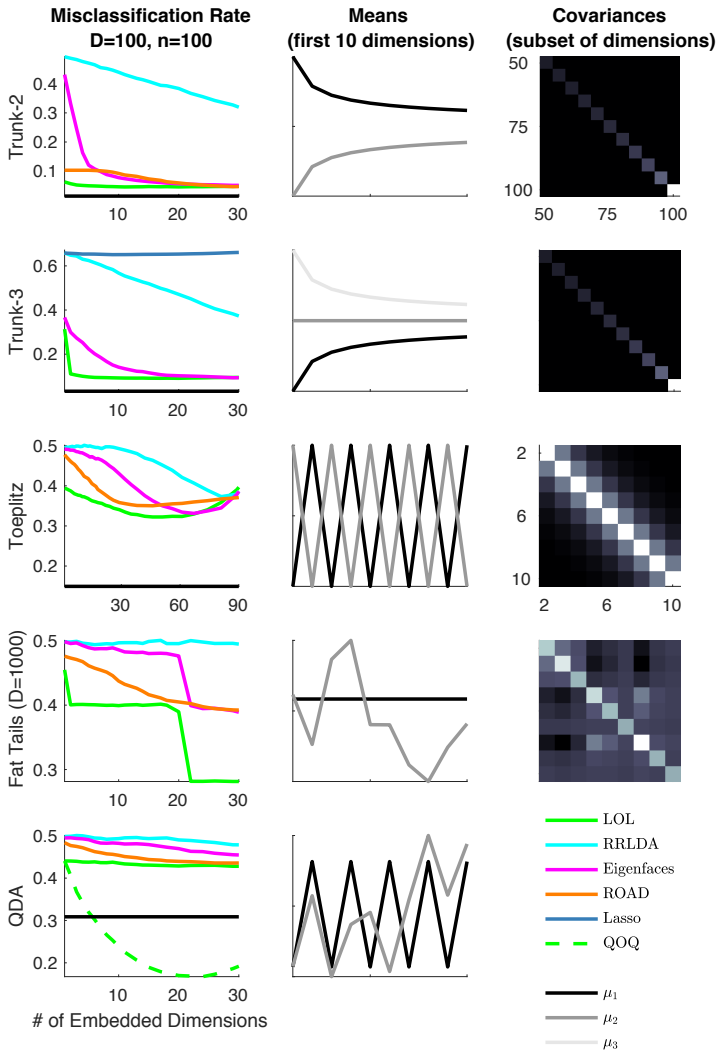


Figure 1: LOL outperforms other linear classifiers in a wide variety of settings, including those for which we have proven LOL should (rows 1 and 3), and those beyond our current theoretical grasp (rows 2, 4, and 5). This is true regardless of the dimensionality into which we embed.

We also finalized the real data analysis using LOL. In particular, we considered four datasets, the Prostate and Colon datasets have extensively been studied in the sparse literature. LOL yields better perforance, and for Colon, with lower dimensionality. MNIST is an even more prominent dataset, LOL achieves the best performance for all dimensions. MRN is a new dataset that we generated; it has over 500 million features, and 112 samples. We subsampled to 100 samples for cross-validation purposes. To our knowledge, no other machine learning tool is capable of even operating on 500 million features. Moreover, we demonstrate that our implementation outperforms first doing PCA on the data, for any number of dimensions that we embed into. We then also investigate the amount of time it takes to run LOL on very wide datasets. For a 128 million dimensional dataset, with 2000 samples, requiring nearly half a terabyte of space just to store, we have an approximate implementation that runs on a single machine and only takes about 3 minutes.

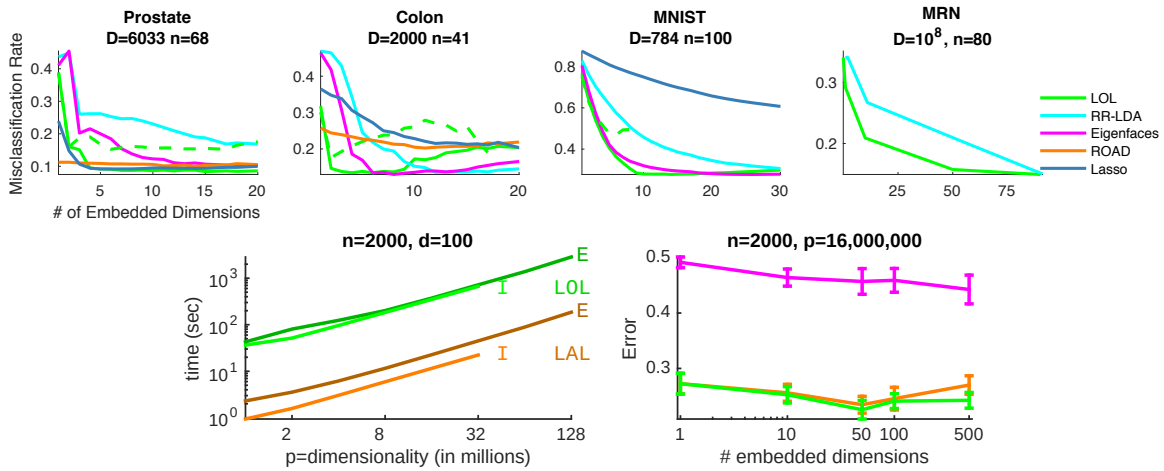


Figure 2: Top: LOL outperforms essentially all other methods for essentially all problems and number of dimensions to embed into. Bottom: LOL (green) and LAL (brown, using random projections to approximate) for both fully in memory (IE) and semi-external memory (E) implementations. Magenta compares performance to our scalable implementation of eigenfaces on the same problem.

2.2 meda @JesseLP

Matrix Exploratory Data Analysis (meda) is a package being developed to allow for easy generation of modern summary statistics effective for high-dimensional data analysis.

- Source code: https://github.com/neurodata/meda
- Example output generated from Fisher's Iris data is here: http://docs.neurodata.io/meda

The goal of this package is to realize the following checklist: Given a new set of n samples of vectors in $\mathbb{R}^d$

1. histogram of feature types (binary, integer, non-negative, character, string etc.)
2. # NaNs per row? Per column? Infs per row? Per column? "Zero" variance rows? columns?
3. Heat map of raw data that fits on screen (k-means++ to select 1000 samples, CUR to select 100 dimensions)
4. 1st moment statistics
   (a) mean (line plot + heatmap)
   (b) median (line plot + heatmap)
5. 2nd moment statistics
   (a) correlation matrix (heatmap)
   (b) matrix of energy distances (heatmap)
6. density estimate
   (a) 1D marginals (Violin + jittered scatter plot of each dimension, if n > 1000 or d>10, density heatmaps)
   (b) 2D marginals (Pairs plots for top 8 dimensions, if n*d>8000, 2D heatmaps)
7. Outlier plot
8. cluster analysis (IDT++)
   (a) BIC curves
   (b) mean line plot
   (c) covariance matrix heatmaps
9. spectral analysis
   (a) cumulative variance (with elbows) of data matrix
   (b) eigenvectors (pairs plot + heatmap)

- To rescale the data in case of differently scaled features, we will implement the following options:
  - raw
  - linear options
    * linear squash between 0 & 1
    * mean subtract and standard deviation divide
    * median subtract and median absolute deviation divide
    * make unit norm
  - nonlinear
    * rank
    * sigmoid squash
- To robustify in the face of outliers, we will utilize Geometric median and robust estimation in Banach spaces
- if features have categories

1. sort by category
2. color code labels by category

- if points have categories: label points in scatter plots by symbol

For point 6 (a) in the above checklist we have developed functionality in `meda` to plot 1-dimensional heatmaps. The 1D heatmap is a different representation of a histogram, using color to denote count instead of bin height, see figure 3.

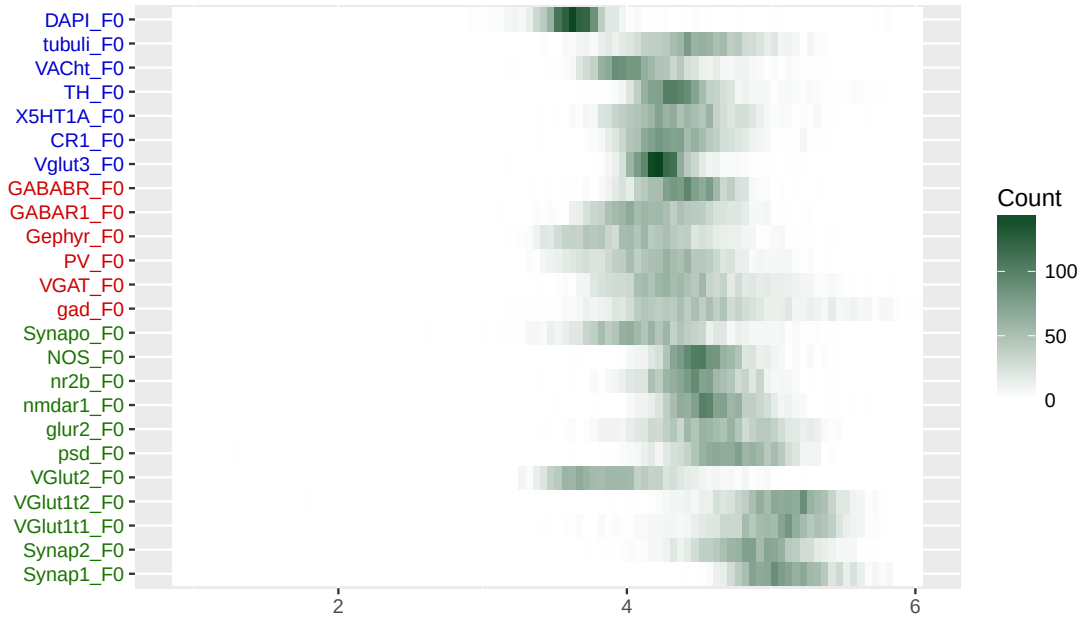


Figure 3: A one dimensional heatmap for each of the $\log_{10}$ transformed feature columns of the Kristina15 synaptome dataset. Colors correspond to the count of data points falling in to each bin. Scott's binning streategy is used which, in this case, yields 105 bins of equal width $w = 0.05$. Label colors and groups starting from the bottom: Synap1_F0 - Synapo_F0 (green, excitatory), gad_F0 - GABABR_F0 (red, inhibatory), Vglut3_F0 - DAPI_F0 (blue, other).

Updates in meda include the addition of plots that explore the clusters generated by hierarchical clustering. We are using mclust in our hierarchical clustering function. At each level we use the Bayseian Information Criterion (BIC) to determine if the data should be split into two clusters or kept as one. The dendrogram (figure 4 left) shows the binary tree clustering structure with branch size denoting the size of the cluster. The stacked level means plot (figure 4 right) shows the means of features in each node of the tree.
Updates: Colored markers indicating whether BIC suggests $K = 2$ (green triangle) or to stop with $K = 1$ (red squares) have been added to the dendrograms.

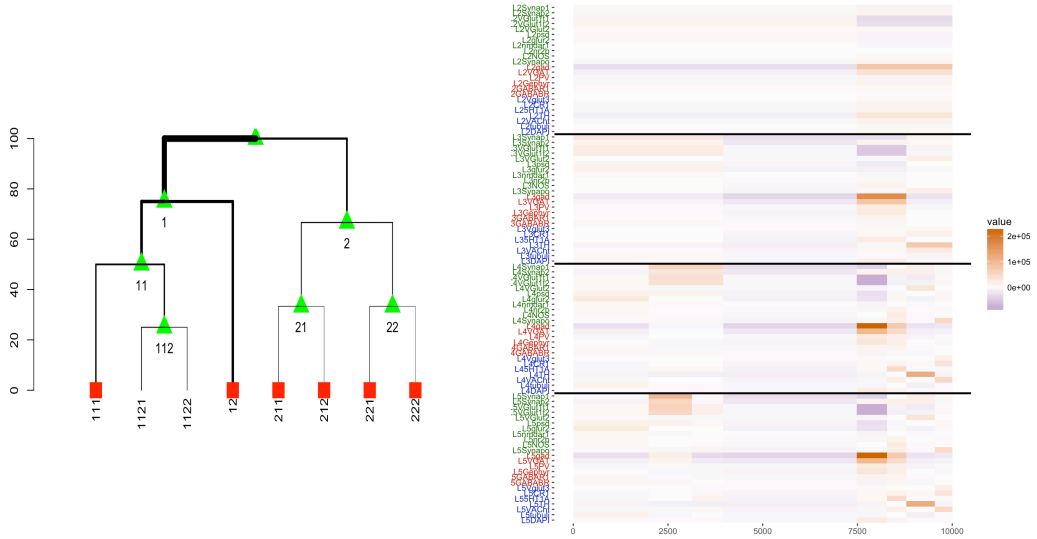


Figure 4: Left: A dendrogram showing the results of hierarchical mclust. The splits are constrained to be binary and branch sizes show relative cluster sizes. Right: A stacked level means plot showing for each node in the dendrogram the feature means.

