NeuroData SIMPLEX Report: May 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] C. E. Priebe, Y. Park, M. Tang, A. Athreya, V. Lyzinski, J. T. Vogelstein, Y. Qin, B. Cocanougher, K. Eichler, M. Zlatic et al., "Semiparametric spectral modeling of the Drosophila connectome," arXiv preprint arXiv:1705.03297, 2017.

[2] D. Fishkind, S. Adali, H. Patsolic, L. Meng, V. Lyzinski, and C. Priebe, "Seeded Graph Matching," arXiv preprint arXiv:1209.0367, 2017.

[3] H. Patsolic, Y. Park, V. Lyzinski, and C. Priebe, "Vertex Nomination Via Local Neighborhood Matching," arXiv preprint arXiv:1705.00674, 2017.

Talks

[1] T. Tomita, "Roflmao: Robust Oblique Forests with Linear Matrix Operations," SIAM International Conference on Data Mining, Apr 2017, Contributed talk.

Conferences

[1] T. Tomita, "Roflmao: Robust Oblique Forests with Linear Matrix Operations," SIAM International Conference on Data Mining, Apr 2017, Contributed poster.

2 Statistical Theory and Methods

2.1 meda @JesseLP

A docker container https://hub.docker.com/r/neurodata/synaptograms/ has been developed to view multi-channel muti-dimensional image data. Array tomography (AT) provides a good use-case. AT data consist of (3+c)-dimensional data, where c is the number of features/channels collected. A putative synapse location corresponds to an 11x11x11 pixel cube in the image and this cube will have multiple corresponding channels. These channels can express the presence or absence of a certain protein marker such as Synapsin or VGlut2.

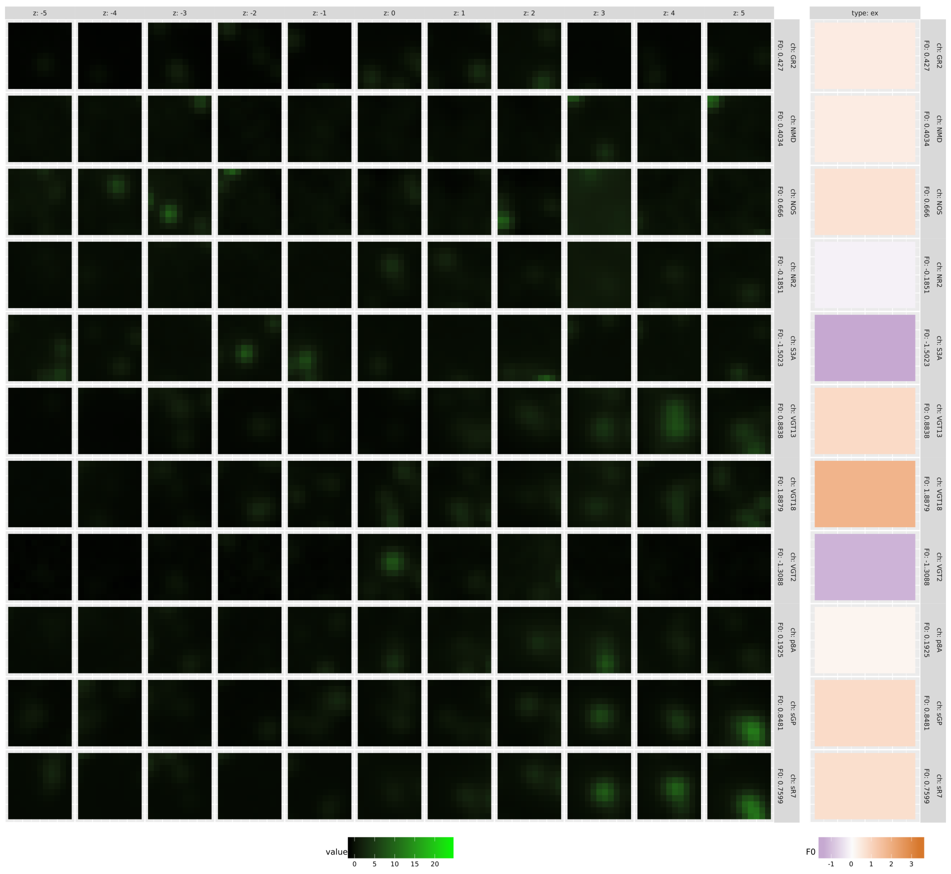


Figure 1: Using a synapse detection algorithm we believe that there is a synapse at this location. The location is defined by an 11x11x11 cube with 11 feature channels. Each row of the figure corresponds to the cube over a specific feature channel. The cube has been splayed out in the z-direction over the columns. The far right column corresponds to the sum statistic of each row. This plot enables viewing of an 11x11x11xc object nicely on screen, where c in this case happens to be 11.

2.2 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.
In practice, MGC often achieves the same testing power using much less sample size than its competitors, which is important for data collection purpose. This is shown in Figure 2.

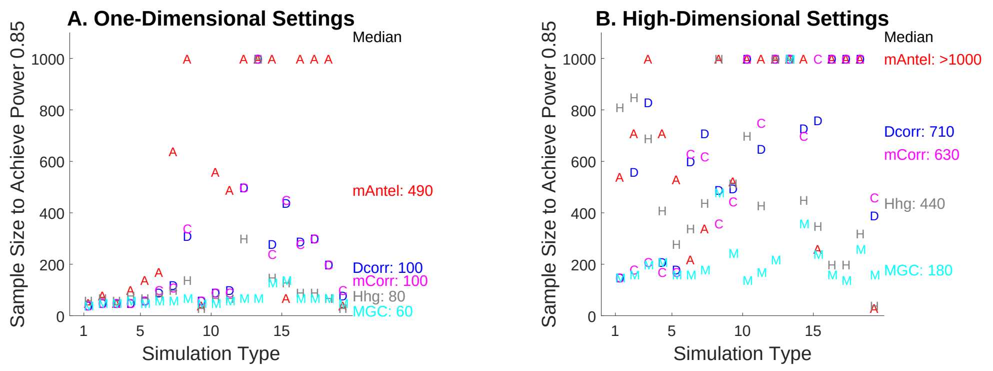


Figure 2: Sample size of different methods to achieve a power of 85% at type 1 error level 0.05, for the 20 different settings under 1-dimensional (A) and high-dimensional (B) settings. The x-axis is the simulation type, and y-axis shows the minimal sample size of each method to achieve the required testing power, the smaller the better. We bound the sample size to 1000 in visualization, with the median sample size for each method reported in the far right column. The results indicate that Mgc is a superior choice for finite-sample dependency testing, e.g., for the second best method Hhg, on the median it requires greater than twice the sample size of Mgc to achieve the same power.

2.3 Multiscale Network Testing for Two-Graph

We invented multiscale network test via diffusion maps and MGC, and extends its utility into testing two graphs of the same node set with different edge sets. Assume two graphs $\mathbf{G}_1$ and $\mathbf{G}_2$ are generated via a latent variable $\mathbf{u}_i = (u_{1i}\, u_{2i}\, \cdots\, u_{5i}) \in \mathbb{R}^5$ as follows:

$$\begin{aligned} u_{ki} &\overset{i.i.d.}{\sim} Unif(0,1), \quad i = 1, 2, \ldots, n;\ k = 1, 2, \ldots, 5 \\ w_i &:= (1 - u_{i1})^2, \quad i = 1, 2, \ldots, n \\ A^{(1)}_{ij} | \mathbf{u}_i, \mathbf{u}_j &\sim Bernoulli\big(< \mathbf{u}_i/5, \mathbf{u}_j/5 > \big), \quad \forall i < j;\ i, j = 1, 2, \ldots, n;\ \mathbf{u}_i, \mathbf{u}_j \in \mathbb{R}^5 \\ A^{(2)}_{ij} | w_i, w_j &\sim Bernoulli\big(< w_i, w_j > \big), \quad \forall i < j;\ i, j = 1, 2, \ldots, n. \end{aligned} \tag{1}$$

That is, Each graph is generated by a random dot product graph (RDPG), and the underlying dependency is reflected via the quadratic function of one-dimensional latent variable; this implies both multi-dimensional and nonlinear relationship where MGC is preferred to other benchmarks in testing network dependency in nodal attributes.

