SIMPLEX Monthly Report

October, 2017

Johns Hopkins University, labs of Joshua T. Vogelstein and Randal Burns

ndstore

We have now added a resource RESTful service which allows users to create, list and delete project management information. This service is very useful for power users to manage their projects without having to log into the management console.

We continue to improve our caching layer in the cloud. We have now added a cache manager which manages the Redis cache for ndstore. The manager monitors the cache and starts evicting data when the data in cache exceeds a pre-set limit. This allows for efficient management of Redis memory and data can be moved transparently. We also implemented a Readers-Writer Lock using Redis atomic services. This lock can be now used by different processes using the redis-client. We plan to work towards trying to add this lock into the open-source redis-py client.

ndingest

We have now built wrappers for AWS lambda, SQS, S3 and DynamoDB in \textit{*ndingest*}. The next stage is to write AWS security policies and ARN roles so that different aspects of these services can communicate with each other in the cloud. We are also developing scripts to deploy these services or update configurations from the command line. In addition, we have also developed a migration script to migrate our existing data to the cloud. This is essential since textit{*ndingest*} is designed to ingest raw data not data already in ndstore.

We have also developed an \textit{*ingest-client*} with JHU-APL to allow users to be able to use this service. The \textit{*ingest-client*} is designed to be modular allowing users to add their respective plugins which match their data naming conventions. This is an improvement over the earlier phase where users had to convert their data to match a standard naming convention. This tool is also complemented with an ingest service which allows users to create, join, track and delete ingest jobs.

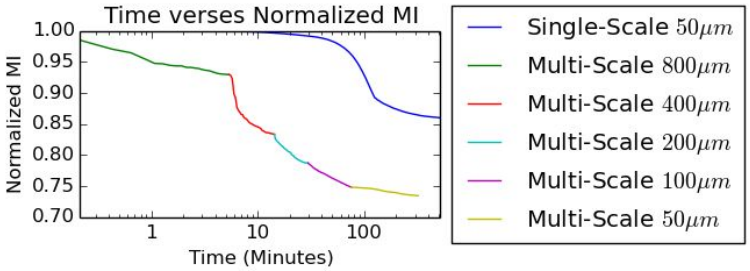
ndramondb

New Webservices that return JSON formatted object describing metadata were implemented for all RAMON metadata queries. The will be preferred to the existing interfaces that return HDF5 objects for Web applications. JSON is a simple text serialization format that is widely supported. It removes the HDF5 software dependency to access neuroscience metadata. HDF5, while powerful, is a cumbersome dependency in many operating systems and frameworks.

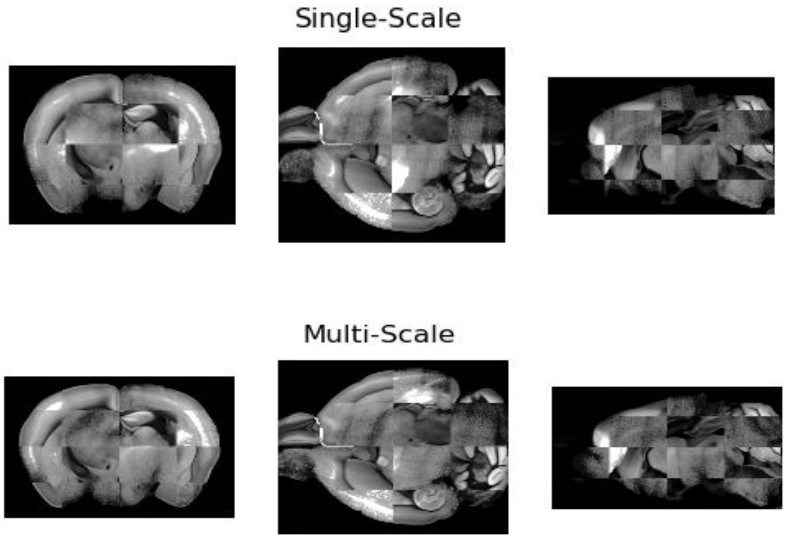
The JSON Web services were specifically designed for the ndviz visulation tool, which is built on Javascript. Ndviz now supports clickable metadata for any annotated location, i.e. a marked region in an annotation project. Clicking an annotation brings up the type of annotation (neuron, axon, dendrite, synapse) properties of the annotation (weight, confidence), and other metadata (author, creation date).