The internals of the meda package are now being re-worked so that data processing and plotting of results will be separate. This will allow for better modularity and faster turn-around.

2.3 Multiscale Generalized Correlation (MGC)

We developed the Multiscale Generalized Correlation method to better detect associations between two datasets X and Y. We demonstrate that Oracle MGC is a consistent test statistic (power converge to 1 as sample size increases) under standard regularity conditions, is equivalently to the global correlation under linear dependency (i.e., each observation X_i is a linear transformation of Y_i), and can be strictly better than the global correlation under common nonlinear dependencies. Thus Oracle MGC dominates the global correlation, and the sample MGC (i.e., choose the optimal scale by p-value map approximation, as the testing power are not available in the absence of the true model and training data) also empirically dominates the global correlation. The performance advantage of MGC is shown in Figure 5.

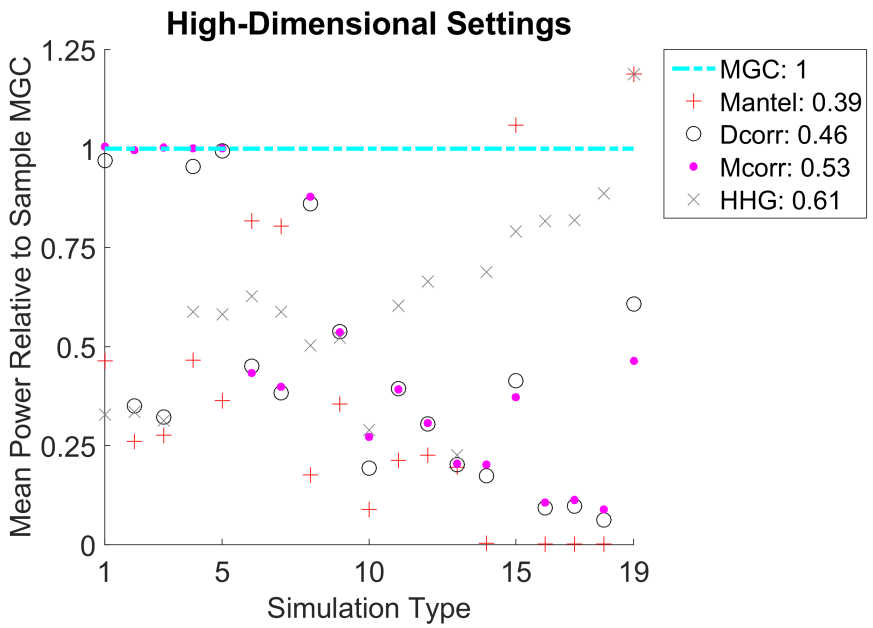


Figure 5: The average power advantage of MGC versus popular benchmarks, throughout 20 different linear and nonlinear high-dimensional dependencies.

Upon final draft polishing and addressing feedback from statisticians and biologists, the newest draft is updated to arXiv and submitted for publication.

2.4 Network Dependence Test via Diffusion Maps and MGC

Deciphering the association between network structures and corresponding nodal attributes of interest is a core problem in network science. We propose a new nonparametric procedure for testing dependence between network topology and nodal attributes, via diffusion maps and `MGC`. Specifically, under an exchangeable graph, we verify that the diffusion maps provide a set of conditionally independent multivariate coordinates for the nodes, which can be combined with `MGC` (or in general, any distance-based correlation measures) to yield consistent statistic for network dependence testing. In simulation, the new approach achieves superior testing performance under a variety of common network models than existing benchmarks. The diffusion maps provides a robust metric compared to adjacency matrix or geodesic distance, while `MGC` can better capture nonlinear dependencies, with their combined advantages shown in Figure 6.

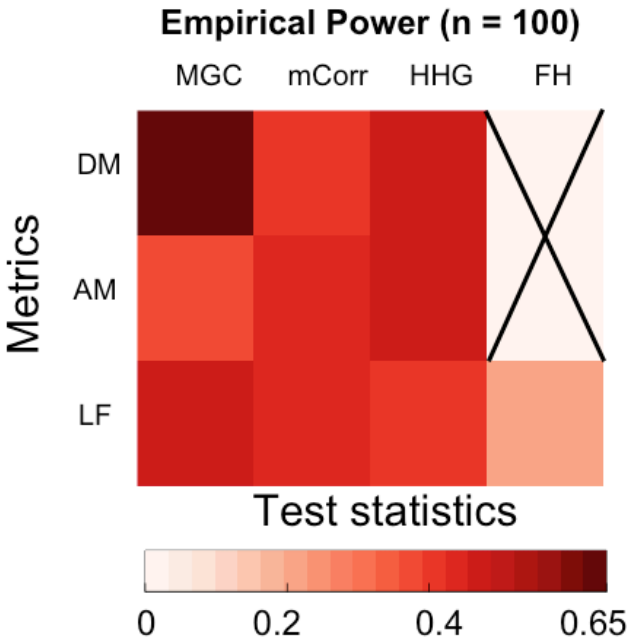


Figure 6: Power comparison for all possible combinations of metrics and correlation measure, under the stochastic block model with three blocks. `MGC` with the diffusion maps (DM) yields the best power, comparing to using other metrics like adjacency matrix (AM), latent factors (LF), and other test statistics like distance correlation (mcorr), Heller-Heller-Gorfine (HHG) test, or Fosdick and Hoff (FH) method.

This month we made significant progress in writing the manuscript and improving the exposition. The current draft was submitted to ASA Nonparametric Statistics Section Student Paper Awards, and we are notified as finalists for awards and special presentation section in the Joint Statistical Meeting this year.

Deciphering the association between network structures and corresponding nodal attributes of interest is a core problem in network science. We propose a new nonparametric procedure for testing dependence between network topology and nodal attributes, via diffusion maps and MGC. Specifically, under an exchangeable graph, we verify that the diffusion maps provide a set of conditionally independent multivariate coordinates for the nodes, which can be combined with MGC (or in general, any distance-based correlation measures) to yield consistent statistic for network dependence testing. Moreover, our method is computationally inexpensive and robust against parameter mis-specifications, very efficient in capturing a wide variety of nonlinear and high-dimensional relationships, and readily extend-able to testing independence between two graphs.

Figure 7 illustrates the advantage of the proposed method on testing dependency between two graphs. The graphs are simulated by the random dot product graph, with the underlying latent variables being related by a quadratic function. By repeatedly generating dependent sample graphs, the testing power equals the percentage of rejection of the independence hypothesis. Although all methods are consistent (having power 1 as number of vertices increases), the proposed approach using MGC is able to achieve perfect testing power at a very small size, which is significantly better than other benchmarks.

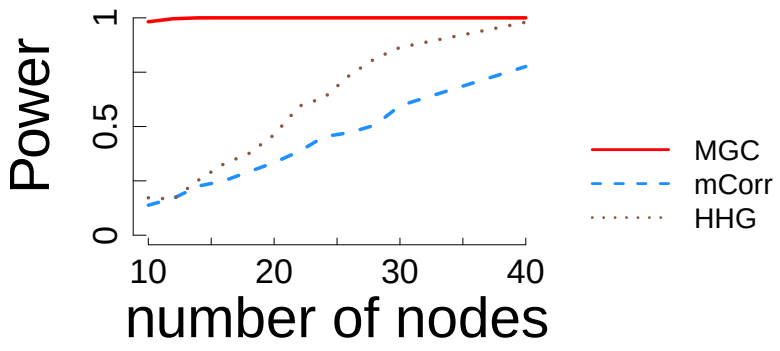


Figure 7: The power curve with respect to increasing number of vertices for the two-graph dependency testing simulation. The proposed approach quickly attains perfect power at a very small vertex size, while other benchmarks often require a much larger graph for perfect testing.

An early draft is recently awarded the Best Student Paper Awards by the American Statistical Association Nonparametric Statistics Section, which will be presented in a special section in the Joint Statistical Meeting this year. We collected and addressed feedback from experts in graph inference, and submitted the complete manuscript this month.

2.5 Randomer Forest (RerF)

This quarter, we compared classification performances of RF, RerF, RR-RF, and XGBoost on 119 benchmark datasets. RR-RF is identical to RF except that the data is randomly rotated prior to building each tree. XGBoost is a computationally efficient implementation of gradient boosted trees and has been the winner of many recent Kaggle competitions. For each dataset, for each algorithm, error was subtracted by that of RF and normalized by the chance probability of error. These normalized relative errors were then binned and the counts in each bin were computed. The y-axis represents the bins. Color indicates how many times the normalized relative error of an algorithm fell into a particular bin. For instance, the figure shows that RerF had a normalized relative error 0.05 to 0.10 less than that of RF on approximately 15 datasets. The "0 to 0" bin indicates the number of times the normalized relative error was exactly 0. Overall the figure indicates that RerF rarely loses to RF by much and frequently does substantially better. RR-RF and XGBoost, on the other hand, frequently perform worse than RF by a large margin.

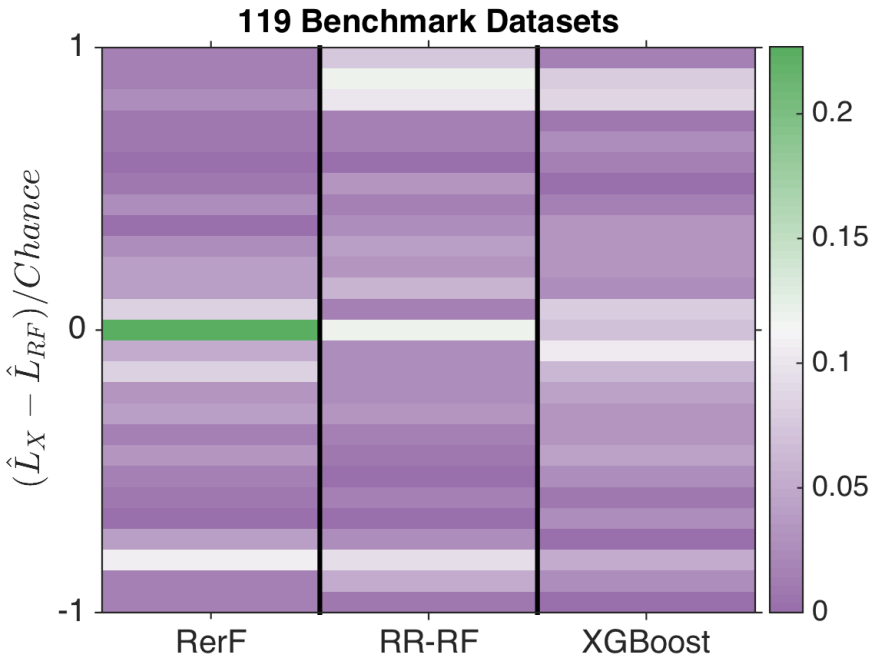


Figure 8: Classification performances of RF, RerF, RR-RF, and XGBoost on 119 benchmark datasets.

Additionally, we have made progress towards a more scalable implementation of RerF in R. The previous port of RerF to R rotated input data at the tree level instead of at the more desired node level. The new port of RerF now rotates at the node level. The change to R will allow a fast RerF implementation by integrating the algorithm into FlashR. This tool can be found here.

RerF is now working in pure R code. The running time of this implementation is roughly the same as the Matlab implementation for various sizes of input. We are re-implementing portions of the code in both C and FlashR in an attempt to reduce the running time of the algorithm.