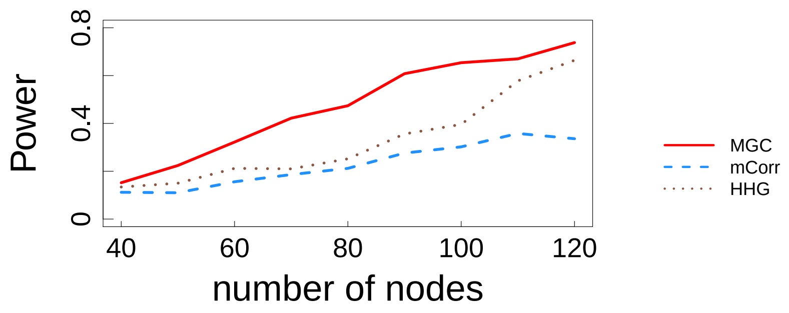


Figure 3: The power curve with respect to increasing number of nodes for the two-graph dependency testing simulation (Equation (1)). The proposed approach achieves higher power than other methods.

Figure 3 shows the testing power of MGC, mCorr, and HHG against the number of nodes n, all based on the diffusion maps, and it demonstrates that the proposed approach is able to achieve higher testing power under relatively small number of nodes. Note that if noise is included in the set-up, or the nonlinear relationship is more complex than quadratic, the proposed approach still enjoys the same advantage, i.e., the testing power converges to 1 faster than all other methods, though the actual number of nodes to achieve perfect power will likely increase under noisy and complex dependency.
The draft is submitted this month and available on arXiv.

2.4 Randomer Forest (RerF)

Previously, we had demonstrated that RerF tends to outperform other tree ensemble algorithms on synthetic datasets as well as a large suite of benchmark datasets. Our current effort is to understand when we can expect RerF to win and when we can expect it to lose. As a first step, we have made scatter plots of classification error against two metrics known to be influential in the performance of ensembles: 1) strength of individual weak learners and 2) diversity of weak learners. Below, the plots suggest that strength seems to have a stronger correlation with classification performance than does diversity. RR-RF tends to have lower average tree strength than both RF and RerF. RerF tends to have higher average tree strength than RF.

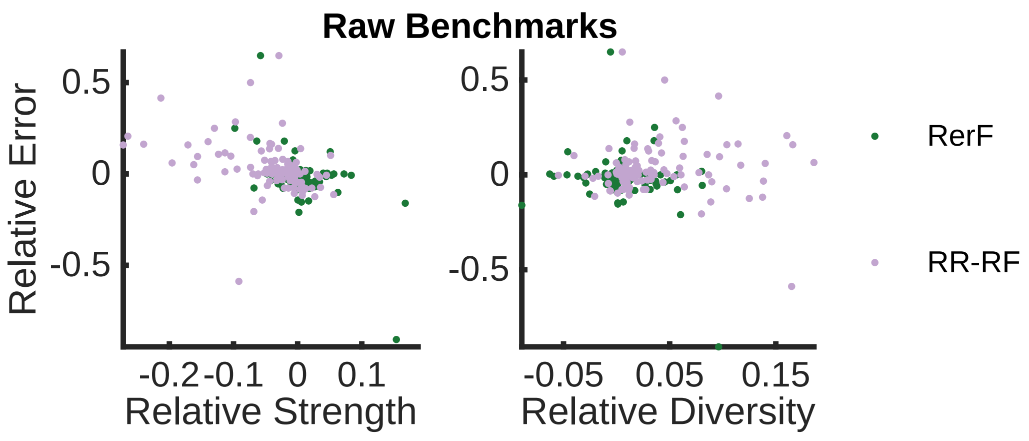


Figure 4: For each of the 114 benchmark datasets, the errors of RerF and RR-RF relative to RF were plotted against 1) the average strength of these algorithms relative to RF (left) and 2) the diversity of trees of these algorithms relative to RF. Tree strength is defined as the individual classification accuracy of a tree. Diversity is defined as the variance in predictions made by the individual trees.

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.

The R version of Rerf, R-Rerf, has been tested on a 400Mb artificial dataset. The training time on this data set is about 3.5 minutes/tree using 1 core, with the training time scaling linearly with the number of cores added. The increased memory requirements of the multicore implementation are higher than anticipated though – requiring 8 times the size of the input data per core. To reduce the memory requirements we are testing depth first vs breadth first tree growing methods.

2.5 Vertex Screening

We are developing a vertex screening method to recover the signal subgraph. Specifically, we have m pairs of graph and label sampled independently from some distribution $F_{G,Y}$,

$$(G_1, Y_1), (G_2, Y_2), (G_3, Y_3), ..., (G_m, Y_m) \overset{i.i.d.}{\sim} F_{G,Y}.$$

It is often the case that the signal is sparse in G. That is to say, there is a small subgraph $G[S]$ induced by signal vertices S which contains all information about Y. The vertex screening method wants to recover the signal vertices S. It consists of three steps: feature extraction, computing distance correlations, and thresholding.

The first step is to extract a feature vector for each vertex in a graph. We use notation $\hat{X}_i[u, \cdot]$ to denote the feature extracted for vertex u in graph i where $i \in [m]$ and $u \in [n]$. A simple approach to obtain a feature vector is to set $\hat{X}_i[u]$ to the uth row of adjacency matrix A_i, that is $\hat{X}_i[u, \cdot] = A_i[u, \cdot]$. In this case, $\hat{X}_i[u, \cdot]$ is a vector in $\mathbb{R}^n$ which can be a high dimensional space. Alternatively, Adjacency Spectral Embedding could also extract a feature vector $\hat{X}_i[u, \cdot]$ which lies in $\mathbb{R}^d$. The second step computes a correlation between $\{\hat{X}_i[u, \cdot]\}_{i=1}^m$ and $\{Y_i\}_{i=1}^m$ for each $u \in V$. The correlation could be distance correlation (Dcorr) or multiscale generalized correlation (MGC). Let c_u be the correlation, that is

$$c_u = Dcorr(\{\hat{X}_i[u, \cdot]\}_{i=1}^m, \{Y_i\}_{i=1}^m) \text{ or } MGC(\{\hat{X}_i[u, \cdot]\}_{i=1}^m, \{Y_i\}_{i=1}^m).$$

The last step order c_us by their magnitudes. Then, we threshold the correlations by a critical value c. The vertices survive thresholding will be the estimated signal vertices $\hat{S}$, that is

$$\hat{S} = \{u \in V | c_u > c\}.$$

The estimated signal subgraph and the corresponding adjacency matrix $G[\hat{S}]$ and $A[\hat{S}]$. The Algorithm 1 describes the general procedure of vertex screening using adjacency vector as feature and MGC. A simulation experiment is shown in the Figure 5 which demonstrates that the vertex screening can significantly improve the graph classification performance for various sample size. We are currently working on theories of vertex screening and finishing the first draft.