ndreg

NeuroData's registration module (ndreg) uses the Large Deformation Diffeomorphic Metric Mapping (LDDMM) for image alignment. LDDMM computes a smooth invertible mapping between template image I_0 and target image I_1. The plot below shows the timing results of experiments registering a CLARITY brain template to the Allen Reference Atlas (ARA) target. In the first experiment the CLARITY image was registered to the (ARA) using a single-scale approach on a 50 μm grid. In the second experiment registration was done by a coarse to fine multi-scale method. Registration that was done at a lower resolution was used to initialize the algorithm at the subsequent higher resolution level. Resolution levels of 800, 400, 200, 100 and then 50 μm were used. The Mutual Information (MI) between the deformed CLARITY and ARA images was normalized to a range of [0, 1.0] where 1.0 indicates that no registration occurred and 0 indicates the best possible scenario ($I_0 = I_1$). The plot shows that the multi-scale optimization was much faster than the single-scale method.

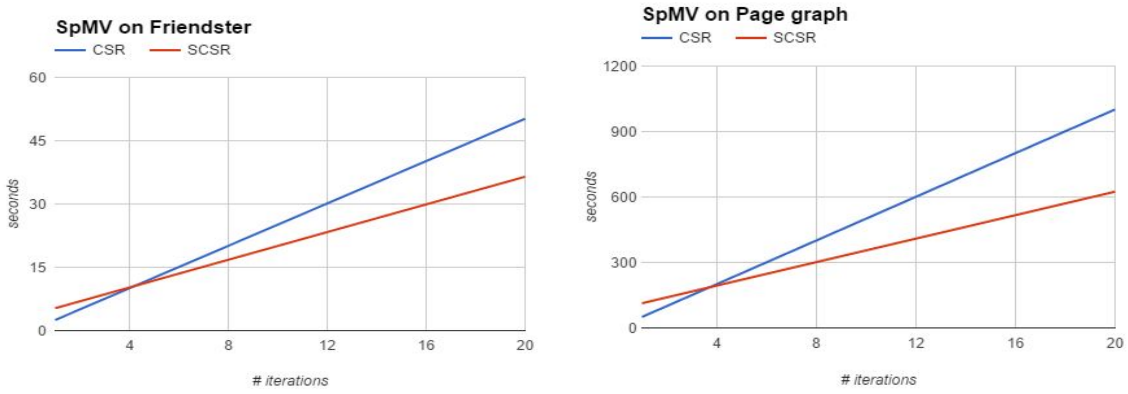


The images below are checkerboard composite images of the deformed CLARITY image and the ARA. They show that usage of the multi-scale approach did not reduce registration quality when compared to the single-scale method.



FlashX

Sparse matrix multiplication in FlashX uses its own format (SCSR) for sparse matrices to improve performance. We investigated the overhead of using customized format to thoroughly evaluate the performance of sparse matrix multiplication in FlashX. Thus, it is essential to accelerate the format conversion from a standard format such as CSR to our customized format SCSR. Figure \ref{fig:flashx1}, below, shows the benefit of using the SCSR format including the overhead of format conversion. When an application requires more than 4 SpMVs, converting the format of a sparse matrix improves the performance of the application. Thus, majority of the applications benefit from the format conversion.