Previously, we did not have any simulation experiments in which we know for a fact that RF is the "right" thing to do. We conducted such experiments to see how much RerF and RR-RF lose by allowing oblique split directions. The simulated datasets were constructed as follows. Data was sampled in p dimensions over a unit hypercube centered at the origin. Datapoints all falling into the same orthant were assigned the same class label. Therefore, for p dimensions, there are 2^p unique class labels. The true decision boundary separating the classes is purely axis-aligned. In such a case, RF is the best classifier among the class of all ensemble tree classifiers.

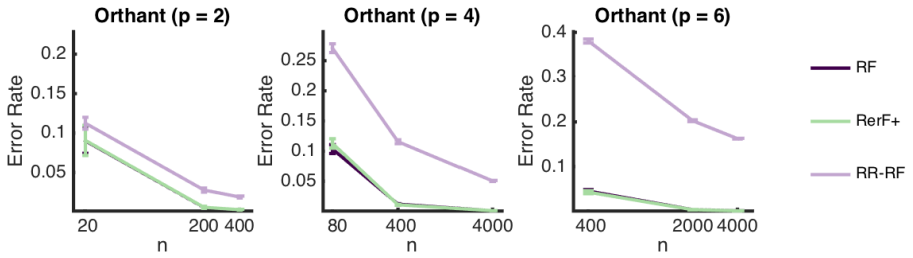


Figure 9: Error rate of RF, RerF, and RR-RF on the "Orthant" dataset as a function of n, the number of training samples, for three values of p. The results indicate that there is no significant difference in performance between RerF and RF, while RR-RF performs significantly worse across all settings.

Most recently, we have begun investigating the theoretical behavior of RerF. Mathematical analysis of the RF and RerF procedures is difficult. Therefore, we have started with simplified procedures. The main simplifications we have made are that trees only have a depth of one (also known as decision stumps), and that each tree is trained on the full training set rather than a bootstrap sample. In the figure, we surprisingly see that RerF outperforms RF across a variety of settings (see caption for more details).

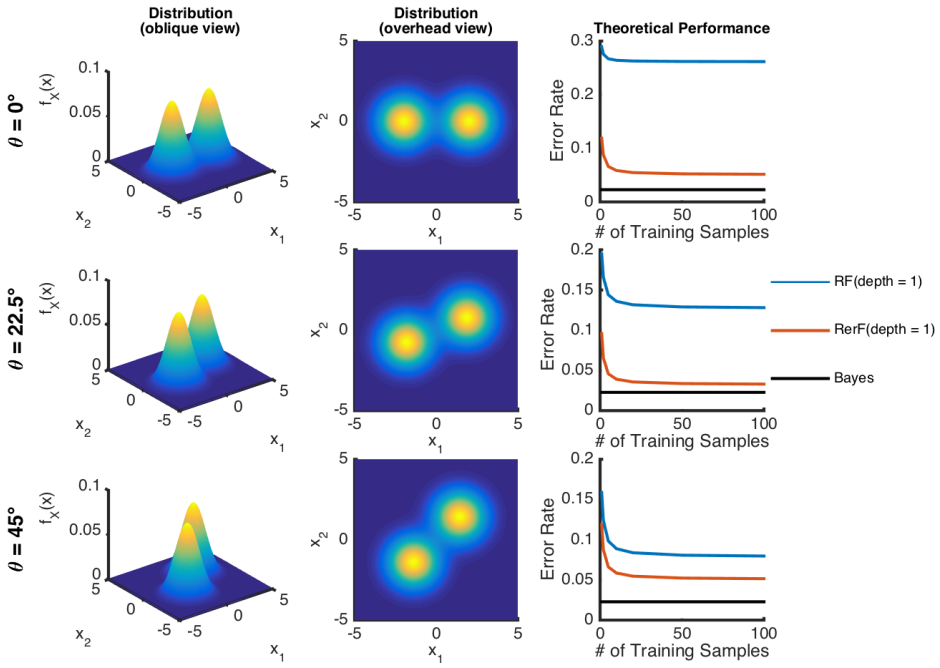


Figure 10: Theoretical performances of simplified RF and RerF models on a simple two-dimensional binary classification problem. In the top row, the two classes are distributed according to normal distributions having means that differ only in the first dimension and both having identity covariances. These distributions are shown in the left and middle panels. The right-most panel shows the theoretical error rate as a function of the number of training samples for RF and RerF. The Bayes optimal error rate is also shown for reference. The middle and bottom rows are the same as the top row, except the distributions have been rotated by 22.5° and 45° respectively. In all three cases, the RerF classifier outperforms RF for all training set sizes.

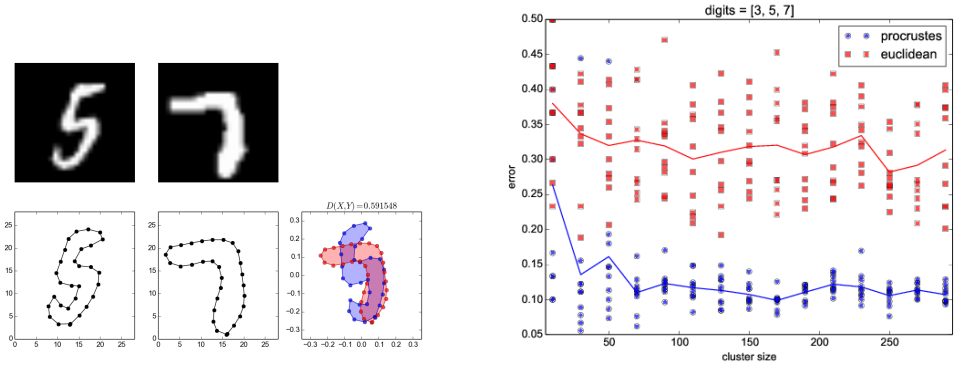
The R version of RerF is now functional and is an order of magnitude faster than the Matlab implementation. This version of RerF allows the user to specify the minimum size of a node and a parameter to tweak the rotation matrix. Additional basic functionality is being added to this tool including bagging, out-of-bag error reporting, max tree depth, and pruning.

2.6 Non-Parametric Shape Clustering

We developed a prototype algorithm for 2-dimensional shape clustering, which is invariant under affine transformations. This employs the procrustes distance between two objects, which requires feature extraction to obtain landmark points; see Fig. 12 for an example. We intend to apply a variant of this method to analyze neural synapses, since we have such datasets from our collaborators. However, to this particular dataset, the preprocessing techniques may be highly sophisticated, and a new metric to compare different objects may be necessary. Moreover, these shapes are 3-dimensional. We are currently working on this project with our collaborators, extending our existing techniques to this particular dataset.
We also started working on a related, and more general, project. We intend to develop non-parametric clustering algorithms with statistical guarantees. We will use an energy-statistics based approach. Given two datasets X and Y, there is an energy function $\mathcal{E}(X, Y)$ test statistic which allows us to infer if X and Y have the same distribution. Our results thus far suggest that this can be written as a quadratic optimization problem with quadratic constraints:

$$\max_{x,z\in\mathbb{R}^N} x^T \Delta z \qquad \text{s.t. } x_i^2 = 1, x + z = 0 \tag{1}$$

where Δ is a dissimilarity data matrix. There is not enough literature on this interesting problem, so this will very likely lead to new methods which can have interesting applications, in particular to neuroscience datasets.



(a) MNIST digits with extracted landmarks and alignment.

(b) Classification error against the size of each cluster (the three classes have the same number of points) is shown in blue. The red line is standard K-means with Euclidean distance for comparison.

Figure 11: MNIST handwritten digits and classification error results.

We have been mostly focused on developing non-parametric clustering methods. To this purpose, we are exploring ideas from energy statistics, which is non-parametric, robust, and rotational invariant, thus it incorporates the main ingredients that we are looking for. The main difficulty is to formulate an algorithm based on this, i.e. to identify the correct test statistic, or to formulate it as a feasible optimization problem. Consider K-Means clustering problem which is $\min_{\{\mathcal{C}_k\}} \sum_{k=1}^{K} \sum_{x \in \mathcal{C}_k} \|x - \mu_k\|^2$, where $\mathcal{C}_k$ is the kth cluster and μ_k the mean of its points. We showed that this problem is equivalent to

$$\max_{G} \text{Tr}\left(G^T K G\right) \qquad \text{s.t.} \quad G \geq 0, G^T G = I, G G^T e_1 = e_1. \tag{2}$$

where $e_1 = (1, 1, \ldots, 1)^T$. This is a Quadratically Constrained Quadratic Problem (QCQP), which is usually NP-hard. Analogously, consider the energy function $\mathcal{E}(F, G) = 2\mathbb{E}\|X - Y\| - \mathbb{E}\|X - X'\| - \mathbb{E}\|Y - Y'\|$ between $X, X' \sim F$ and $Y, Y' \sim G$. We showed that this can be written as $\mathcal{E}(A, B) = e_1^T \Delta e_1$, where Δ is a dissimilarity matrix between the two sets of data points $A \overset{iid}{\sim} F$ and $B \overset{iid}{\sim} G$. Consequently, a simple two-class clustering problem would be

$$\max_{x, z \in \mathbb{R}^N} x^T \Delta z \qquad \text{s.t. } x_i^2 = 1, x + z = 0, \tag{3}$$

which is also a QCQP problem. We are currently investigating this problem and trying to generalize it correctly for more classes. A simple check of the energy function as a test statistic is shown in Fig. 12. Under the null $F = G$, T converges to a quadratic form of normally distributed random variables. This seems to be the case in the first (blue) histogram, while it is definitely not the case in the other (red and green) histograms. For the blue histogram a single test gives $T \approx 0.32$ (small), for the red histogram $T \approx 4000$ (large), and for the green histogram $T \approx 105$ (large), with only a few points. Thus, energy statistics based approach is able to distinguish between different distributions, even when the clusters have the same mean, which is a property that K-Means cannot resolve.

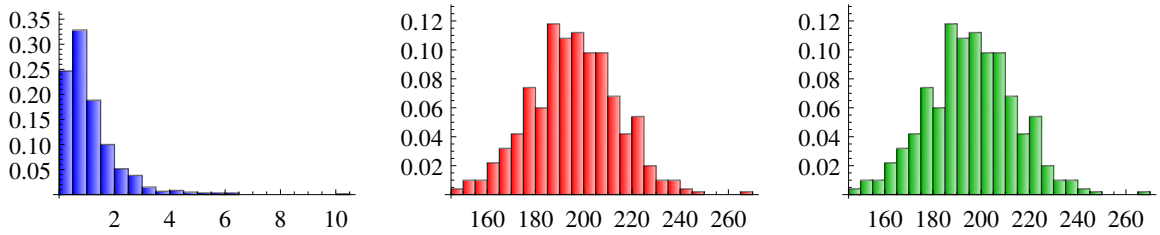


Figure 12: Distribution of test statistic $T \equiv \frac{nm}{n+m}\mathcal{E}(A, B)$ for an ensamble obtained from two distributions: $A \overset{iid}{\sim} \mathcal{N}(\mu_A, \sigma_A^2)$ and $B \overset{iid}{\sim} \mathcal{N}(\mu_B, \sigma_B^2)$, where $|A| = n$ and $|B| = m$. Blue histogram: $\mu_A = \mu_B = 0$ and $\sigma_A = \sigma_B = 1$; Red histogram: $\mu_A = -\mu_B = 1$ and $\sigma_A = \sigma_B = 1$; Green histogram: $\mu_A = \mu_B = 0$, $\sigma_A = 1$ and $\sigma_B = 1.5$.