Algorithm 1 Vertex Screening. Find the signal vertex estimate $\hat{S}$.

```
1: procedure Input $\{(A_i, Y_i)\}_{i=1}^m$ and $c \in [0, 1]$
2:     for $u = 1 : n$ do
3:         $c_u = MGC(\{\hat{X}_i[u, \cdot]\}_{i=1}^m, \{Y_i\}_{i=1}^m)$
4:     Output $\hat{S} = \{u | c_u > c\}$
```

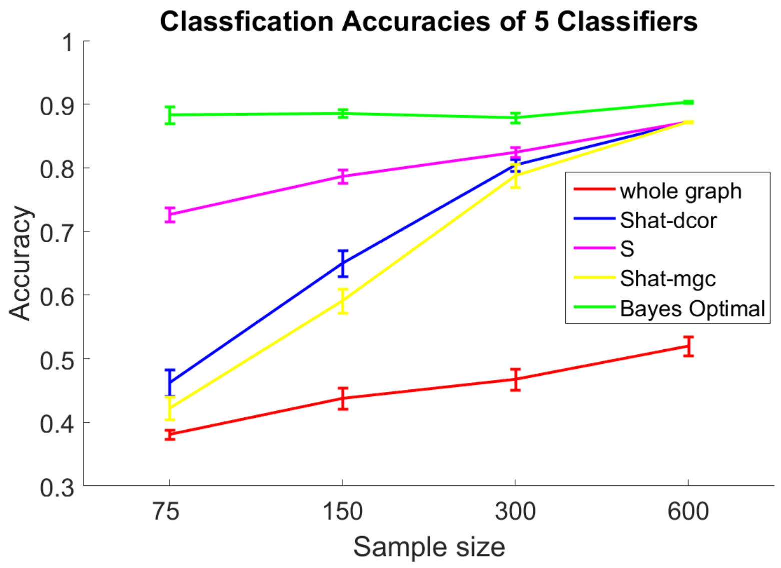


Figure 5: The classification accuracies of five approaches with their standard errors are shown. We generate graphs from 3 different inhomogeneous Erdos-Renyi model, then apply 5 classifiers to classify these graphs: Bayes optimal classifier (green), Bayes plugin on $G[S]$ (purple), Bayes plugin on $G[\hat{S}]$ with $\hat{S}$ estimated by Dcorr (blue), Bayes plugin on $G[\hat{S}]$ with $\hat{S}$ estimated by MGC (yellow), and Bayes plugin on G (red). The classifiers with vertex screening (blue and yellow) have significantly better classification performance compared to without screening (red), and are close to Bayes optimal (green) when given 600 graphs.

2.6 Joint Embedding

The latest draft is posted on arXiv and submitted for publication. The code can be found here. We are also working on making a R package with joint embedding included.

2.7 Law of Large Graphs

We note that low-rank methods can often be more easily interpreted. Moreover, eigenmode is observed among the embedded latent positions, with respect to different lobes in particular. This suggests to use low-rank methods from another perspective. For all the 70 different regions based on the Desikan atlas (35 for each hemisphere), each one is assigned to one of the 10 lobes (5 for each hemisphere), i.e. Frontal, Parietal, Occipital, Temporal, and other. And we do a permutation test as following.
70 vertices are connected spatially as in the Desikan atlas. Let the adjacency matrix of these 70 vertices to be A. $A_{ij} = 1$ means vertex i and vertex j are spatially connected. We say vertex j is a neighbor of vertex i if $A_{ij} = 1$. We define l_i be the lobe i.d. for vertex i.
We define a uniform 1-flip to be:

- Select a pair of adjacent vertices (vertex i_1 and vertex j_1) across the boundary of lobes uniformly, i.e. $A_{i_1 j_1} = 1$ and $l(i_1) \neq l(j_1)$;
- Uniformly select another pair of adjacent vertices (vertex i_2 and vertex j_2 where $i_1 \neq i_2$ and $j_1 \neq j_2$) across the same boundary of lobes uniformly, i.e. $A_{i_2 j_2} = 1$ and $l(i_1) = l(i_2)$ and $l(j_1) = l(j_2)$;
- Reassign vertex j_1 to lobe l_{i_1} and reassign vertex i_2 to lobe l_{j_2}.

By the definition, after a uniform 1-flip, the number of vertices in each lobe keeps the same, where only two vertices are changed to a different lobe.
We define a uniform k-flip to be:

- Sequentially run the uniform 1-flip k times.

Note that after a uniform k-flip, the number of vertices in each lobe still keeps the same.
Let $X = [X_1, \cdots, X_n]^\top$ be the latent positions, where X_i is the latent position for vertex i sampled from distribution f. Test statistic $T(X, l)$ is defined as:

$$T(X,l) = \frac{\sum_{i \neq j, l(i)=l(j)} \|X_i - X_j\|_2}{\sum_{i \neq j, l(i)=l(j)} 1} - \frac{\sum_{i \neq j, l(i)\neq l(j)} \|X_i - X_j\|_2}{\sum_{i \neq j, l(i)\neq l(j)} 1}$$

H_0: Differences between latent positions within lobes are the same compared to across lobes, i.e. $E_f[T(X,l)] = 0$.
H_A: Differences between latent positions within lobes are smaller compared to across lobes, i.e. $E_f[T(X,l)] < 0$.
We ran 1000 simulations for each number of flips and plot the results for the permutation test as in Figure 6. The x-axis represents the different number of flips, while the y-axis represents the measure according to the lobe assignment, i.e. within lobes distances minus the across lobe distances. The dashed line is the baseline for the measure based on the true lobe assignment without any flipping. As the number of flips increases, we can see a clear evidence that H_0 is rejected with respect to H_A. Thus the embedded latent positions based on the low-rank method reflect the eigenmode with respect to different lobes.

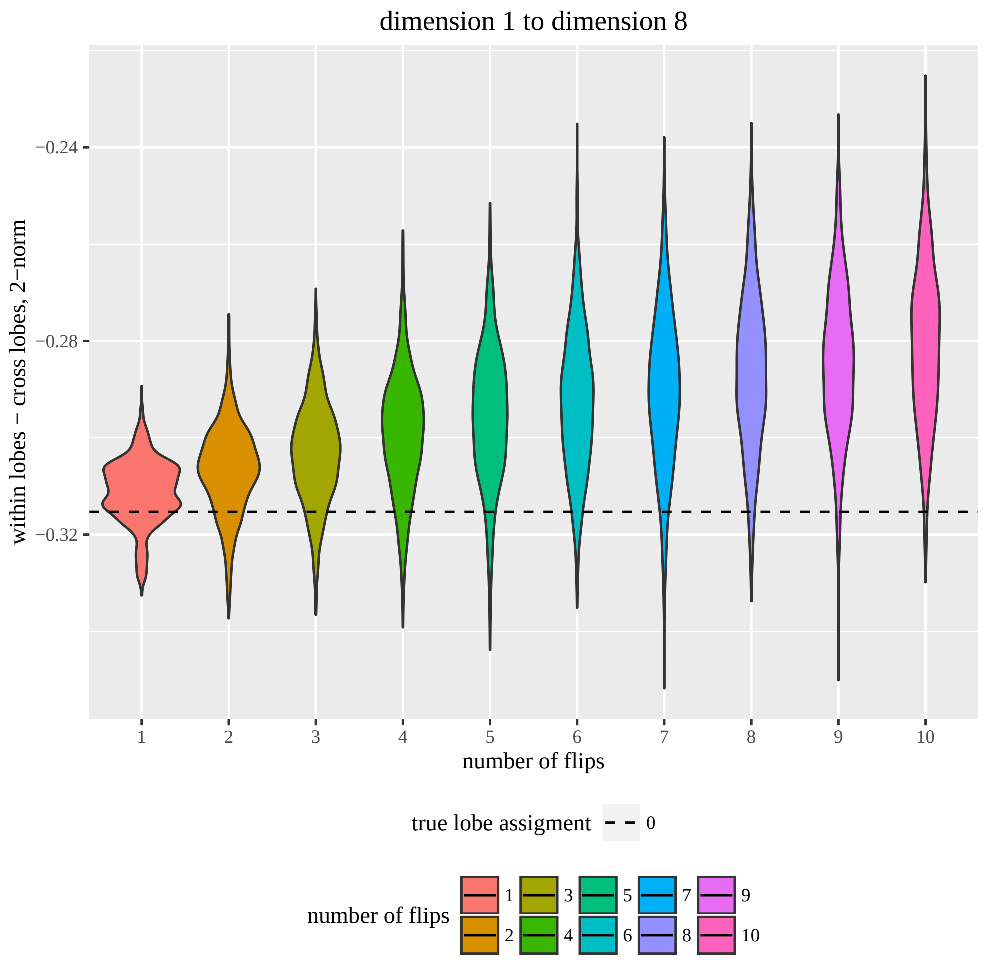
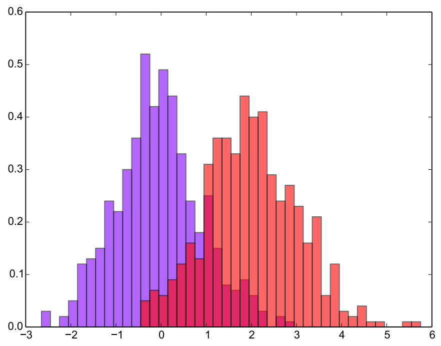