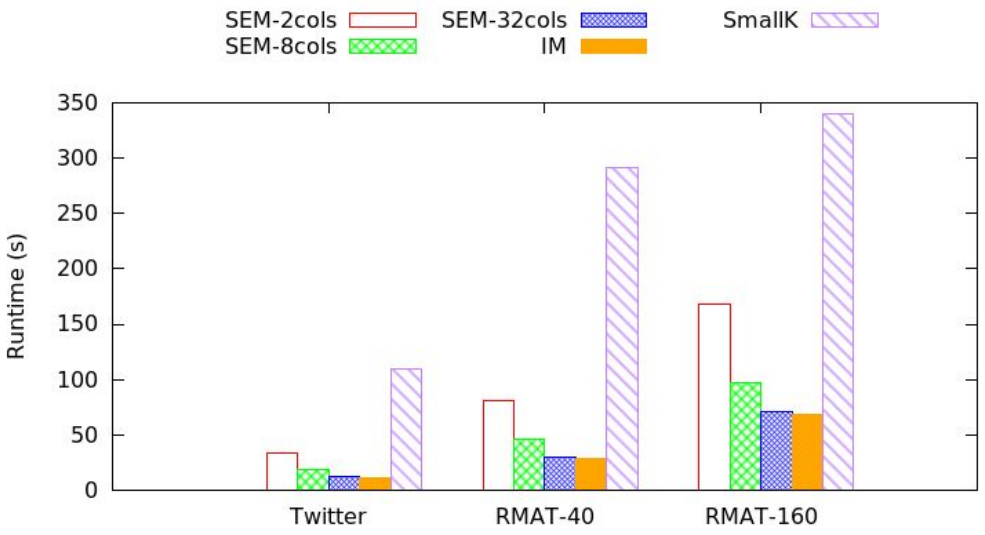


The performance ratio of the in-memory and semi-external memory sparse matrix multiplication in FlashX is affected by multiple factors. Shown in Figure \ref{fig:fx2}, we demonstrate some of the factors with SBM graphs with the same number of vertices and edges. We vary the number of clusters and the number of edges inside clusters. We measure the performance of SpMV on both clustered and unclustered graphs. When vertices are ordered based on the cluster structure, more clusters and more edges inside clusters increase CPU cache hits, which leads to less computation overhead and larger performance gap between in-memory and semi-external memory executions. However, if vertices are ordered randomly, these two factors have less obvious impact on performance.

We implement Daniel Lee's NMF algorithm with our semi-external memory sparse matrix multiplication (denoted as SEM-NMF) and evaluate its performance by comparing it with a high-performance NMF implementation SmallK on billion-scale graphs. The dense matrices for NMF can be as large as the sparse matrix. As such, we split the dense matrix vertically so that each of the vertical partitions can fit in memory. We also evaluate the effect of the memory size on the performance of SEM-NMF by varying the number of columns in memory from the dense matrices. In the experiment, we factorize each of the graphs into two n × 16 non-negative dense matrices.

We significantly improve the performance of SEM-NMF by keeping more columns of the input dense matrix in memory (Figure \ref{fig:nmf}). The performance improvement is more significant when the number of columns that fit in memory is small. When we keep eight columns of the input dense matrix in memory, SEM-NMF achieves over 60% of the performance of its in-memory execution.

SEM-NMF significantly outperforms SmallK and other NMF implementations in the literature. SmallK is the closest competitor. We run the same NMF algorithm in SmallK and SEM-NMF outperforms SmallK by a large factor on all graphs (Figure \ref{fig:nmf}). There are many MapReduce implementations in the literature. They run on sparse matrices with tens of millions of non-zero entries but generally take one or two orders of magnitude more time than our SEM-NMF on the sparse matrices with billions of non-zero entries.


SEM-2cols
SEM-8cols
SEM-32cols
IM
SmallK
Runtime (s)
350
300
250
200
150
100
50
0
Twitter
RMAT-40
RMAT-160

FlashY

FlashY is created to encapsulate the R packages created for multiple-graph processing and clustering.