Energy statistics provides a nonparametric test for equality of distributions. It is rotational invariant which is a highly desirable quality for clustering. For a two-class problem, $X, X' \sim \mu$ and $Y, Y' \sim \nu$, where μ, ν are CDFs, it reads

$$\mathcal{E}(X, Y) = 2\mathbb{E}\|X - Y\| - \mathbb{E}\|X - X'\| - \mathbb{E}\|Y - Y'\|. \tag{4}$$

We are developing a clustering framework based on (4). Our criteria is that $\mathcal{E}$ should be a maximum when data points are correctly classified. It is possible to show that there is a map from the data space of X, Y to the probability space of μ, ν which is a Hilbert space whose inner product can be obtained from a kernel function related to (4), $\langle \mu, \nu \rangle = k(x, y)$. This enables us to formulate our clustering problem as follows:

$$\max \left\{ \mathrm{Tr} L^{1/2} Z^T K Z L^{1/2} \right\} \quad \text{s.t.} \quad Z_{ij} \in \{0, 1\}, \sum_i Z_{ij} = N_j, \sum_j Z_{ij} = 1, Z^T Z = L^{-1} \tag{5}$$

where N_j is the number of elements in the jth cluster, $L^{-1} = \mathrm{diag}(N_1, \ldots, N_k)$, and K is the Gram matrix obtained from the kernel. Let $\mathcal{X}$ be the pooled data matrix. If we replace $K \to \mathcal{X}^T \mathcal{X}$ we recover the well-known k-means problem, which in this formulation is related to spectral clustering and normalized cuts. Problem (5) is NP-hard and a numerical implementation is prohibitive even for small data sets. We are investigating how to solve (5) in a feasible way. As an evidence that (5) is the correct optimization problem, and more importantly, it illustrates the power behind our proposal, in Fig. 13 we generate data and plot the objective in (5) versus n, where n is the number of points randomly shuffled from one class to the other. Therefore, for $n = 0$ the function must be a maximum. We do this for the kernel related to (4) (blue dots) and compare with the kernel related to k-means (red dots). Clearly, (5) based on (4) is able to distinguish between different cluster even for complex data sets that are not linearly separable. Moreover, in our formulation there are no free-parameters in the kernel.

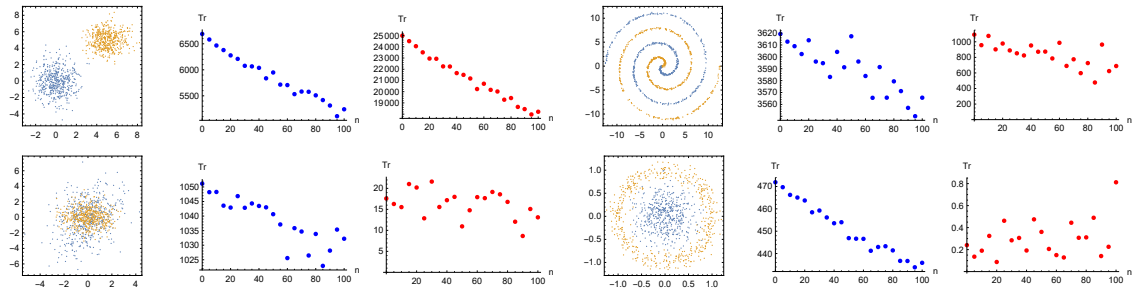


Figure 13: Two dimensional datasets and the objective function in (5) as a function of n, where n is the number of shuffled points from its correct class to the wrong class. Blue dots are for (4) and red dots for k-means. A good function must be monotonically decreasing. We can clearly see that (4) is way more powerful than k-means.

2.7 Discriminability

We develop a measure of discriminability (or reliability). It is intuitive to understand and easy to implement. Discriminability is defined to be the probability of within subject distances being smaller than the cross subject distances. If we let $x_{i,t}$ denote the t[th] trial of subject i and $\Delta(,)$ be the metric, the (population) discriminability D is:

$$D := P(\Delta(x_{i,t}, x_{i,t'}) \leq \Delta(x_{i,t}, x_{i',t''})).$$

Previously, we search for the optimal processing pipeline which has the maximal discriminability.

Currently, we are considering the discriminability from a different perspective. Specifically, we want to use the discriminability as an internal measure of the consistency in clustering. If we let $x_{i,t}$ denote the t[th] sample of cluster i and $\Delta(,)$ be the metric, the discriminability D is: $D := P(\Delta(x_{i,t}, x_{i,t'}) \leq \Delta(x_{i,t}, x_{i',t''}))$. Large discriminability implies the clusters are more consistent, that is we can better differentiate samples from different clusters.

We are also developing a clustering algorithm 1 which maximizes the discriminability. The algorithm will assign each sample i a cluster identity k_i such that the discriminability is maximized. The algorithm is similar to K-means, but we believe discriminability provides a more robust measure than within cluster distances which is maximized in K-means.

Discriminability Algorithms

Algorithm 1 Cluster samples through maximizing discriminability.

Require: Samples $\{\boldsymbol{x}_i\}$ and number of clusters K.
Ensure: Cluster identity $\{k_i\}$.

```
function DiscriminabilityClustering
    Initialize {k_i} randomly
    while not convergent do
        for i do
            for j in 1 : K do
                Set k_i = j
                ComputeDiscriminability({x_i},{k_i})
            Set k_i to the cluster with maximum discriminability
    Output {k_i}
```

Algorithm 2 Compute discriminability estimate $\hat{D}$.

Require: Test-retest samples $\{\boldsymbol{x}_{i,t}\}$.
Ensure: Sample discriminability $\hat{D}$.
  function ComputeDiscriminability
    for i, t, i', t' **do** ▷ compute pairwise distances
      $PD[i, t, i', t'] = \delta(\boldsymbol{x}_{i,t}, \boldsymbol{x}_{i',t'})$
    Dsum = 0;
    Count = 0;
    for $i = 1, ..., n$ **do**
      for $t = 1, ..., s$ **do**
        $d =$ across subject distances to $\boldsymbol{x}_{i,t}$
        for $t' = 1, ..., s$ and $t' \neq t$ **do**
          $\hat{D}_{i,t,t'} = \sum_j \mathbf{I}\{PD[i, t, i, t'] < d[j]\}/Length(d)$ ▷ compare within and across distances
          $Dsum = Dsum + \hat{D}_{i,t,t'}$
          $Count = Count + 1$
    $\hat{D} = Dsum/Count$ ▷ Sample Discrminability

Algorithm 3 The function returns a p-value for testing the null hypothesis that $D = 0.5$.

Require: Test-retest samples $\{\boldsymbol{x}_{i,t}\}$, the number of permutations np.
Ensure: The p-value $p \in [0, 1]$ for testing the hypothesis that $D = 0.5$.
  function OneSampleTest
    $\hat{D} = ComputeDiscriminability(\{\boldsymbol{x}_{i,t}\})$ ▷ compute true sample discriminability
    for $j = 1, ..., np$ **do**
      $\{\boldsymbol{x}^{(j)}_{i,t}\} = Permute(\{\boldsymbol{x}_{i,t}\})$ ▷ permute the subject labels
      $d[j] = ComputeDiscriminability(\{\boldsymbol{x}^{(j)}_{i,t}\})$ ▷ compute discriminability of permuted samples
    $p = \sum_j \mathbf{I}\{\hat{D} < d[j]\}/np$ ▷ compute p-value

Algorithm 4 The function returns a p-value for testing the null hypothesis that $D(\boldsymbol{\psi}_1) = D(\boldsymbol{\psi}_2)$.

Require: Raw data $\{f_\phi(\boldsymbol{v}_i)\}$, pipeline $\boldsymbol{\psi}_1$, pipeline $\boldsymbol{\psi}_2$, the number of bootstrapped samples nb.

Ensure: The p-value $p \in [0, 1]$ for testing the hypothesis that $D(\boldsymbol{\psi}_1) = D(\boldsymbol{\psi}_2)$.

function TwoSampleTest

$\{^1\boldsymbol{x}_{i,t}\} = g_{\boldsymbol{\psi}_1}(\{f_\phi(\boldsymbol{v}_i)\})$

$\{^2\boldsymbol{x}_{i,t}\} = g_{\boldsymbol{\psi}_1}(\{f_\phi(\boldsymbol{v}_i)\})$ ▷ process the raw data with two pipelines

$\hat{D}(\boldsymbol{\psi}_1) = ComputeDiscriminability(\{^1\boldsymbol{x}_{i,t}\})$

$\hat{D}(\boldsymbol{\psi}_2) = ComputeDiscriminability(\{^1\boldsymbol{x}_{i,t}\})$ ▷ compute sample discriminability estimates

for $j = 1, ..., nb$ **do**

for $i = 1, ..., n$ **do**

$i_1, i_2 = Sample(n)$ ▷ randomly select two subjects

$\lambda = SampleUniform$

for t = 1,...,s **do**

$^1\boldsymbol{x}_{i,t}^{(j)} = {}^1\boldsymbol{x}_{i_1,t}\lambda + {}^1\boldsymbol{x}_{i_2,t}(1-\lambda)$ ▷ Linearly combine two observations

$\hat{D}^{(j)}(\boldsymbol{\psi}_1) = ComputeDiscriminability(\{^1\boldsymbol{x}_{i,t}^{(j)}\})$

$\hat{D}^{(j)}(\boldsymbol{\psi}_2) = ComputeDiscriminability(\{^2\boldsymbol{x}_{i,t}^{(j)}\})$

$ind = 0$

for $j = 1, ..., nb$ **do** ▷ generate the null distribution

for $j' = i + 1, ..., nb$ **do**

$d[ind] = \hat{D}^{(j)}(\boldsymbol{\psi}_1) - \hat{D}^{(j')}(\boldsymbol{\psi}_1)$

$ind = ind + 1$

$d[ind] = \hat{D}^{(j)}(\boldsymbol{\psi}_2) - \hat{D}^{(j')}(\boldsymbol{\psi}_2)$

$ind = ind + 1$

$p = \sum_j \mathbf{I}\{\hat{D}(\boldsymbol{\psi}_1) - \hat{D}(\boldsymbol{\psi}_2) < d[j]\}/Length(d)$ ▷ compute p-value

2.8 Joint Embedding

We developed a method to jointly embed multiple graphs/networks. Previous spectral embedding techniques work on each graph separately. Our joint embedding approach generalizes Adjacency Spectral Embedding to multiple graphs. Specifically, the joint embedding method identifies a linear subspace spanned by rank one symmetric matrices and projects adjacency matrices of graphs into this subspace. Given m graphs $\{G_i\}_{i=1}^m$ with $\mathbf{A}_i$ being the corresponding adjacency matrix, the d-dimensional joint embedding of graphs $\{G_i\}_{i=1}^m$ is given by

$$(\hat{\mathbf{H}}, \hat{\mathbf{D}}_1, ..., \hat{\mathbf{D}}_m) = \underset{\mathbf{D}_i, \|h_k\|=1}{\operatorname{argmin}} \sum_{i=1}^{m} \|\mathbf{A}_i - \mathbf{H}\mathbf{D}_i\mathbf{H}^T\|^2$$
$$\text{subject to} \quad \mathbf{D}_i \text{ being diagonal.}$$

Here, h_k is the kth column of matrix $\mathbf{H}$. The $\hat{\mathbf{H}}$ are estimated latent positions for vertices, and the diagonal of $\hat{\mathbf{D}}_i$ can be treat as the feature of graph i. We performed theoretical and numerical analysis of the joint embedding. The code and paper can be found here.

We study predicting individual composite creativity index (CCI) through brain connectomes obtained by Multimodal Magnetic Resonance Imaging. In total, 113 healthy, young adult subjects were scanned and their CCI is assessed by independent judges using the Consensual Assessment Technique. First, we jointly embed brain graphs of all subjects. Figure 14 shows a typical graph and $\hat{h}_6\hat{h}_6^T$ estimated by the joint embedding. Next, we construct a linear regression model by treating the diagonal of $\hat{\mathbf{D}}_i$ as explanatory variables and CCI as the response variable.

Overall, the regression model for predicting CCI is significant at level 0.05 compared to the null model. We found CCI positively correlated to overall connectivity of the brain. Furthermore, we found CCI significantly negatively related to $\hat{h}_6\hat{h}_6^T$. This implies that compared to within hemisphere connectivity across hemisphere connectivity has a larger positive impact on human creativity.

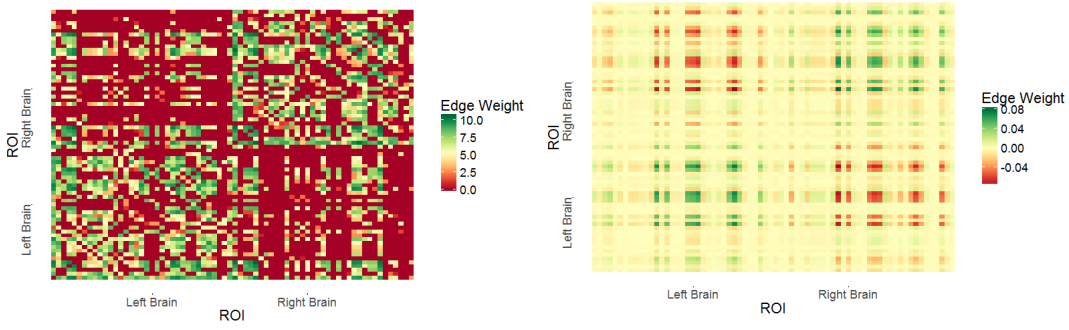


Figure 14: The left panel shows the graph derived from a typical subject. There is much more neural connectivity within each hemisphere. The right panel shows the rank one matrix $\hat{h}_6^T\hat{h}_6$, which has positive connectivity within each hemisphere, but negative connectivity across hemispheres.