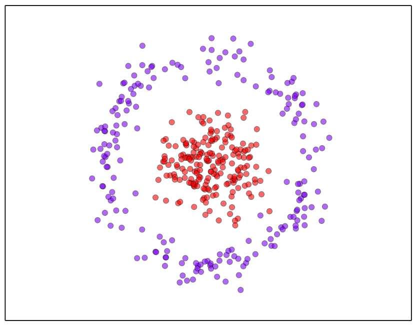
Figure 6: **Violin plot of the permutation test.** We ran 1000 simulations for each number of flips. The x-axis represents the different number of flips, while the y-axis represents the measure according to the lobe assignment, i.e. within lobes distances minus the across lobe distances. The dashed line is the baseline for the measure based on the true lobe assignment without any flipping. As the number of flips increases, we can see a clear evidence that H_0 is rejected with respect to H_A. Thus the embedded latent positions based on the low-rank method reflect the eigenmode with respect to different lobes.

2.8 Non-Parametric Shape Clustering

We have been developing a non-parametric method for clustering based on energy statistics. We were able to formulate the problem in a precise mathematical form, which amounts to solve a quadratic optimization problem with quadratic constraints, which is NP-hard. Using the relation between energy statistics and kernel functions, we implemented an algorithm in the same spirit as kernel k-means, but consistent with energy statistics. Below we show some preliminary results, where we cluster one-dimensional normal data, and also two-dimensional circles with noise. The results of our method are shown in the column $\mathcal{E}$-clustering. We compare with standard k-means and gaussian mixture models (GMM). We compute the accuracy (which is between $[0, 1]$, the higher the better) based on the true labels. In this example we can see that $\mathcal{E}$-clustering attains the same accuracy as GMM and k-means for normally distributed data, while it is superior for data that are not normally distributed. Now that we have a precise mathematical formulation and also a working clustering algorithm, we will start to perform more consistent numerical tests.



$\mathcal{E}$-clustering	GMM	k-means
0.823	0.798	0.826
0.852	0.852	0.853
0.842	0.842	0.839



$\mathcal{E}$-clustering	GMM	k-means
0.74	0.516	0.536
0.706	0.516	0.503
0.633	0.553	0.52

Figure 7: (a) Two clusters in one dimension where $x \sim \frac{1}{2}(\mathcal{N}(\mu_1, \Sigma_1) + \mathcal{N}(\mu_2, \Sigma_2))$, with $\mu_1 = 0$, $\mu_2 = 2$, $\Sigma_1 = \Sigma_2 = 1$, and 500 points on each cluster. (b) We generate two circles with random noise. The radius of the circles are 1 and 0.2. The gaussian noise has size 0.2. We have 200 points on each class. The tables below each figure show the accuracy, based on the true labels, for three experiments. We compare our method with GMM and k-means. We can see that for gaussian data the three of them achieve the same accuracy, which is close to Bayes error, while for the data in (b) energy statistics based clustering is superior.

2.9 Robust Law of Large Graphs

In order to see a bias-variance tradeoff phenomenon with respect to the parameter q in the MLqE estimator, we change our simulation setting as following: We consider the 2-block SBM with respect to the exponential distributions parameterized by

$$B = \begin{bmatrix} 4 & 2 \\ 2 & 7 \end{bmatrix}, \qquad \rho = \begin{bmatrix} 0.5 & 0.5 \end{bmatrix}.$$

And let the contamination also be a 2-block SBM with the same structure parameterized by

$$B' = \begin{bmatrix} 9 & 6 \\ 6 & 13 \end{bmatrix}, \qquad \rho = \begin{bmatrix} 0.5 & 0.5 \end{bmatrix}.$$

Figure 8 shows the mean squared error in average by varying the parameter q in MLqE with fixed $n = 100$, $m = 20$ and $\epsilon = 0.1$ based on 1000 Monte Carlo replicates. Different types of lines represent the simulated MSE associated with four different estimators. From the figure, we can see that the ASE procedure takes advantage of the graph structure and improves the performance of the corresponding estimators independent of the selection of q. Moreover, within a proper range of q, the MLqE wins the bias-variance tradeoff and shows the robustness property compare to the MLE. And as q goes to 1, MLqE goes to the MLE as expected.

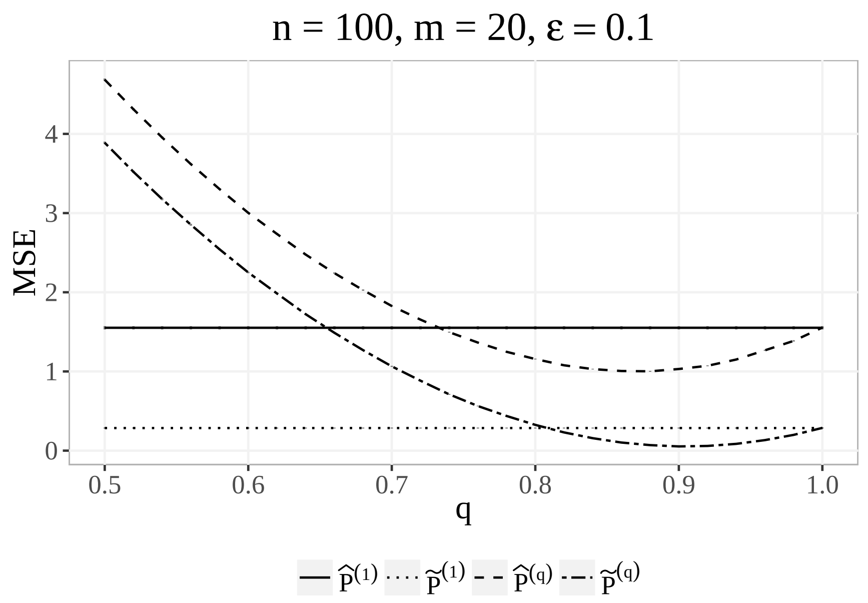
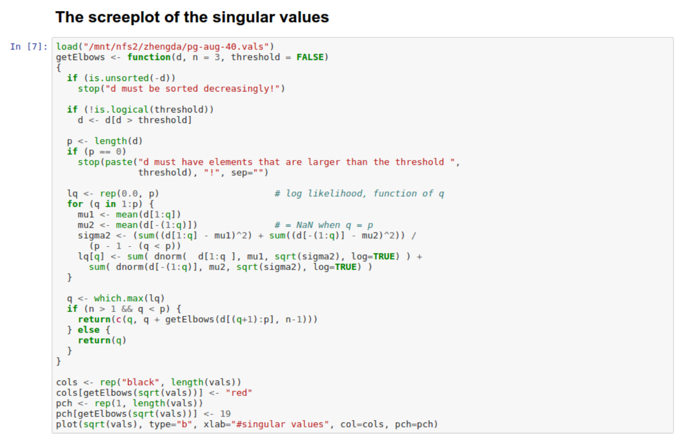


Figure 8: Mean squared error in average by varying the parameter q in MLqE with fixed $n = 100$, $m = 20$ and $\epsilon = 0.1$ based on 1000 Monte Carlo replicates. Different types of lines represent the simulated MSE associated with four different estimators. 1. ASE procedure takes advantage of the graph structure and improves the performance of the corresponding estimators independent of the selection of q; 2. Within a proper range of q, MLqE wins the bias-variance tradeoff and shows the robustness property compare to the MLE. Also as q goes to 1, MLqE goes to the MLE as expected.