We consider improving the stability in clustering objects. When the sample size is low but there is large variability due to the high dimension, the inferences based on the existing clustering methods (k-means, Gaussian mixture model, etc.) tend to produce large error. This problem is critical and common in graph clustering, as the number of graphs is typically low but the number of vertices is high. To solve this, a new clustering method, namely ``jk-means'' is developed in R. With a truncation, substantial error reduction can be obtained via the bias-variance tradeoff. Shown in the figure, the new jk-means clustering tool (red) produces significantly better result than the other models.

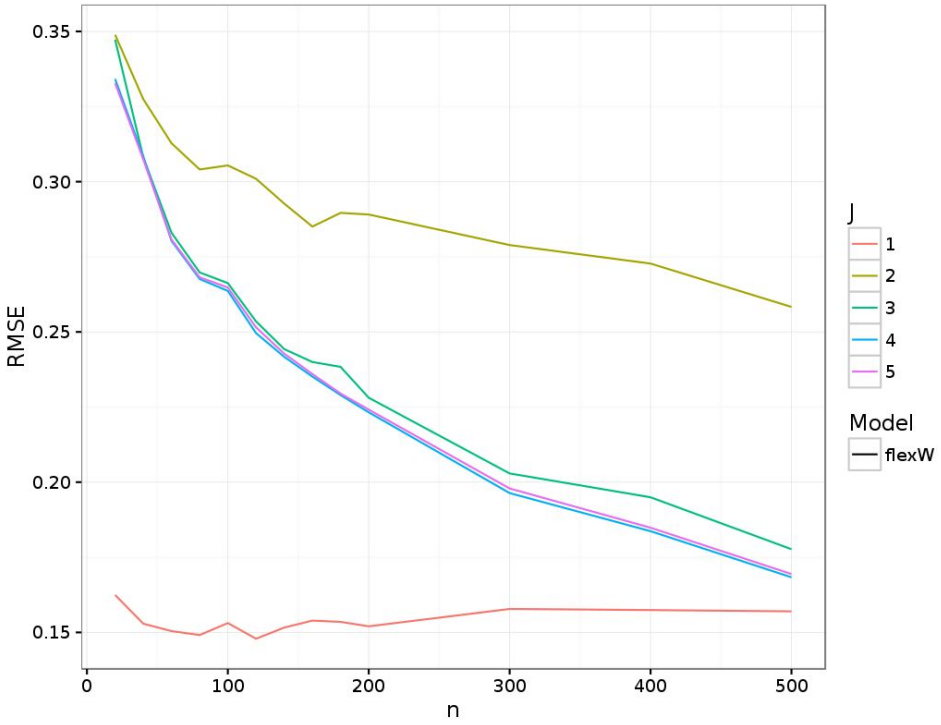
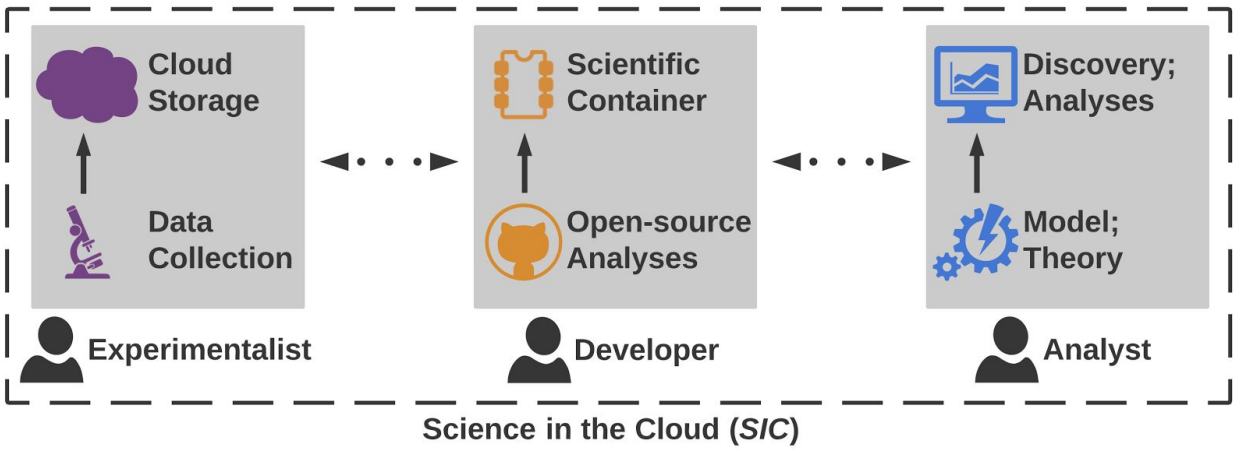