2.9 Robust Law of Large Graphs

To estimate the mean of a collection of weighted graphs under a low rank random graph model (e.g. Stochastic Blockmodel) when observing contaminated graphs, we propose an estimator which not only inherits robustness from element-wise robust estimators but also has small variance due to application of a rank-reduction procedure. Under appropriate conditions, we prove that our estimator outperforms standard estimators via asymptotic relative efficiency. Previously we illustrated our theory and methods by Monte Carlo simulation. And now we focus on the real data experiment.

The real data we consider is a structural connectomic data. The graphs are based on diffusion tensor MR images. It contains 114 different brain scans, each of which was processed to yield an undirected, weighted graph with no self-loops, using the m2g/ndmg pipelines. The vertices of the graphs represent different regions in the brain defined according to an atlas. We used the desikan atlas with 70 vertices. The weight of an edge between two vertices represents the number of white-matter tract connecting the corresponding two regions of the brain. As we know, ndmg is a better pipeline compared to m2g, which means that the mean graph derived from ndmg should be a more accurate estimate to actual population mean graph. In order to evaluate the performance of the four estimators, we build estimates based on the samples from m2g, while using the sample mean graph from ndmg as an estimate of the probability matrix P. Specifically, each Monte Carlo replicate corresponds to sampling m graphs out of the 114 from the m2g dataset and computing the four estimates based on the m sampled graphs. We then compared these estimates to the sample mean for the 114 graphs from the ndmg dataset. We ran 100 simulations for the sample sizes $m = 2, 5, 10$. We also considered all possible dimensions for adjacency spectral embedding by ranging d from 1 to 70 in order to investigate the impact of the dimension selection procedures. We plot the result in figure 15

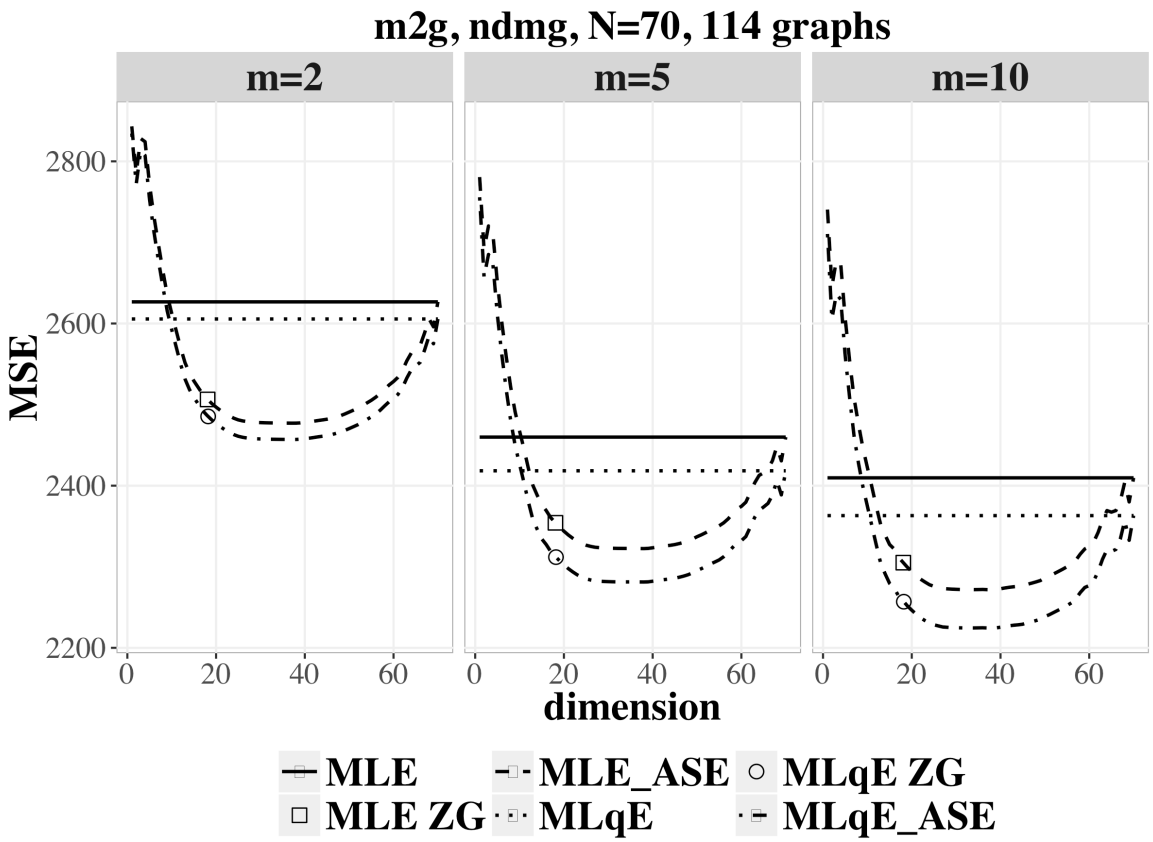


Figure 15: **Comparison of MSE of the four estimators for the desikan atlases at three sample sizes based on m2g and ndmg pipelines. 1. MLE (horizontal solid line) vs MLqE (horizontal dotted line):** MLqE outperforms MLE since robust estimators are always preferred in practice; **2. MLE (horizontal solid line) vs MLE_ASE (dashed line):** MLE_ASE wins the bias-variance tradeoff when embedded into a proper dimension; **3. MLqE (horizontal dotted line) vs MLqE_ASE (dashed dotted line):** MLqE_ASE wins the bias-variance tradeoff when embedded into a proper dimension; **4. MLqE_ASE (dashed dotted line) vs MLE_ASE (dashed line):** MLqE_ASE is better, since it inherits the robustness from MLqE. And the square and circle represent the dimensions selected by the Zhu and Ghodsi method. We can see it does a pretty good job. But more importantly, a wide range of dimensions could lead to an improvement.

We had theorems for the MLqE under the exponential distribution. Actually the results can be generalized to a broader class of distribution families, and even a different entry-wise robust estimator other than MLqE with the following conditions:

1. Let $A_{ij} \overset{ind}{\sim} (1-\epsilon) f_{P_{ij}} + \epsilon f_{C_{ij}}$, then $E[(A_{ij} - E[\hat{P}_{ij}^{(1)}])^k] \leq \text{const} \cdot k!$, where $\hat{P}^{(1)}$ is the entry-wise MLE as defined before; This is to ensure the that observations will not deviate from the expectation too far away, such that the concentration inequality can apply.
2. There exists $C_0(P_{ij}, \epsilon) > 0$ such that under the contaminated model with $C > C_0(P_{ij}, \epsilon)$,

$$\lim_{m\to\infty} \left| E[\hat{P}_{ij}] - P_{ij} \right| < \lim_{m\to\infty} \left| E[\hat{P}_{ij}^{(1)}] - P_{ij} \right|;$$

   It requires the contamination of the model to be large enough (a restriction on the distribution) and $\hat{P}$ to be robust enough with respect to the contamination (a condition on the estimator).
3. $\hat{P}_{ij} \leq \text{const} \cdot \hat{P}_{ij}^{(1)}$; (This might be generalized to with high probability later)
   Since we use the results of $\hat{P}^{(1)}$ to bound $\hat{P}^{(q)}$, the proof can apply directly with this condition for an arbitrary $\hat{P}$.
4. $\text{Var}(\hat{P}_{ij}) = O(m^{-1})$, where m is the number of observations.
   We will get exactly the same results under this condition. However, even if the variance of the new estimator is not of order $O(m^{-1})$, we will get similar results with a different term related to m.

Although we only present the results under exponential distributions, the results can be generalized to a broader class of distribution families, and even a different entry-wise robust estimator other than MLqE with the following conditions:

1. Let $A_{ij} \overset{ind}{\sim} (1-\epsilon)f_{P_{ij}} + \epsilon f_{C_{ij}}$, then $E[(A_{ij} - E[\hat{P}_{ij}^{(1)}])^k] \leq \text{const}^k \cdot k!$, where $\hat{P}^{(1)}$ is the entry-wise MLE as defined before;
   This is to ensure that observations will not deviate from the expectation too far away, so that the concentration inequalities hold.
2. There exists $C_0(P_{ij}, \epsilon) > 0$ such that under the contaminated model with $C > C_0(P_{ij}, \epsilon)$,

$$\lim_{m\to\infty} \left| E[\hat{P}_{ij}] - P_{ij} \right| < \lim_{m\to\infty} \left| E[\hat{P}_{ij}^{(1)}] - P_{ij} \right|;$$

   It requires the contamination of the model to be large enough (a restriction on the distribution) and $\hat{P}$ to be robust enough with respect to the contamination (a condition on the estimator).
3. $\hat{P}_{ij} \leq \text{const} \cdot \hat{P}_{ij}^{(1)}$;
   Since we use the results of $\hat{P}^{(1)}$ to bound $\hat{P}^{(q)}$, the proof can apply directly with this condition for an arbitrary $\hat{P}$.
4. $\text{Var}(\hat{P}_{ij}) = O(m^{-1})$, where m is the number of observations.
   We will get exactly the same results based on this order. However, even if the variance of the new estimator is not of order $O(m^{-1})$, we will get similar results with a different term related to m.

Previously we consider the model to be based on exponential distribution, which is continuous and monotone. Now we consider Poisson distribution instead. Poisson distribution is a commonly used distribution for nonnegative graphs with integer values. And we will prove that it satisfies the conditions for generalization and as a result all theories apply directly.

Let $A_{ij} \overset{ind}{\sim} (1-\epsilon)f_{P_{ij}} + \epsilon f_{C_{ij}}$ with f to be Poisson, then we proved that $E[(A_{ij} - E[\hat{P}_{ij}^{(1)}])^k] \leq \text{const}^k \cdot k!$, where $\hat{P}^{(1)}$ is the entry-wise MLE as defined before. So Condition 1 is satisfied. Intuitively, since exponential distribution has a fatter tail compare to Poisson, we should have the bound for central moment of Poisson directly from the results for exponential distribution. Condition 2 is satisfied as long as the contamination is large enough while keep using the robust MLqE. For Condition 3, the extreme case happens when there are m data $x_1, \cdots, x_m$ with $0 \leq x_1 = \cdots = x_k \leq \bar{x} \leq x_{k+1} = \cdots = x_m \leq m\bar{x}/(m-k)$. In order to have ML$q$E larger than MLE $\bar{x}$, we need the weights of the first m data to be smaller than the weights of the rest $m-k$ data. So $e^{-\bar{x}} < \bar{x}^{x_m} e^{-\bar{x}}/x_m!$. Then $x_m! < \bar{x}^{x_m}$. By the lower bound in Stirling's formula, we have $x_m < e\bar{x}$ when $x_m > 0$. Note that if $x_m = 0$ then MLE equals MLqE since all data equals zero. Thus MLqE is bounded by $e\bar{x}$. As a result, $\hat{P}_{ij} \leq e\hat{P}_{ij}^{(1)}$ and Condition 3 is satisfied. At last, Condition 4 follows directly from theory of minimum contrast estimators.

So for all the theorems proved before, we can replace the exponential distribution by Poisson distribution and all the results still hold.

3 Scalable Algorithm Implementations

3.1 FlashX

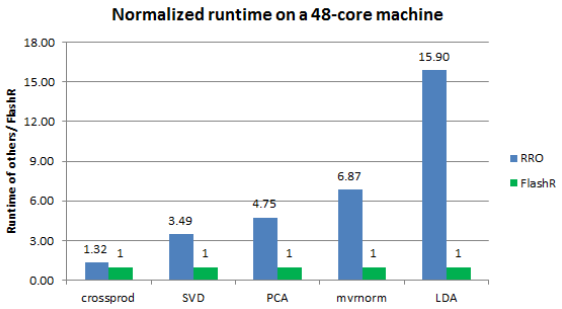


Figure 16: Normalized runtime of FlashR vs. Revolution R when executing R implementations of machine learning primitives on a dataset with one million data points and 1000 features on a large parallel machine with 48 CPU cores. FlashR outperforms Revolution R in all computations. When the computation gets more complex, the speed advantage of FlashR over Revolution R gets larger.