3 Scalable Algorithm Implementations

3.1 FlashX

This month, we improve FlashR deployment to simplify the use of FlashR in a production environment. First, we integrate FlashR into Jupyter Notebook. As such, users can analyze large datasets stored on a server, simply using a Web browser. Figure 9 shows an example of using this environment to compute singular value decomposition on a graph with billions of vertices and generate the screeplot for singular values. In addition, we deploy FlashR in docker to enable FlashR to run in various environments easily, such as Macbook, Windows laptops and clouds.



```r
load("/mnt/nfs2/zhengda/pg-aug-40.vals")
getElbows <- function(d, n = 3, threshold = FALSE)
{
  if (is.unsorted(-d))
    stop("d must be sorted decreasingly!")

  if (!is.logical(threshold))
    d <- d[d > threshold]

  p <- length(d)
  if (p == 0)
    stop(paste("d must have elements that are larger than the threshold ",
               threshold), "!", sep="")

  lq <- rep(0.0, p)                     # log likelihood, function of q
  for (q in 1:p) {
    mu1 <- mean(d[1:q])
    mu2 <- mean(d[-(1:q)])              # = NaN when q = p
    sigma2 <- (sum((d[1:q] - mu1)^2) + sum((d[-(1:q)] - mu2)^2)) /
      (p - 1 - (q < p))
    lq[q] <- sum( dnorm(  d[1:q ], mu1, sqrt(sigma2), log=TRUE) ) +
      sum( dnorm(d[-(1:q)], mu2, sqrt(sigma2), log=TRUE) )
  }

  q <- which.max(lq)
  if (n > 1 && q < p) {
    return(c(q, q + getElbows(d[(q+1):p], n-1)))
  } else {
    return(q)
  }
}

cols <- rep("black", length(vals))
cols[getElbows(sqrt(vals))] <- "red"
pch <- rep(1, length(vals))
pch[getElbows(sqrt(vals))] <- 19
plot(sqrt(vals), type="b", xlab="#singular values", col=cols, pch=pch)
```

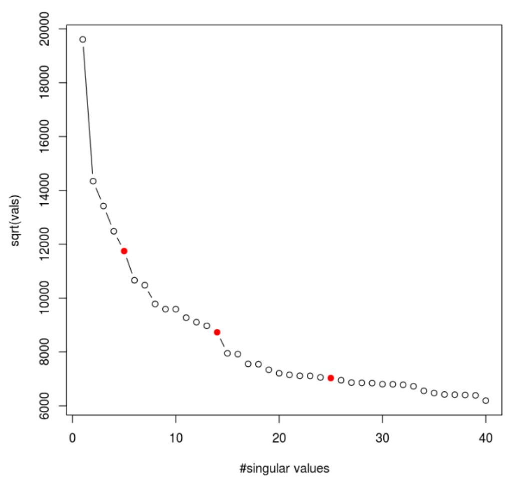


Figure 9: Generate a screeplot for singular values of a graph with billions of vertices.

3.2 knor: K-means NUMA Optimized Routines

knor is a highly optimized k-means library that performs one to two orders of magnitude better than other state-of-the-art machine learning libraries like Spark's MLlib, H_2O and Dato. As part of our commitment to developing user-friendly and easily integratable opensource tools, we developed both R and Python bindings for **knor**. We have one line-installable R package that has all the in-memory functionality of **knor**. We link the Github repo https://github.com/flashxio/knorR. We also develop Python bindings that have the same functionality as those within R. Our bindings are integrated into our main repo https://github.com/flashxio/knor/tree/dev/python. We measure the performance of our bindings relative to the native C++ and find them to compare well when we perform experiments with data on disk for our in-memory routine called **knori**. Figure 10 displays our performance. We provide base docker images to test the functionality and performance of our bindings at https://hub.docker.com/r/flashxio/knorr-base/. Figure 11 displays our pullable docker images.

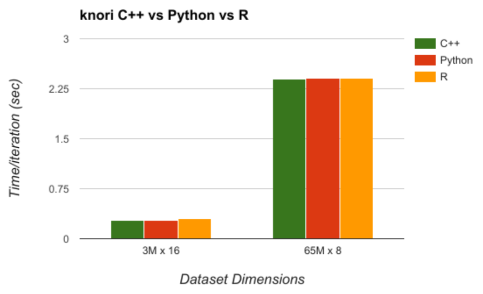
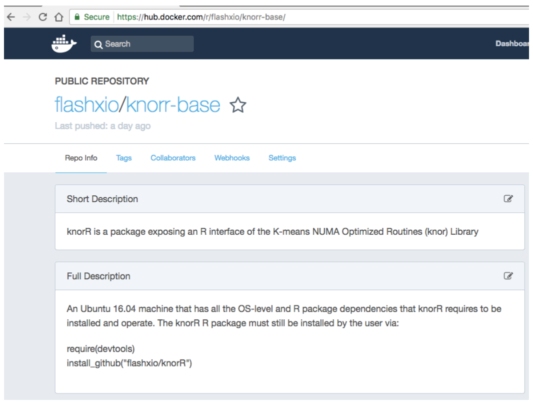


Figure 10: The performance of bindings when data are located on disk.



https://hub.docker.com/r/flashxio/knorr-base/

PUBLIC REPOSITORY

flashxio/knorr-base

Last pushed: a day ago

Short Description

knorR is a package exposing an R interface of the K-means NUMA Optimized Routines (knor) Library

Full Description

An Ubuntu 16.04 machine that has all the OS-level and R package dependencies that knorR requires to be installed and operate. The knorR R package must still be installed by the user via:

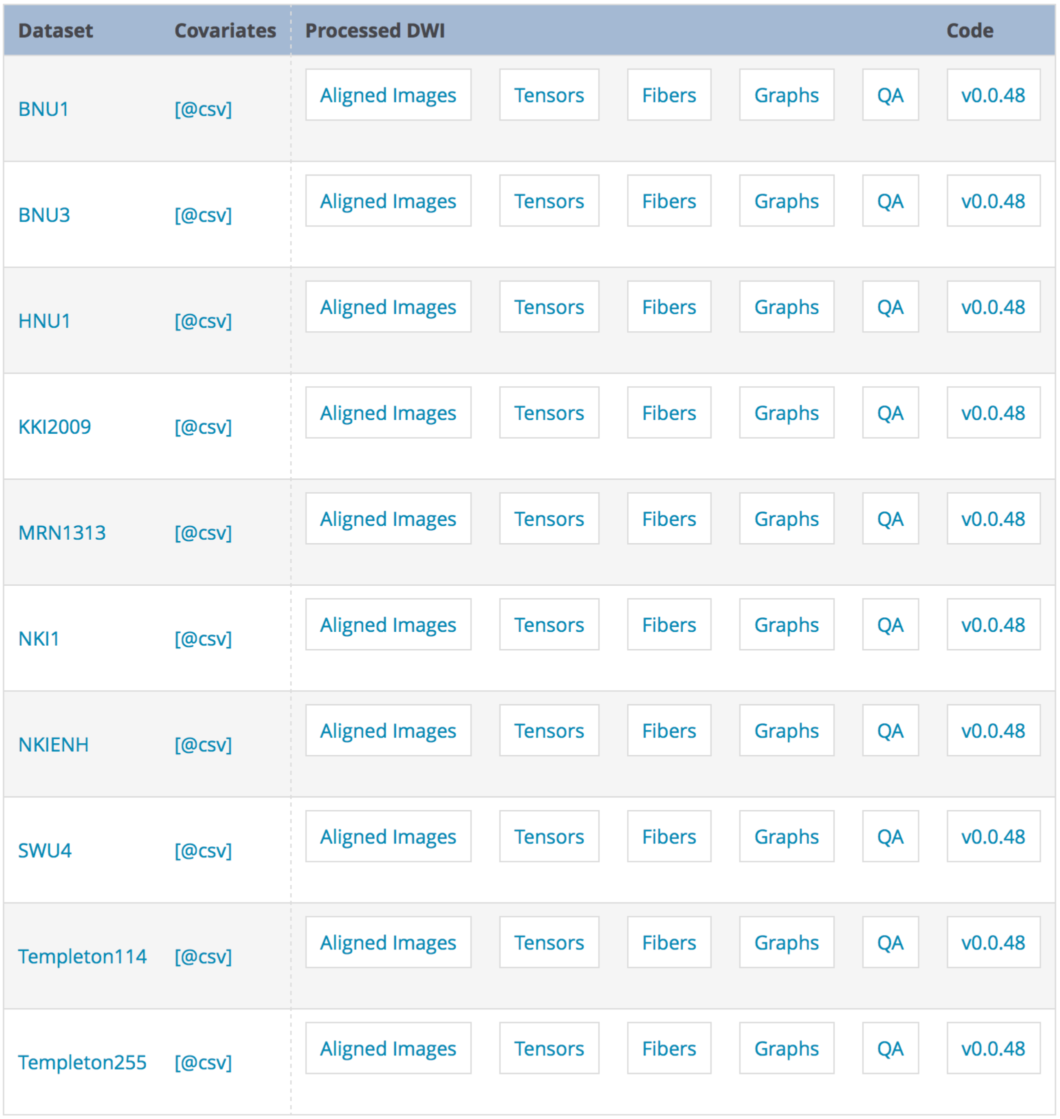
```
require(devtools)
install_github("flashxio/knorR")
```

Figure 11: The docker hub base R image.

4 Data: What's in the Cloud

We have now pushed several of our canonical datasets to the cloud, including 8 whole brain CLARITY specimens and two electron microscopy datasets: bock11, and kasthuri11:

Reference	Modality	Species	Bits	Proj	Ch	T	GV	Res	GB
Bhatla[1]	EM	C. elegans	8	3	3	1	437	6	248
Bock[2]	EM	M. musculus	8	1	1	1	20,249	11	13,312
Harris[3]	EM	R. rattus	8	3	3	1	19	4	9
Kasthuri[4]	EM	M. musculus	8	1	1	1	1,063	8	577
Lee[5]	EM	M. musculus	8	1	1	1	22,334	8	11,264
Ohyama[6]	EM	D. melanogaster	8	1	1	1	2,609	7	2,458
Takemura[7]	EM	D. melanogaster	8	1	1	1	190	5	203
Bloss[8]	AT	M. musculus	8	1	3	1	363	4	215
Collman[9]	AT	M. musculus	8	1	14	1	13	4	2
Unpublished	AT	M. musculus	16	1	24	1	29	3	23
Weiler[10]	AT	M. musculus	16	12	288	1	215	3	141
Vladimirov[11]	Ophys	D. rerio	16	1	1	100	9	4	9
Dyer[12]	XCT	M. musculus	8	1	1	1	3	3	3
Randlett[13]	LM	D. rerio	16	1	28	1	4	2	4
Kutten[14]	CL	M. musculus	16	1	23	1	7,191	6	6,727
Grabner[15]	MR	H. sapiens	16	1	3	1	<1	1	<1
Totals	–	–	–	29	349	–	47,508	–	28,441



Dataset	Covariates	Processed DWI					Code
BNU1	[@csv]	Aligned Images	Tensors	Fibers	Graphs	QA	v0.0.48
BNU3	[@csv]	Aligned Images	Tensors	Fibers	Graphs	QA	v0.0.48
HNU1	[@csv]	Aligned Images	Tensors	Fibers	Graphs	QA	v0.0.48
KKI2009	[@csv]	Aligned Images	Tensors	Fibers	Graphs	QA	v0.0.48
MRN1313	[@csv]	Aligned Images	Tensors	Fibers	Graphs	QA	v0.0.48
NKI1	[@csv]	Aligned Images	Tensors	Fibers	Graphs	QA	v0.0.48
NKIENH	[@csv]	Aligned Images	Tensors	Fibers	Graphs	QA	v0.0.48
SWU4	[@csv]	Aligned Images	Tensors	Fibers	Graphs	QA	v0.0.48
Templeton114	[@csv]	Aligned Images	Tensors	Fibers	Graphs	QA	v0.0.48
Templeton255	[@csv]	Aligned Images	Tensors	Fibers	Graphs	QA	v0.0.48

Figure 12: A current snapshot of the diffusion magnetic resonance imaging data resulting from the pipeline m2g for generating human connectomes at scale.

5 Scientific Pipelines: Infrastructure & Dataset Specific Progress

5.1 ndstore

With the release of Python 3.6 in December of last year, Python 3 has now reached sufficient maturity for us to begin migrating our infrastructure from Python 2.7 to Python 3.x. In addition to the large number of new features in the programming language that were introduced in Python 3, significant performance and stability improvements have also been introduced. We are particularly interested in using the much richer set of exceptions and increased performance for iterators present in Python 3. As a user-facing web service, NDStore must be both performant and stable, and return meaningful error messages to users when something goes wrong (e.g. bad user parameters, an internal system issue, or even a bug in the code).
To that end, we have been working with our collaborators at JHUAPL to move to a standardized Python 3 codebase for large (teravoxel) imaging volumes. We have successfully deployed a Python 3 endpoint, caching infrastructure, and some miscellaneous background processing modules (e.g for data ingest) in Amazon Web Services. We are now working to determine how best to unify our existing infrastructure with our new Python 3 infrastructure, with the overarching goal of maintaining a consistent interface across all of our assorted data types (e.g. electron and light microscopy, functional time-series imaging, magnetic resonance imaging) while minimizing duplication of effort between JHU and JHUAPL.
We have also spent considerable effort testing the data ingest client among JHU team members. We are hoping to deploy the ingest client to collaborators in the coming months. Allowing collaborators to ingest data with minimal JHU intervention increases the corpus of data we are able to collect, which provides more resources for training, running algorithms, or doing analysis.

5.2 ndviz

NeuroDataViz was rewritten to replace Leaflet entirely with Neuroglancer, an open-source web-based visualization tool from Google. Neuroglancer handles requesting data from a remote server and rendering data using WebGL. Whereas Leaflet requested 2D image tiles, Neuroglancer requests 3D image volumes. Although this results in a slight increase in request size, requesting 3D volumes allows Neuroglancer to render three canonical plane views (i.e. xy, xz, and yz) without requesting the data in triplicate as well as reducing latency when traveling in the z-direction. By replacing Leaflet with Neuroglancer we can add the same benefits to NDViz essentially “for free”.

Our first major addition to the Neuroglancer code base involved refactoring the ?SliceView? functionality to support arbitrary data types. Neuroglancer manages data loading using functionality called ?SliceView?. Tiles are requested based on the viewport and chosen plane(s) (e.g. xy). Because 3D volumes can substantially increase the data size, both data loading and data unloading become equally important. The infrastructure in SliceView manages this entire process. However, SliceView was designed solely for image data. In conjunction with Jeremy Maitin-Shepard, who developed Neuroglancer, we refactored the SliceView code to support arbitrary data formats. We then added infrastructure to manage volumes containing not image data, but point data to create ?vector graphics? layers that can display points, lines, or polylines. We are using this point layer functionality to display vector fields derived from the feature point matches between image slices (see figure).