Illustration of the jk-means, compared with Gaussian Mixture Model (j=5). The j=1 jk-means model (red) has significantly lower error compared to the others, when the sample size is smaller than 500.

Science in the Cloud (SIC)

A framework for *science in the cloud* (SIC) was developed which was designed for extensibility of scientific findings. This work was developed alongside a live demonstration, 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.

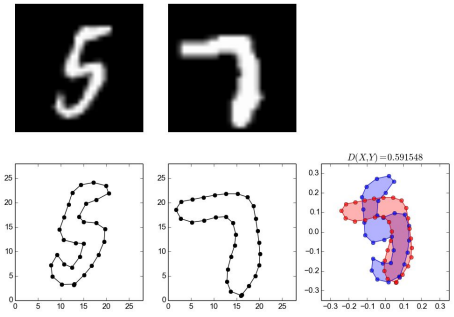


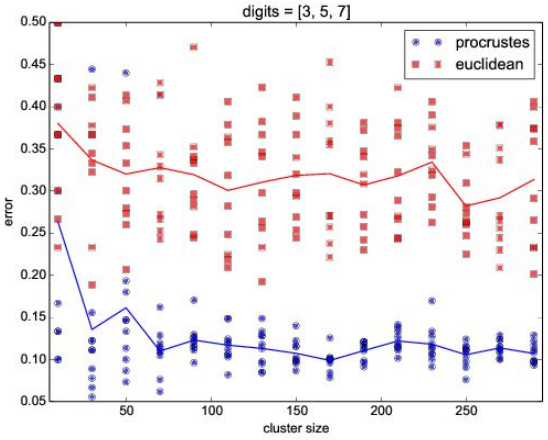
Science in the Cloud (*SIC*)

Non-Parametric and Shape Clustering

A shape is the geometric information contained in an object after translation, scaling, and rotation is removed, and is represented by an ordered set of landmark points. We are interested in using a distance shape metric in K-means clustering algorithm, enabling us to aggregate objects with similar shapes together. Our shape distance is defined through Procrustes alignment.

To test our algorithm we used the well-known MNIST handwritten digits dataset. For each digit we extract landmark points, and align them through Procrustes. This procedure defines a distance between any two objects, and is illustrated in the first figure below. Using this distance we then cluster three classes of digits using a modified K-means algorithm. The classification error against the size of each cluster --- the three classes have the same number of points --- is shown in the second figure below in blue. The red line is standard K-means with Euclidean distance for comparison. This particular example shows a pretty low error, around 10%, for an unsupervised clustering method.