After having efficient matrix operations in the past months, including sparse matrix multiplication and various dense matrix operations, we integrate all matrix operations in a single computation framework called FlashR, which provides both high compatibility with R and efficiency. FlashR now overrides about 70 R matrix functions in the R base package. As such, we can run existing R code with little modification or no modification at all. For example, we ported a few R implementations of machine learning algorithms in the MASS package with little modification. We compare the speed of FlashR against Revolution R, which is also designed to parallelize and accelerate R code, on a large parallel machine with 48 CPU cores (Figure 16). Even for the simple matrix operation such as crossprod, FlashR outperforms Revolution R. As the computation gets more complex, the advantage of FlashR over Revolution R gets larger. When executing the LDA implementation (Linear Discriminant Analysis) in the MASS package, FlashR outperforms Revolution R by over an order of magnitude.

We use FlashR to process the billion-scale datasets to demonstrate its scalability (Table 1). We use three datasets here: (i) the Criteo dataset has over four billion data points with binary labels (click vs. no-click), used for advertisement click prediction; (ii) PageGraph is the adjacency matrix of a graph, which has 3.5 billion vertices and 128 billion edges; (iii) PageGraph-32ev are 32 singular vectors that we computed on the largest connected component of Pagegraph with the tools we built previously. In these experiments, we run the iterative algorithms (Logistic regression, k-means and PageRank) on the datasets until they converge.

Table 1: The runtime and memory consumption of FlashR on the billion-scale datasets on the 48 CPU core machine. The runtime of iterative algorithms is measured when the algorithms converge. We run PageRank on the PageGraph dataset, run k-means on PageGraph-32ev and the remaining algorithms on Criteo.

	Runtime (s)	Memory (GB)
Correlation	91.23	1.5
PCA	136.71	1.5
NaiveBayes	76.55	3
LDA	2280	8
Logistic regression	4154.40	26
k-means	1110.82	28
PageRank	3900	135

Even though we process the billion-scale datasets in a single machine (with 48 CPU cores), none of the algorithms are prohibitively expensive. Simple algorithms, such as Naive Bayes and PCA, require one or two passes over the datasets and take only one or two minutes to complete. Logistic regression and k-means take about $10-20$ iterations to converge. Because the PageRank implementation uses the power method, it takes 100 iterations to converge. Nevertheless, all of the iterative algorithms take about one hour or less.

FlashR scales to datasets with billions of data points easily when running out of core. Most of the algorithms have negligible memory consumption. PageRank consumes more memory because the sparse matrix multiplication in PageRank keeps vectors in memory for semi-external memory computation. The scalability of FlashR is mainly bound by the capacity of SSDs. The functional programming interface generates a new matrix in each matrix operation, which potentially leads to high memory consumption. Thanks to lazy evaluation and virtual matrices, FlashR only needs to materialize the small matrices to effectively reduce memory consumption.

We advance FlashR for exploratory analysis on a billion-scale graph. This includes functions for filtering data points and plotting histograms and heatmaps on billions of data points. Figure 17 shows an example of plotting the in-degree histogram of vertices in the page graph and a heatmap for the distribution of the vertices of the page graph in a two-dimension space.

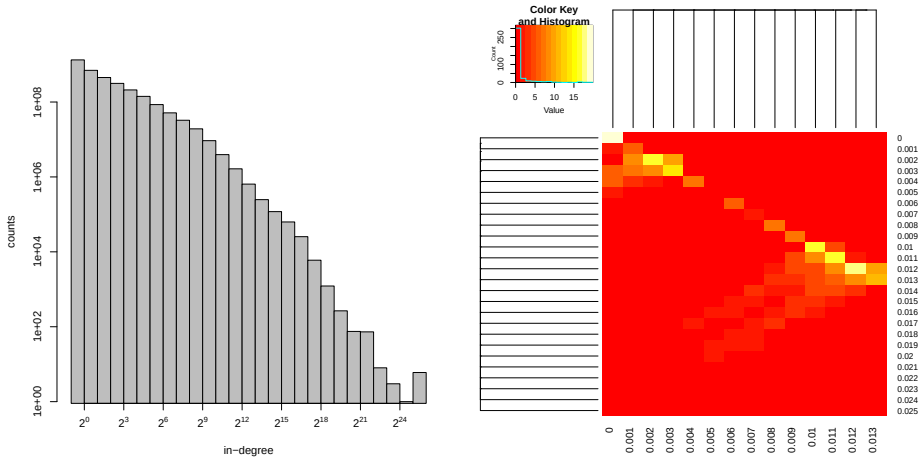


Figure 17: Left: the in-degree distribution of the Page graph. Right: a heatmap of the distribution of the vertices of the page graph in a two-dimension space; the coordinates of the vertices in the two-dimension space is determined by the first left and right singular vectors.

3.2 ndstore

We continue to now migrate all our annotation datasets to the cloud. The annotation projects will now be hosted in MySQL backed with AWS EBS storage and will be converted to AWS DynamoDB soon. There will be more than 30 odd annotation datasets avaliable in the cloud for public use. The datasets can be accessed at http://neurodata.io.
We added support for new resource and authentication web-servics in our python wrapper called ndio. This will allow our users to continue to use ndio for future interactions with the web back-end. They can now utilize the new web-services we have added and create resources using this which is more easier then using the RESTful web-services directly.
We are also in the process of deploying a status page for our web-services. The status page will act as a dashboard for all our users and a single point of contact for them to check if our services are online or suffering from an outage. It will also allow us to inform all our users who are subsribed to the status page of future planned outages. This will streamline our services and is standard industry practice for other companies running web-services. The status page is located here https://neurodata.statuspage.io/. The MRI ingest service is now active and being used by some users in the lab to ingest data into ndstore. All of this service was already deployed in the cloud and the ingested data will be avaliable at http://mri.neurodata.io.
Autoingest is a ndstore service for inserting image and annotation data into the data-store. The user posts information aboout the data inculding location, co-ordinates and datatype. The server uses this information and with a pull model ingests the raw image or annotation data from slices into cuboids and inserts them in the database. This service was earlier operational on local hardware deployed at Johns Hopkins. We modified this service so that it could now be run on Amazon Web Services and S3 object store. Raw data can now be staged on a publicly accessible web server or in a S3 bucket provided by us. The data can be now be inserted into MySQL as well as AWS S3 object store.
Propagate is a ndstore service for building scaling levels over base resolution of data. This is efficient for serving data at lower resolutions for processing and visualization. There also exist auto-zoom in and out capabilties to materialize the data at higher or lower resolutions on the fly if these scaling levels are not built or being built. This service was modified so that scaling levels could be built on the data inserted into AWS S3 object store.
In addition, we also added neariso scaling levels in addition to the exisitng scaling hierarchy. This reduces the data transfer size for 3D viewers and is the preferred interface for tools which use ndstore such as BigDataViewer and Neuroglancer. This will support easier insertion and visualization of data from collaborators.

3.2.1 ndingest

The access policies for the ndingest is now complete. We are currrently testing our the new service and will soon release it in beta mode to some our close collaborators. This service will allow us to speedup our data ingest rates manifold. We plan to benchmark this service once we are done deploying it. The ingest client developed witj JHU-APL, has now been converted so that it can use multiple threads in python. This capability will allow us to upload multiple slices of the data simultaneously and allow us to upload data to the cloud at a much faster rate.

3.3 ndviz

The vast majority of current high resolution imaging techniques in neuroscience are either 3-dimensional or contain a 3-dimensional component (e.g. 3-d data over time). Visualization of data produced by said techniques has traditionally been confined to the three canonical planes; xy, yz, and xz. However, advancements in Web graphics rendering have made dynamic, 3-dimensional visualization possible in modern Web browsers. Using WebGL, code running in a users Web browser can access the end users Graphics Processing Unit (GPU), taking advantage of specialized graphics hardware present in most modern desktops, laptops, and even smart phones.

Using WebGL for dynamic rendering improves both the performance and capability of a graphical Web application. To that end, we have integrated Neuroglancer, a Web visualization tool built for 3-dimensional data, into NeuroDataViz. Neuroglancer has provided us with a baseline 3-dimensional rendering tool, which we can use for specialized features (e.g. dynamic false coloring). By building on a common framework, we can contribute features back to the neuroscience community as well as take advantage of new features developed by our collaborators (or even other neuroscience users).

With this new version of NeuroDataViz, we can visualize all of our existing data in the three canonical planes mode. We are now working to build 3-dimensional meshes, both for annotated Electron Microscopy data and for thresholded Light Microscopy data. A sample of EM meshes is available in the figure below. We are now developing tools for automatically generating 3-d meshes (shapes) for both datatypes on-demand as the user makes a selection in the 3-d view.

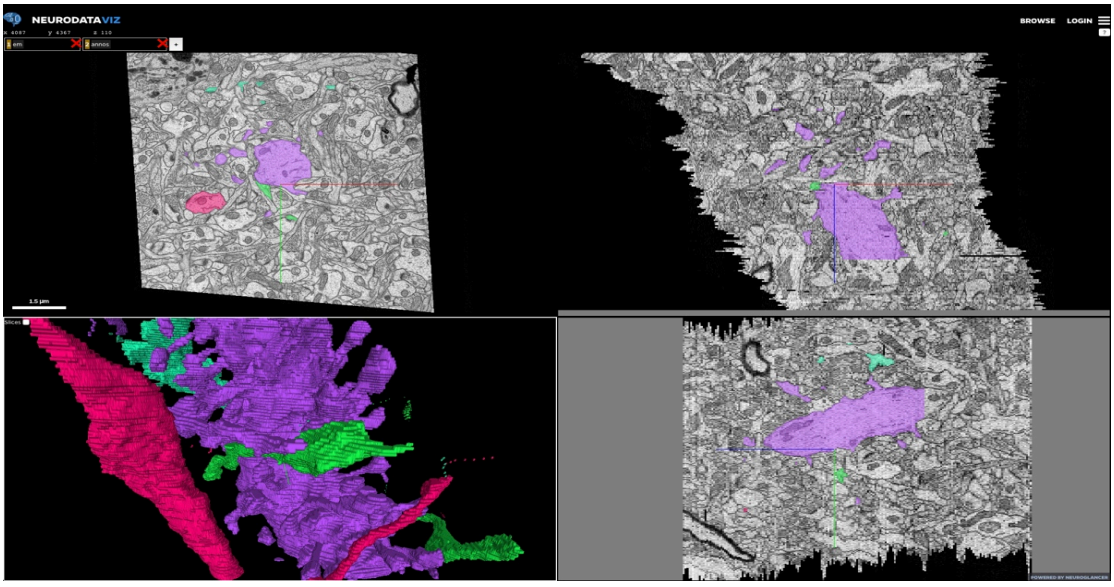


Figure 18: NeuroDataViz, powered by Neuroglancer with pre-computed meshes displayed in 3-d. Data from Harris et al. "A resource from 3D electron microscopy of hippocampal neuropil for user training and tool development," Nature Scientific Data 2015.