We are planning on contributed the SliceView modifications back to Neuroglancer to share with the neuroscience community at large and are currently in the process of finalizing and integrating those changes into the main Neuroglancer repository.

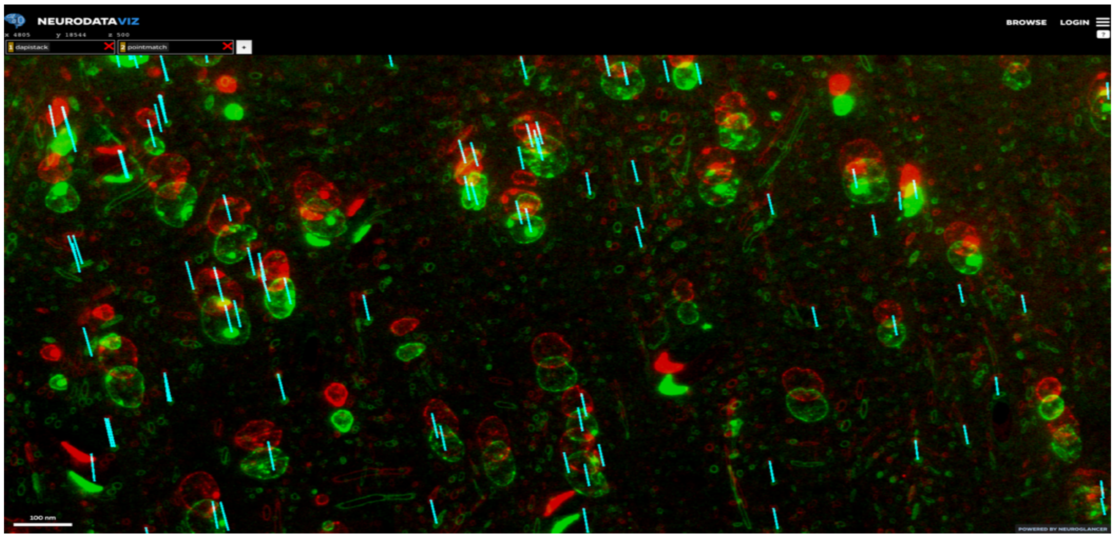


Figure 13: NeuroDataViz, powered by Neuroglancer with point matches rendered as 3D lines (in cyan) overlayed on two serial sections of Array Tomography data. Data collected by Forrest Collman et al at the Allen Insitute for Brain Science, Seattle WA.

5.3 CLARITY

5.3.1 CLARITY registration Pipeline

We tested our complete registration pipeline on a new CLARITY brain image from our Stanford collaborators. The image was acquired at a $0.585\mu m \times 0.585\mu m \times 5\mu m$ resolution using light-sheet microscopy. After sub-volume stitching, the volume was ingested into ndstore and propagated to lower resolutions. It was then downloaded at the 6th resolution level ($37.44\mu m \times 37.44\mu m \times 5\mu m$) for registration with Allen Reference Atlas (ARA). The registration pipeline began by aligning the ARA's Scanning Two-Photon (STP) tomography image using a 12-parameter affine model under Mutual Information (MI) matching. Next deformable registration was then done using MI-based Large Deformation Diffeomorphic Metric Mapping (MI-LDDMM). CLARITY-aligned annotations were generated by applying the resulting inverse transform to the ARA annotations. These annotations were uploaded into ndstore for visualization in NeuroDataViz's new Neuroglacer interface.

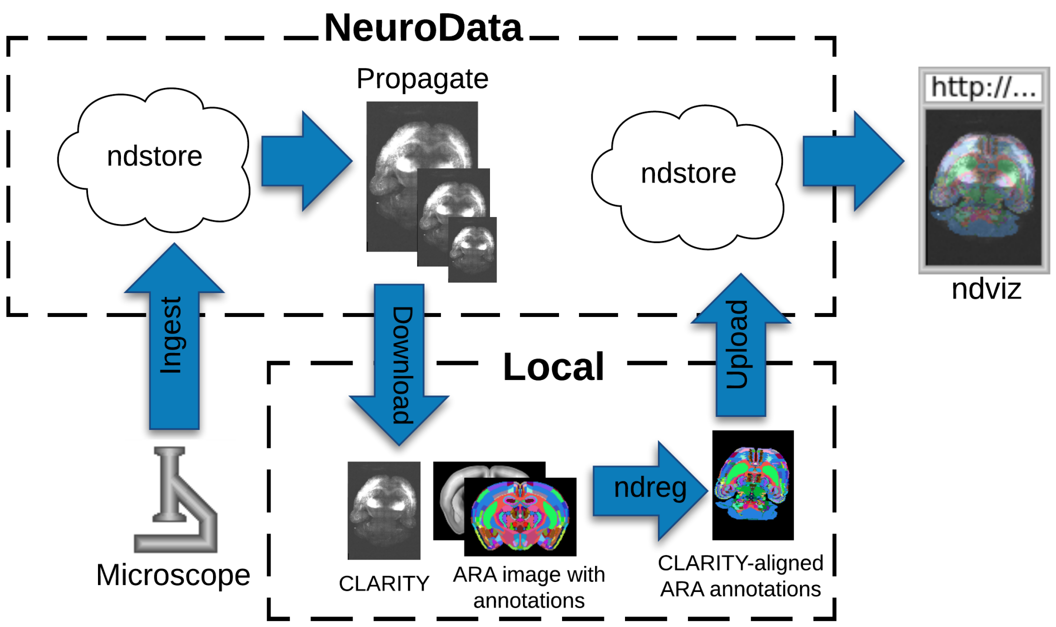


Figure 14: Workflow from microscope to visualization in NeuroDataViz (ndviz)

5.4 fngs

Through the fngs pipeline, we develope a robust processing pipeline and web service for providing automated acquisition of functional connectomes from structural and functional MRI. The fngs pipeline was developed around the glass-box principle; each step of the pipeline produces intuitive and descriptive quality assurance so users can be confident that the internals of the pipeline are performing properly. We provide an open-source docker container, links to all code, and have numerous tutorials and demos available Tutorials for users to receive a step-by-step introduction to the pipeline. Moreover, we provide an interactive schematic of the pipeline Schematic. Note that the headings in each box at the bottom link to the documentation for the respective steps of the pipeline. Using the one-click cloud deployment of fngs, we were able to analyze 1200 human connectomes in 7 hours for a total of $80. The cloud deployment procedure of the fngs pipeline can be seen in Figure (15).