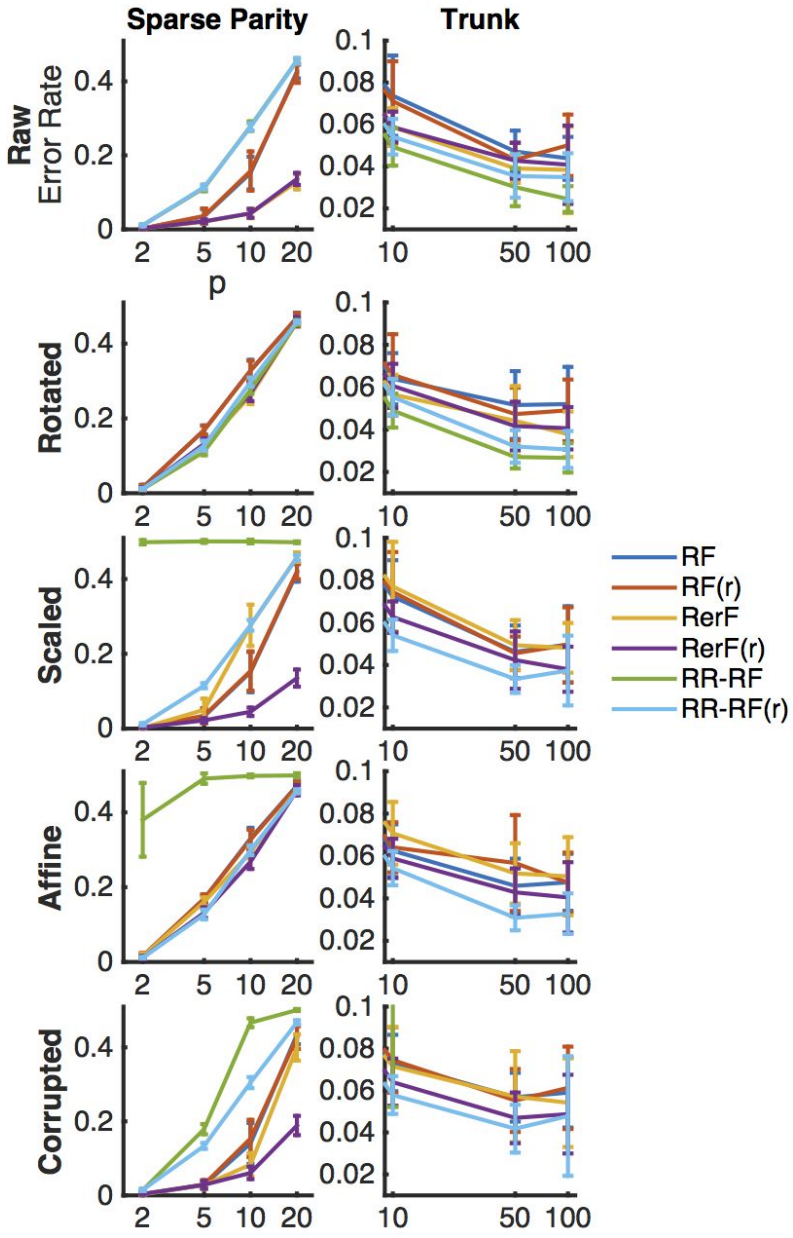




Randomer Forest (RerF)

Random Forest (RF) remains one of the most widely used general purpose classification methods, due to its tendency to perform well in a variety of settings. One of its main limitations, however, is that it is restricted to only axis-aligned recursive partitions of the feature space. Consequently, RF is particularly sensitive to the orientation of the data. Several studies have proposed “oblique” decision forest methods to address this limitation. However, the ways in which these methods address this issue compromise many of the nice properties that RF possesses. In particular, unlike RF, these methods either don’t deal with incommensurate predictors, aren’t well-adapted to problems in which the number of irrelevant features are overwhelming, have a time and space complexity significantly greater than RF, or require additional hyperparameters to be tuned, rendering training of the classifier more difficult. Additionally, oblique methods tend to be more sensitive to data corruption than RF is. Our proposed method, which we call RerF, seeks to address these limitations.

Most recently, we performed simulations in which we artificially rotated, scaled, rotated+scaled (labeled “affine” in the figure), and corrupted two simulated datasets. The results demonstrate that both RerF and another existing oblique method called RR-RF are more sensitive to scaling and corruption of the data than is RF. The variants of these two methods that use rank transformations to rescale the data first, called RerF(r) and RR-RF(r), become significantly more robust to scale and corruption. While rank transformation is not the only scale normalizing method (z-scoring and forcing all values to be between [0,1] are others), it is the only one that simultaneously helps against data corruption. Across all simulation settings, RerF(r) is the overall best.