3.4 knor: K-means NUMA Optimized Routines

The **knor** library is a set of tools enabling users to perform the popular k-means algorithm at scale at speeds of 10x-100x time of that of popular frameworks in use today like Spark's MLlib, H^2O and GraphLab(Turi, Dato). We provide highly optimized routines for the following cases:

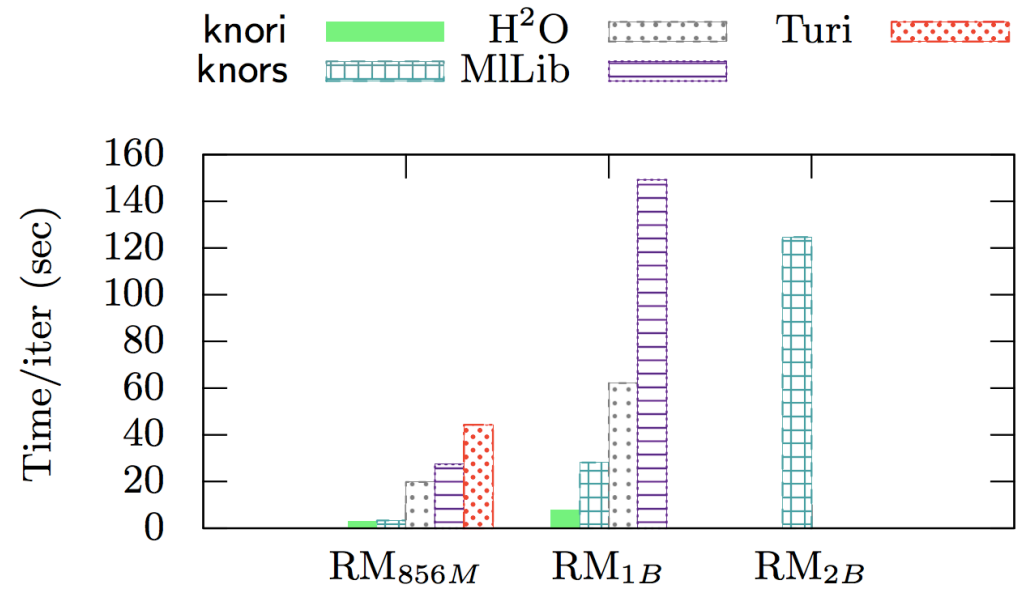
- Big-data that fits into the main memory (RAM) of a single machine, use **knori**.
- Big-data that fits into the aggregate main memory (RAM) of a cluster of distributed machines in the cloud, use **knord**.
- Big-data that cannot fit into the main memory (RAM) of a single machine, but can be placed out-of-corek, on disk, use **knors**.

As part of efforts to make **knor** as user-friendly as possible we now support it on any platform via Docker containerization. We provide a one line installation procedure found at The repo's landing page. We further provide detailed instructions on how to use and extract data within the **knor** framework.

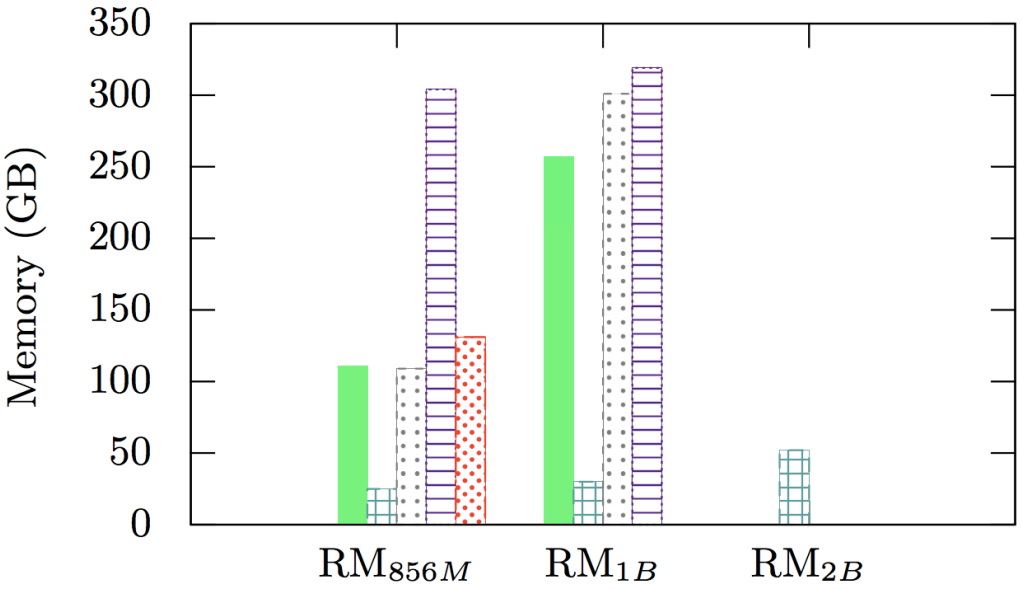
We recently submitted the **knor** paper to High-Performance Parallel and Distributed Computing (HPDC 2017) and publicly released it to arxiv.
Figure 19 compares the performance of our in-memory (**knori**) and our semi-external memory (**knors**) routines to our competitors. Distributed results (**knord**) can be found within the paper.

We previously described the **knor** library. As a set of tools enabling users to perform the popular k-means algorithm at scale at speeds of 10x-100x time of that of popular frameworks in use today like Spark's MLlib, H^2O and GraphLab(Turi, Dato).

We would like to report that our **knor** paper that was submitted to the ACM Symposium on High-Performance Parallel and Distributed Computing (HPDC 2017) and publicly released it to arxiv was **accepted** for publication to the 26th proceedings of HPDC.



(a) Per iteration time elapsed of each routine.



(b) Memory consumption of each routine.

Figure 19: Speed and Memory comparison on randomly generated datasets (i) RM_{856M}, a 856 Million X 16 dataset of size 103GB and (ii) RM_{1B}, a 1 Billion X 32 dataset of size 251 GB and (iii) RM_{2B}, a 2 Billion X 64 dataset of size 1.1 TB. Turi is unable to run on RM_{1B} on our machine and only SEM routines are able to run on RU_{2B} on our machine with 48 Cores and 1 TB of RAM.

3.5 ndreg

We received three new CLARITY image volumes from our colleagues at Stanford University. Each dataset contained two channels of a single mouse brain hemisphere at a 585 μm × 585 μm × 5000 μm resolution. The images were ingested and propagated to lower resolutions using NeuroData infrastructure. NeuroData's registration module (ndreg) was then used to register each image to the Allen institute's mouse Reference Atlas (ARA).
First each image was reoriented to the ARA, the background was subtracted and a mask was generated to eliminate bright regions. After affine alignment, each Stanford image was deformably aligned to the ARA through Large Deformation Diffeomorphic Metric Mapping (LDDMM). Since the ARA and CLARITY images differed greatly in intensity profile, mutual information matching was adopted during this step. Alignment was done in a 3-step multi-resolution approach, with registration at coarser scales initializing the alignment at subsequent finer scales. It was clear from the ARA-CLARITY checkerboard composite images that registration proceeded successfully (Figure 20).

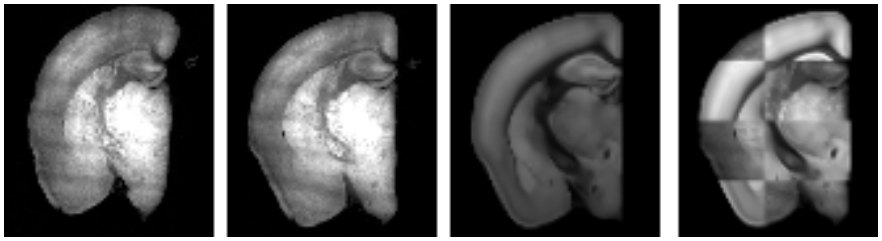
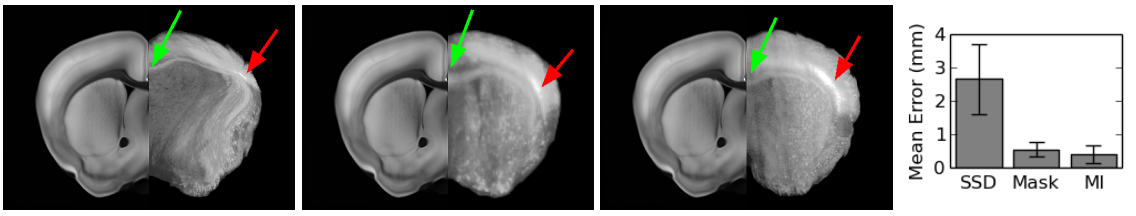


Figure 20: Coronal slices from image volumes. From left to right: CLARITY before LDDMM, CLARITY after LDDMM, ARA, checkerboard composite of ARA and CLARITY after LDDMM.

The NeuroData Registration python module, ndreg, uses Large Deformation Diffeomorphic Metric Mapping (LDDMM) to register a template image I_0 to a target image J_1. It does this by finding smooth invertible map φ such that the matching term $M(I_0 \circ \varphi^{-1}, J_1)$, a function whose value is small when $I_0 \circ \varphi^{-1}$ is aligned with J_1, is minimized. We compared MI-LDDMM, LDDMM with a Mutual Information based matching term, to our previous SSD-LDDMM and Mask-LDDMM techniques. SSD-LDDMM uses a Sum of Squared Differences matching term. As it is based on image subtraction, it assumes that bright regions in I_0 align with bright regions in J_1. Mask-LDDMM is SSD-LDDMM in which I_0 and J_1 are replaced with their respective binary brain mask images M_0 and M_1. Since the masks only contain information on which voxels are inside the brain, only edge information is used by Mask-LDDMM.
We placed fiducial landmarks in the corpus callosum and midbrain of four CLARITY template image and the Allen Reference Atlas (ARA) target. After registration corresponding CLARITY landmarks should be close to those of the ARA. If not then the registration was inadequate. Figure 21 shows that MI-LDDMM registration performed better than the previous methods.



(a) SSD-LDDMM (b) Mask-LDDMM (c) MI-LDDMM (d) Landmark Error

Figure 21: Comparison of SSD-LDDMM (a), Mask-LDDMM (b) and MI-LDDMM (c) registration for a CLARITY mouse brain. Panes (a-c) have an ARA coronal slice on the left juxtaposed to the corresponding aligned CLARITY slice on the right. Green arrows point out that the corpus callosum is misaligned by SSD-LDDMM but aligned correctly by MI matching. Red arrows show that SSD-LDDMM distorts bright regions. Fiducial landmarks were placed throughout the corpus callosum, and midbrain of the acquired volumes. Pane (d) compares mean errors between the deformed CLARITY and ARA landmarks after registration.

We received a rat brain image acquired using iDISCO microscopy from colleagues at the Johns Hopkins School of Medicine. iDISCO is a technique for clearing tissues that enables interrogatoion by light-sheet microscopy. Thus like CLARITY, iDISCO cleared brains can be imaged at high spatial resolution without physical slicing. The iDISCO volume was ingested into the NeuroData store at its full resolution of 5 μm isotropic. We also ingested the widely used Waxholm Rat Atlas, a T2 magnetic resonance image with corresponding labels at a 39 μm isotropic.

Large Deformation Diffeomorphic Metric Mapping (LDDMM) is a deformable image registration (alignment) algorithm which computes smooth invertible transforms between images. Using the NeuroData Registration (ndreg) python module we were able to perform a iDISCO to T2 MRI alignment by LDDMM under mutual information matching. Figure 22 shows the Waxholm Rat Atlas labels overlaid on the iDISCO image.

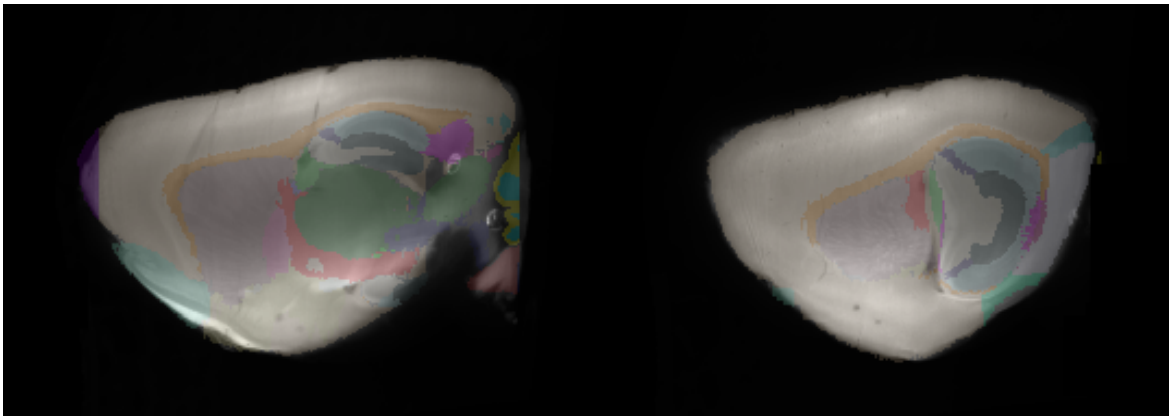


Figure 22: Waxholm rat atlas labels overlaid on sagittal slices of iDISCO rat brain

4 Scientific Pipelines: Infrastructure & Dataset Specific Progress

4.1 Science in the Cloud (SIC)

Science in the cloud (SIC) is an extensibility-focused scientific framework which addresses universal challenges in the computational sciences. Currently, re-performing and extending published analyses whether through new data or code is often unbearably difficult; (i) data may be closed-access; (ii) data may be organized in an ad hoc fashion; (iii) the code may be closed-source or undocumented; (iv) code may have been run with undocumented parameters and dependencies; (v) analyses may have run with specifically hardware compiled code. These properties make validating and extending scientific claims challenging.

