Statistical Exploration of Electronic Structure of Molecules from Quantum Monte-Carlo Simulations

Prabhat

Computational Research Division

Lawrence Berkeley National Laboratory

Berkeley, CA 94720

Dmitry Zubarev

Department of Chemistry

University of California, Berkeley

Berkeley, CA 94720

William A. Lester, Jr.

Department of Chemistry

University of California, Berkeley

Berkeley, CA 94720

Introduction[*]

In this report, we present results from analysis of Quantum Monte Carlo (QMC) simulation data with the goal of determining internal structure of a 3N-dimensional phase space of an N-electron molecule. We are interested in mining the simulation data for patterns that might be indicative of the bond rearrangement as molecules change electronic states. We examined simulation output that tracks the positions of two coupled electrons in the singlet and triplet states of an H2 molecule. The electrons trace out a trajectory, which was analyzed with a number of statistical techniques.

We propose the following high-level questions for our investigation:

- Do high-dimensional phase spaces characterizing electronic structure of molecules tend to cluster in any natural way? Do we see a change in clustering patterns as we explore different electronic states of the same molecule?
- Since it is hard to understand the high-dimensional space of trajectories, can we project these trajectories to a lower dimensional subspace to gain a better understanding of patterns?
- Do trajectories inherently lie in a lower-dimensional manifold? Can we recover that manifold?

Dataset description:

We analyzed two datasets in this project. One is labeled "H2" and the other "H2tri", which is an H2 molecule in bonded singlet and non-bonded triplet states respectively. Both systems belong to the $D_{\infty h}$ point group. Therefore, the 6-dimensional phase space was mapped onto a 4-dimensional hyper-surface. The datasets contain position information for 2 electrons (x1,y1) and (x2,y2). A derived quantity called "scalar", is computed as the distance of the electron pair to their mid-point. Both datasets have 200K timesteps; we therefore have 1M data values.

[*] **Disclaimer** This document was prepared as an account of work sponsored by the United States Government. While this document is believed to contain correct information, neither the United States Government nor any agency thereof, nor The Regents of the University of California, nor any of their employees, makes any warranty, express or implied, or assumes any legal responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by its trade name, trademark, manufacturer, or otherwise, does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or any agency thereof, or The Regents of the University of California. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or any agency thereof or The Regents of the University of California.

Hardware and Software used:

All of our analysis was conducted using existing libraries in R and MATLAB. We used VisIt to perform interactive exploration of our results and to generate movies. We used a high-end Linux workstation (4 quad-core Opterons, 32GB memory) and a MacBook Pro for all of our tests. We also wrote custom C++ code to split the time-series into frames and perform temporal tracking of clusters. We also wrote modules to import R clustering results into VisIt.

Approaches:

In order to address our research questions, we used the following techniques:

1) Clustering:
    a. K-means
    b. Partition around medoids
    c. Gap Method
    d. Model Based clustering
    e. Hierarchical clustering

2) Dimensionality reduction:
    a. PCA
    b. Kernel PCA
        i. Vanilladot
        ii. RBF
        iii. Polynomial
    c. MDS

3) Manifold learning:
    a. LLE/Hessian Eigenmaps
    b. IsoMAP

Exploratory Visualization

Initially, we spent a significant amount of time exploring and getting a sense of the data. While R has very powerful analysis capabilities, its interactive visualization capabilities are limited, and we ended up using VisIt (https://wci.llnl.gov/codes/visit/) for this task. This was particularly important for exploring individual clusters and their multi-dimensional distributions.

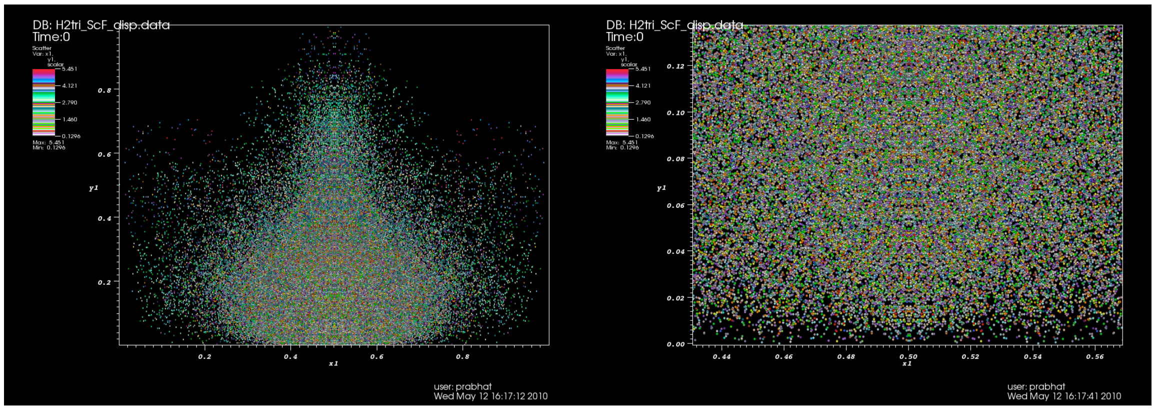


Figure 1: An overview plot of H2tri dataset colored by scalar value (left) and a zoomed in view (right).

Apart from standard pair wise plots for the dataframe in R; we used pseudocolor plots in VisIt. We were able to make background plots (indicating all particles), and highlight specific clusters. We also used the parallel co-ordinates plot with much success. This was important for isolating a cluster of interest, and examining the other dimensions (spatial and scalar) for investigation. We also used VisIt to generate movies from clustering results for timeframes.