We are developing an R implementation of the RerF algorithm in order to make it faster and more easily distributable. The Rerf algorithm requires a random rotation of data at each node during the tree creating process, but this rotation of data at each node increases memory requirements and both training and testing times. So, the initial R implementation will instead rotate the sample data once prior to creating a tree instead of at each node in the tree. This will still increase the same factors but not to the same degree. Initial tests show that this single rotation per tree yields benefits similar to the per node rotation.

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: $D= P (\Delta(x_{i,t} , x_{i,t'}) \leq \Delta(x_{i,t} , x_{i',t''}))$
We want to search for the optimal processing pipeline which has the maximal discriminability. Previously, we process 13 fMRI data sets with 64 pipelines, and show some pipelines yields data sets with high sample discriminability. Currently, we are developing a statistical test to determine whether one pipeline is significant better than the other. To develop a valid test, we need to approximate the distribution of difference between sample discriminability estimates under the null hypothesis that two population discriminability are equal. We design and implement a bootstrap procedure to achieve this which is described below.

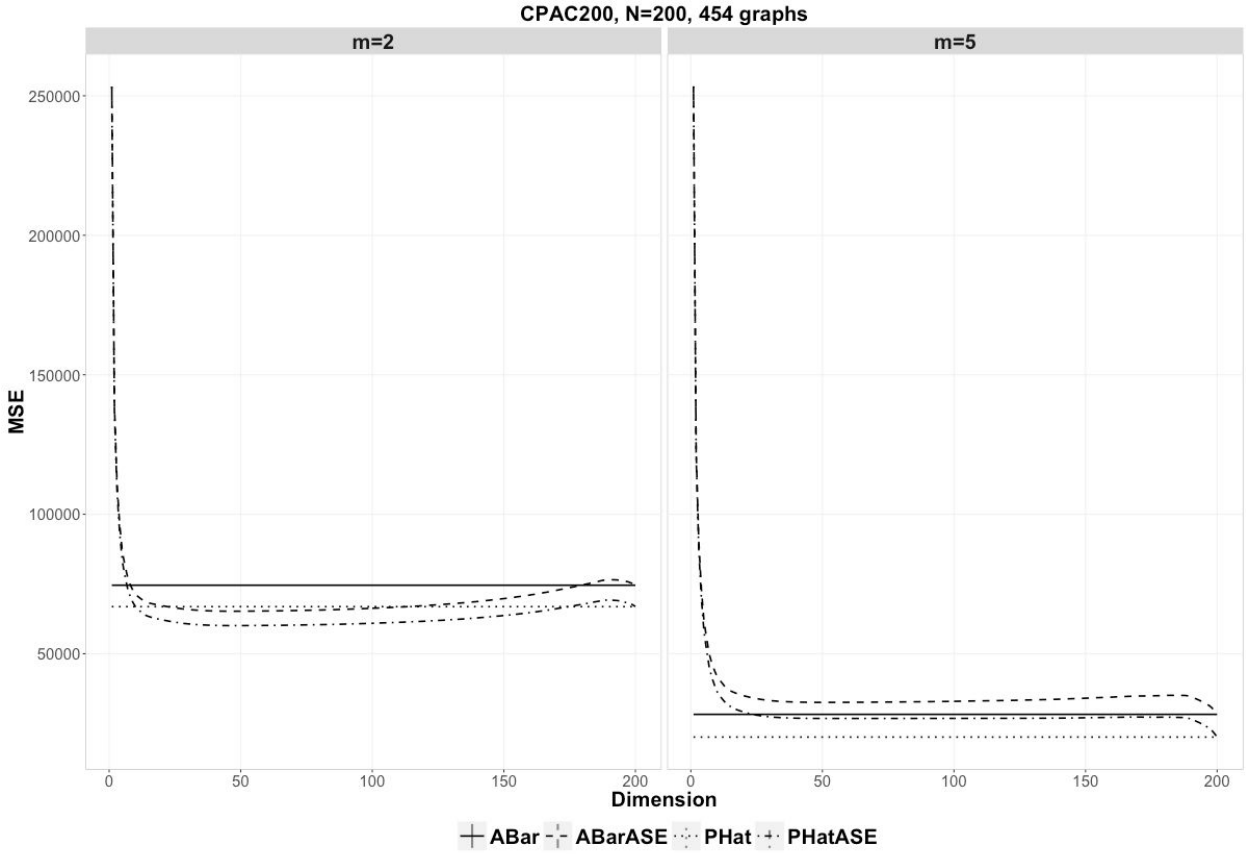
Testing the null hypothesis $D(\Psi_1) = D(\Psi_2)$
Input: Data set, Pipeline Ψ_1, Pipeline Ψ_2
Output: Reject or do not reject the null hypothesis