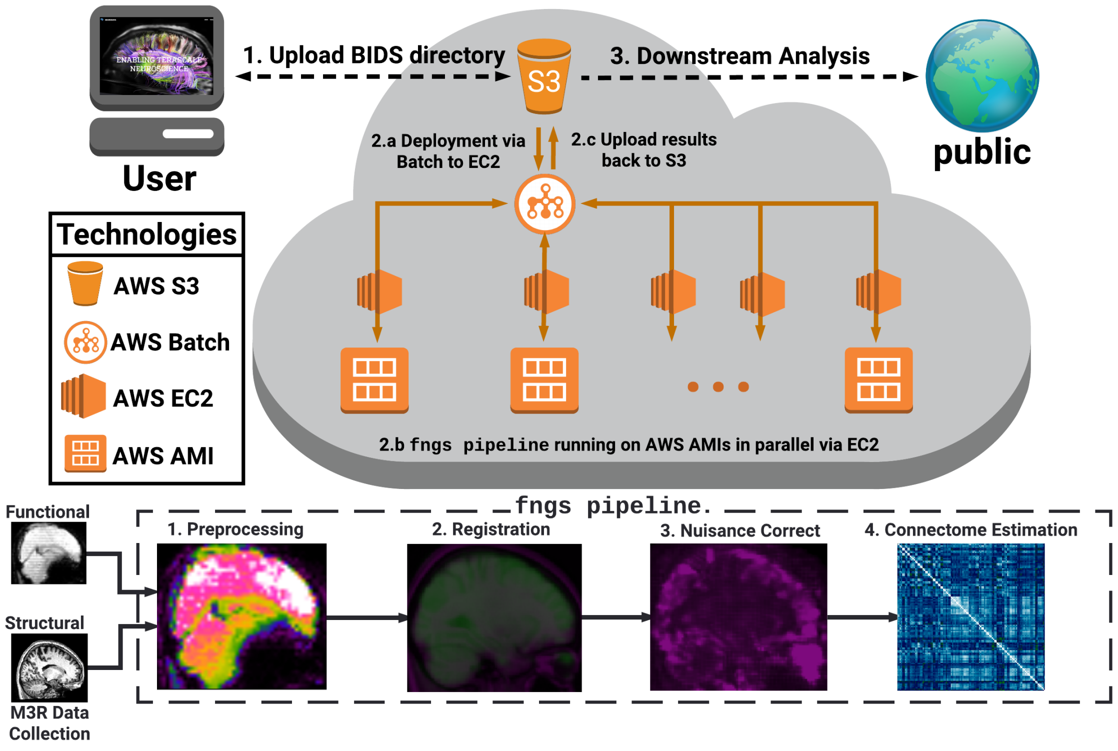


Figure 15: **FNGS workflow**. The deployment workflow for the FNGS pipeline. Users provide a directory according to the BIDs spec to the FNGS cloud controller locally via the docker container at (1). The data is uploaded directly to AWS S3 cloud drives. The controller then initiates the Batch deployment procedure at (2.a), which interfaces between S3 and EC2 cloud computers to provide the MRI scans to EC2 instances pre-loaded with the FNGS pipeline for analysis. After the scans are finished being analyzed on the EC2 instances at (2.b), the results are then re-uploaded back to the S3 cloud drive at (2.c). The user can then navigate to the S3 cloud drive for downstream analyses.

6 Reference Datasets

References

[1] N. Bhatla, R. Droste, S. R. Sando, A. Huang, and H. R. Horvitz, "Distinct neural circuits control rhythm inhibition and spitting by the myogenic pharynx of c. elegans," Current Biology, vol. 25, no. 16, pp. 2075 – 2089, 2015. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0960982215007460

[2] D. D. Bock, W.-C. A. Lee, A. M. Kerlin, M. L. Andermann, G. Hood, A. W. Wetzel, S. Yurgenson, E. R. Soucy, H. S. Kim, and R. C. Reid, "Network anatomy and in vivo physiology of visual cortical neurons," Nature, vol. 471, no. 7337, pp. 177–182, 03 2011.

[3] K. M. Harris, J. Spacek, M. E. Bell, P. H. Parker, L. F. Lindsey, A. D. Baden, J. T. Vogelstein, and R. Burns, "A resource from 3D electron microscopy of hippocampal neuropil for user training and tool development," Scientific Data, vol. 2, p. 150046, Aug 2015.

[4] N. Kasthuri, K. J. Hayworth, D. R. Berger, R. L. Schalek, J. Conchello, S. Knowles-Barley, D. Lee, A. Vázquez-Reina, V. Kaynig, T. R. Jones, M. Roberts, J. L. Morgan, J. C. Tapia, H. S. Seung, W. G. Roncal, J. T. Vogelstein, R. Burns, D. L. Sussman, C. E. Priebe, H. Pfister, and J. W. Lichtman, "Saturated reconstruction of a volume of neocortex," Cell, vol. 162, pp. 648–661, 05 2016. [Online]. Available: http://dx.doi.org/10.1016/j.cell.2015.06.054

[5] W.-c. A. Lee, V. Bonin, M. Reed, B. J. Graham, G. Hood, K. Glattfelder, and R. C. Reid, "the visual cortex," Nature, vol. 532, no. 7599, pp. 370–374, 2016. [Online]. Available: http://dx.doi.org/10.1038/nature17192

[6] T. Ohyama, C. M. Schneider-Mizell, R. D. Fetter, J. V. Aleman, R. Franconville, M. Rivera-Alba, B. D. Mensh, K. M. Branson, J. H. Simpson, J. W. Truman, A. Cardona, and M. Zlatic, "A multilevel multimodal circuit enhances action selection in drosophila," Nature, vol. 520, no. 7549, pp. 633–639, 04 2015.

[7] S.-y. Takemura, A. Bharioke, Z. Lu, A. Nern, S. Vitaladevuni, P. K. Rivlin, W. T. Katz, D. J. Olbris, S. M. Plaza, P. Winston, T. Zhao, J. A. Horne, R. D. Fetter, S. Takemura, K. Blazek, L.-A. Chang, O. Ogundeyi, M. A. Saunders, V. Shapiro, C. Sigmund, G. M. Rubin, L. K. Scheffer, I. A. Meinertzhagen, and D. B. Chklovskii, "A visual motion detection circuit suggested by drosophila connectomics," Nature, vol. 500, no. 7461, pp. 175–181, 08 2013. [Online]. Available: http://dx.doi.org/10.1038/nature12450

[8] E. B. Bloss, M. S. Cembrowski, B. Karsh, J. Colonell, R. D. Fetter, and N. Spruston, "Structured dendritic inhibition supports branch-selective integration in ca1 pyramidal cells," Neuron, vol. 89, no. 5, pp. 1016 – 1030, 2016. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0896627316000544

[9] F. Collman, J. Buchanan, K. D. Phend, K. D. Micheva, R. J. Weinberg, and S. J. Smith, "Mapping synapses by conjugate light-electron array tomography," The Journal of Neuroscience, vol. 35, no. 14, pp. 5792–5807, 2015.

[10] N. C. Weiler, F. Collman, J. T. Vogelstein, R. Burns, and S. J. Smith, "Molecular architecture of barrel column synapses following experience-dependent plasticity." Nature Scientific Data, 2014.

[11] J. Freeman, N. Vladimirov, T. Kawashima, Y. Mu, N. J. Sofroniew, D. V. Bennett, J. Rosen, C.-T. Yang, L. L. Looger, and M. B. Ahrens, "Mapping brain activity at scale with cluster computing," Nature Methods, no. July, jul 2014. [Online]. Available: http://www.nature.com/doifinder/10.1038/nmeth.3041

[12] E. L. Dyer, W. Gray Roncal, H. L. Fernandes, D. Gürsoy, X. Xiao, J. T. Vogelstein, C. Jacobsen, K. P. Körding, and N. Kasthuri, "Quantifying mesoscale neuroanatomy using x-ray microtomography," "arXiv", "2016".

[13] O. Randlett, C. L. Wee, E. A. Naumann, O. Nnaemeka, D. Schoppik, J. E. Fitzgerald, R. Portugues, A. M. B. Lacoste, C. Riegler, F. Engert, and A. F. Schier, "Whole-brain activity mapping onto a zebrafish brain atlas," Nat Meth, vol. 12, no. 11, pp. 1039–1046, 11 2015.

[14] K. S. Kutten, J. T. Vogelstein, N. Charon, L. Ye, K. Deisseroth, and M. I. Miller, "Deformably Registering and Annotating Whole CLARITY Brains to an Atlas via Masked LDDMM," in Proceedings SPIE 9896, Optics, Photonics and Digital Technologies for Imaging Applications IV, P. Schelkens, T. Ebrahimi, G. Cristóbal, F. Truchetet, and P. Saarikko, Eds., 2016.

[15] G. Grabner, A. L. Janke, M. M. Budge, D. Smith, J. C. Pruessner, and D. L. Collins, "Symmetric atlasing and model based segmentation: An application to the hippocampus in older adults," in Medical Image Computing and Computer-Assisted Intervention - MICCAI 2006, 2006, pp. 58–66.