We propose a solution to these gaps in the form of a framework which leverages publicly documented and deployable cloud instances with specific pipelines installed and configured to extend published findings; an implementation we simply term "science in the cloud," or, SIC (Latin for "thus was it written"). To address data access, we put data in the cloud. To address data organization, we utilize recently proposed data standards. To address closed source and undocumented, we generate open-source code and interactive demonstrations. To address software and hardware dependencies, we utilize virtualization, automated deployment, and cloud computing. SIC puts these pieces together to create a computing instance launched in the cloud designed for not only producing reproducible research, but enabling easily accessible and extensible science for everyone. SIC is designed to minimize the bottlenecks between publication and novel discoveries; leveraging the experience of the community, we propose a solution for transitioning to a universal, and "future-proof," deployment of software to the cloud.

A live demonstration is available at http://scienceinthe.cloud, which illustrates this framework. There are six key components which much be considered in sic: data storage, data organization, interactive demos, virtualization, computing, and deployment. The selection made for each of these components will have a significant impact on available selections for the others. The final product will be a highly interdependent network of tools and data.

Through the use of Amazon's cluster-computing engine, AWS Batch, the sic use case now supports scalable deployment in the Amazon cloud natively. Additionally, the Science in the cloud (SIC) manuscript has been accepted for publication at GigaScience.

SIC has been released in Gigascience and on GigaDB, Gigascience's open-access database for publication-based materials. The landing page for SIC has been further developed, and as it is hosted on Github directly, will serve as a permanent pointer to the paper, science, and our other related materials. The url for the landing page is: `http://scienceinthe.cloud`

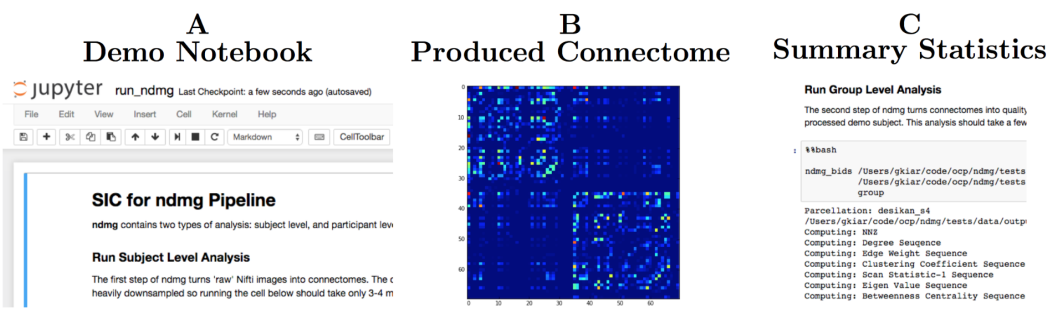


Figure 23: States of the demo notebook in the cloud. A) A Jupyter notebook displaying descriptions and code snippets to be run for both connectome estimation and summary statistic computation. B) After running connectome generation an adjacency matrix will appear to provide a visualization. C) Summary statistic computation calculates several graph features and plots them in a multipanel figure.

Table 1: There are six key components which must be selected for SIC. **Bold** indicates the selections made here, with their positive and negative qualities compared to some alternatives.

Hurdles	Available Tools	Pros of Selection	Cons of Selection
1) Data Storage	**S3** [8], Dropbox [9], Google Drive [10]	API, pay-by-usage	requires user familiarity with Amazon tools
2) Data Organization	**BIDS** [11], NWB [12], MINC [13]	documented, validator, active community	new, not yet fully adopted
3) Interactive demo's	**Jupyter** [14], R Notebook [15], Shiny [16]	versatile, accessible	optimized for Python
4) Virtualization	**Docker** [17], Virtualbox [18], VMware [19]	lightweight, self-documented	--
5) Deployment	**manual**, ECS [20], Kubernetes [21], MyBinder [22], CBRAIN [23]	no additional dependencies	does not scale effectively
6) Computing	**EC2** [20], Google Compute Engine [24], Microsoft Azure [25]	scalable, flexible	requires technological expertise

Figure 24: Excerpted table from [3].

4.2 ndmg

In an effort to further verify that derivatives produced by the ndmg pipeline are high-quality and execution of a given dataset within the pipeline was successful, we have expanded upon a automatically generated set of quality control figures. In particular, we have developed the first, to our knowledge, connectome-specific graph quality control plot. This figure, shown as the "Degree" panel in Figure 25, considers the intra- and inter- hemispheric connectivity of the graph, and plots the degree of each node for both same and across hemispheric connectivity. Many real-world graphs contain node and edge attributes, but often location is not among them (or it is used to define the edges); in connectomics, location is an important feature of each node and plays a role in its connectivity patterns, as in whether a connection will exist within or across hemispheres of the brain. We also are computing the mean connectome for the given dataset in this summary figure.

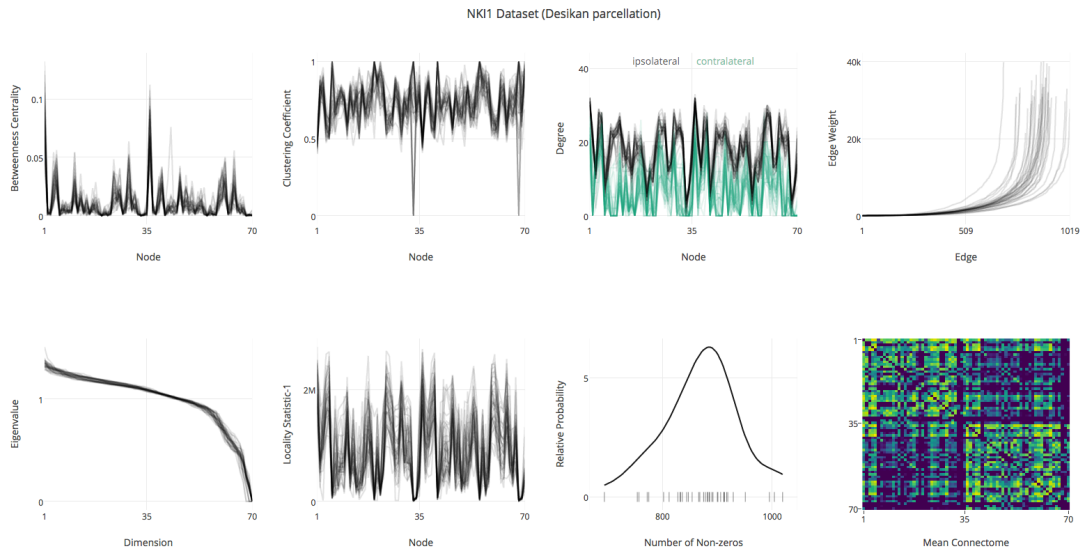


Figure 25: Quality Control of Connectomes

Through the use of AWS Batch, a cloud deployment script has been added to the pipeline which enables users to run the pipeline in the cloud directly on data also stored in the cloud. This additional feature significantly lowers the barrier to entry for use of the pipeline, and enables researchers to process and store their data without requiring physical compute or data storage hardware or management expertise in house. Figure **??** shows an example workflow that could be used by researchers performing a study.

Through use of the recently-developed cloud deployment options in ndmg, a collection of 3,000 human brain scans was processed in a single day for under $1,000 using our pipeline. The results are all publicly available online at our website, http://m2g.io, and we have continued work on building a paper documenting our pipeline around this tool and these exciting results. One such result, as shown in Figure **??**, illustrates the mean connectomes from a variety of datasets, and computes a multi-center mean and standard deviation connectome, as well. This figure both illustrates the consistency of our pipeline across a wide range of data, and sheds light on properties about the structure of the brain. Exploring these figures further, we can identify edges which are most highly variable in the brain, and focus studies trying to understand the properties of the related regions lend themselves to such high variability as opposed to other edges with lower variability.

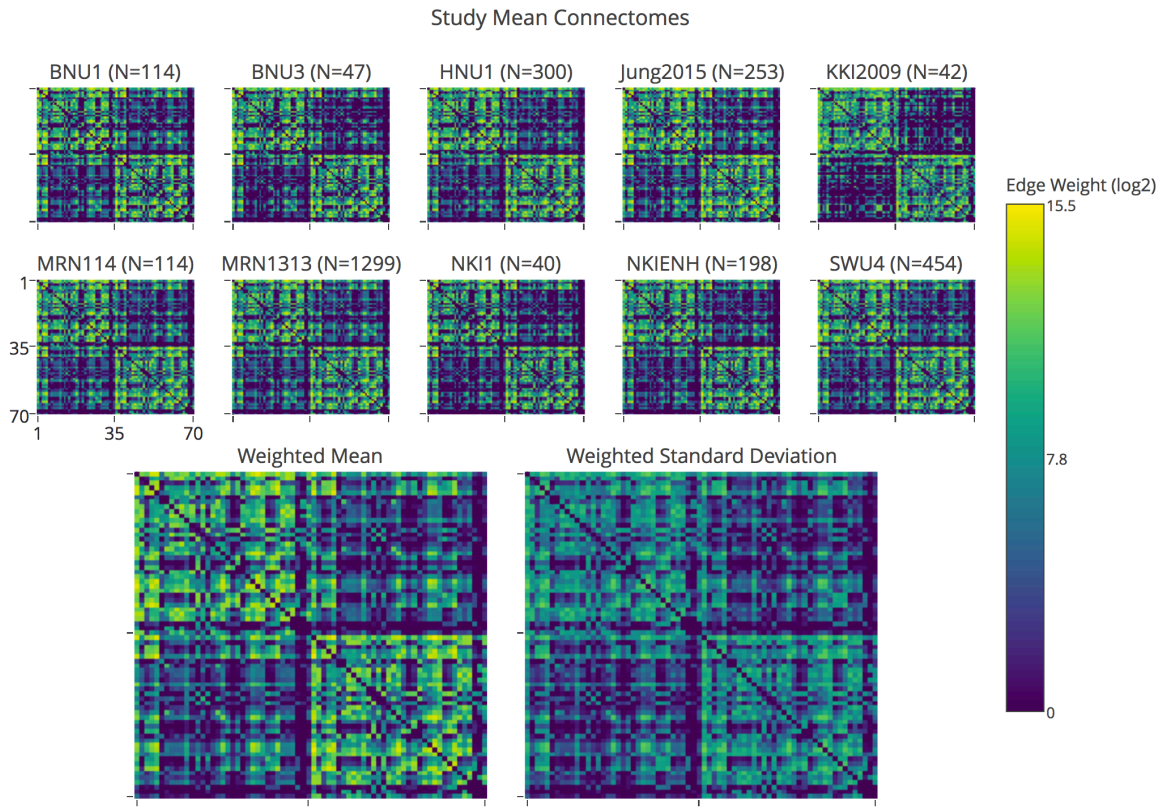


Figure 26: **Multi-Study Mean Connectomes**. Looking at 70-nodes graphs built upon the Desikan parcellation, we have computed the mean connectome from a variety of processed datasets. We then computed a weighted mean and standard deviation of the mean connectomes to produce the largest known population-level connectome to-date, consisting of 2861 sessions. As expected, ipsi-lateral connectivity is consistently more dense than contra-lateral connectivity. Similarly, the standard deviation connectome, which highlights edges that are more highly variable, shows higher ipsi-lateral density. This suggests that not only are same-hemisphere connections more likely to occur, but they have a higher variance than across-hemisphere connections, as well.

4.3 CLARITY

4.3.1 A low-latency pipeline for processing CLARITY data in the cloud

We are working on migrating our exisiting CLARITY pipeline to run entirely on virtualized infrastructure in the cloud. This workflow includes ingesting into ndstore, aligning to a reference atlas, and storing a registered stack back into ndstore. Currently, ndstore is working the cloud, with manual ingest of data. The next steps are to deploy LDDMM via Docker containers to run within the cloud environment.

5 Reference Datasets

References