1. Process the data set with pipelines Ψ_1 and Ψ_2
2. Compute $D_{hat}(\Psi_1)$ and $D_{hat}(\Psi_d)$
3. For {i in 1 through number of repeats}
    a. For {j in 1 through number of subjects}
        i. Randomly select two subjects from data set
        ii. Linearly combine measurements of these subjects
    b. EndFor
    c. Form two bootstrapped data sets processed by Ψ_1 and Ψ_2
    d. Compute $D_{hat}^{(i)}(\Psi_1)$ and $D_{hat}^{(i)}(\Psi_2)$
4. EndFor
5. Compute pairwise differences $D^{(i)}(\Psi_1) - D^{(i')}(\Psi_1)$ and $D^{(i)}(\Psi_2) - D^{(i')}(\Psi_2)$
6. Compute p-value which is the fraction of times that $\$D_{hat}(\Psi_1)-D_{hat}(\Psi_2) > D^{(i)}(\Psi_j) - D^{(i')}(\Psi_j)$
7. Reject the null hypothesis if p-value is less than 0.05.

We are currently running the test procedure on 13 data sets processed 64 ways. After finish running, we will have 13 p-values for each pipeline. We will use Fisher's method to combine results. The following figure is going to be updated with more significant p-value.

Robust Law of Large Graphs

We tested our algorithm's performance on a synthetic dataset based on structural connectomic data. The graphs are based on diffusion tensor MR images collected and available at the Consortium for Reliability and Reproducibility (CoRR). It contains 454 different brain scans, each of which was processed to yield an undirected, weighted graph with no self-loops, using the ndmg pipeline..
The vertices of the graphs represent different regions in the brain defined according to an atlas. We used the CPAC200 atlas with 200 vertices. The weight of an edge between two vertices represents the number of white-matter tract connecting the corresponding two regions of the brain. In order to evaluate the performance of the two estimators, we used a cross validation on the 454 graphs of each size. Specifically, for a given atlas, each Monte Carlo replicate corresponds to sampling m graphs out of the 454 and computing the four estimates using the m selected graphs. We then compared these estimates to the sample mean for the remaining 454-m adjacency matrices.
We ran 1000 simulations on the CPAC200 atlases for both sample size M=2, 5.
We also considered all possible dimensions for adjacency spectral embedding by ranging d from 1 to n in order to investigate the impact of the dimension selection procedures.
We plot the result in the following figure.



Comparison of MSE_{hat} of the four estimators for the CPAC200 atlases at two sample sizes for the CoRR data.

For the figure on the left, we examine the four estimators at sample size m=2. **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. For the figure on the right, we still can see the robustness but the lack of low-rankness for a larger number of samples degrades the effect of ASE.

Bibliography

T. M. Tomita, M. Maggioni, and J. T. Vogelstein. ROFLMAO: Robust Oblique Forests with Linear MAtrix Operations. Proceedings of the 2017 SIAM International Conference on Data Mining. *Under Review*.

Joshua T. Vogelstein, et al. To the Cloud! A Grassroots Proposal to Accelerate Brain Science Discovery. Neuron, NeuroView. *Released on Nov 2, 2016*.

Joshua T. Vogelstein, et al. Grand Challenges in Global Brain Science. F1000 Research. *Under Review.* 2016.