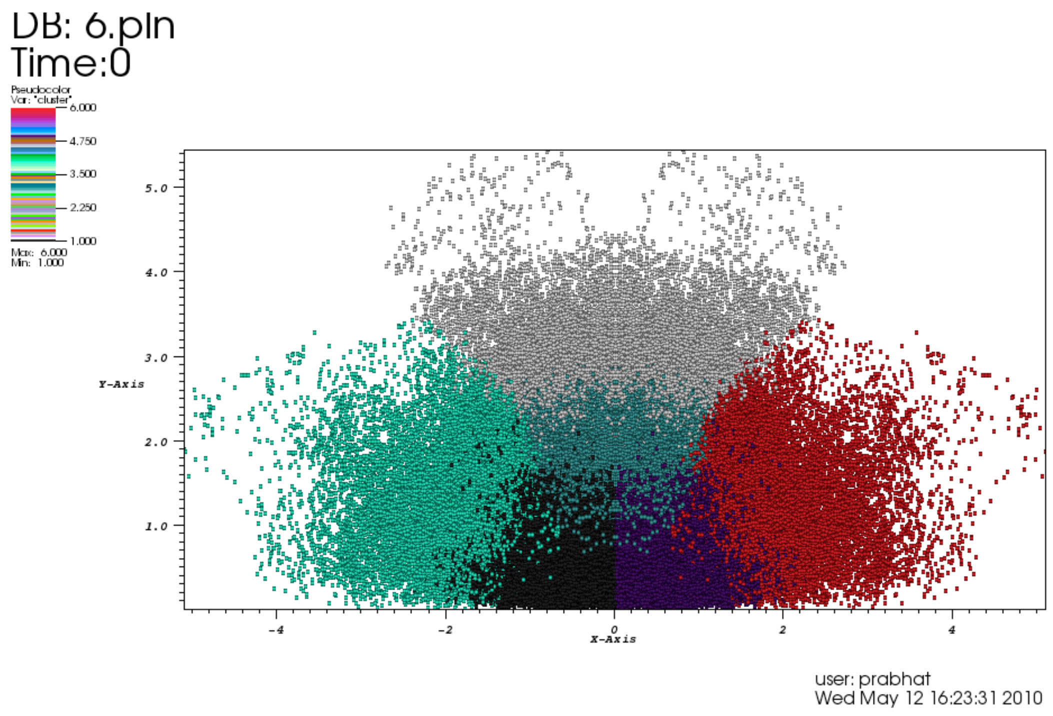


Figure 2: Examining clustering results from K-means procedure. Note how other clusters obscure the central teal-colored cluster.

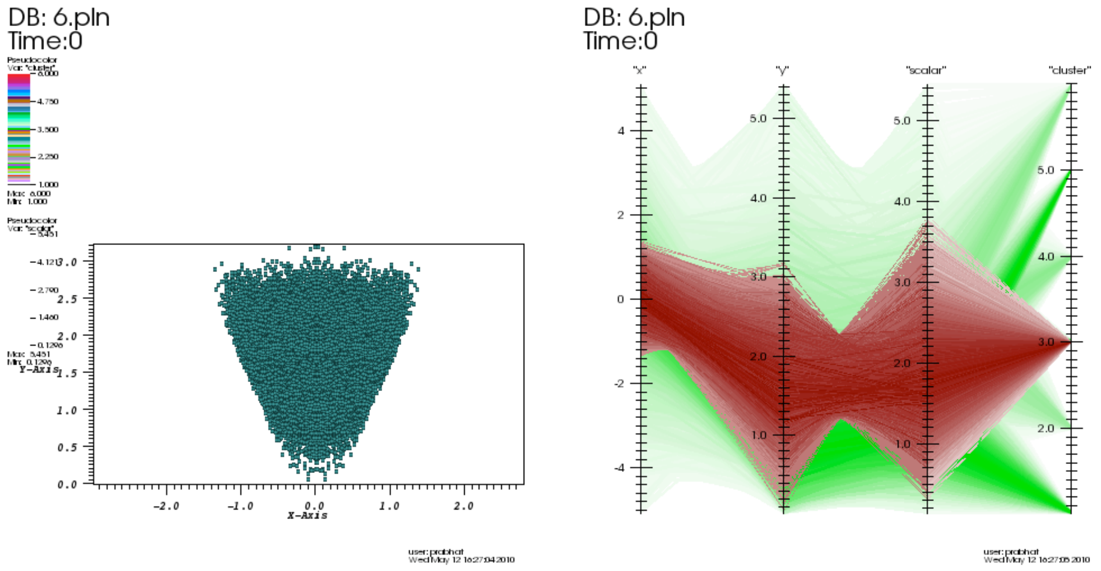


Figure 3: The parallel co-ordinates plot, combined with selection operators assist in isolating the cluster of interest and examining its multi-dimensional distribution.

Clustering Techniques

K-means

To begin our clustering investigation, we applied k-means clustering to all 200K points of the H2 and H2tri datasets using the R kmeans() function. We specified a range of 2 through 30 for the initial exploration. The k-means procedure returns a withinss (within cluster sum of squares) measure, which is an indication of how similar the cluster elements are to each other. We plot this in the figure below.

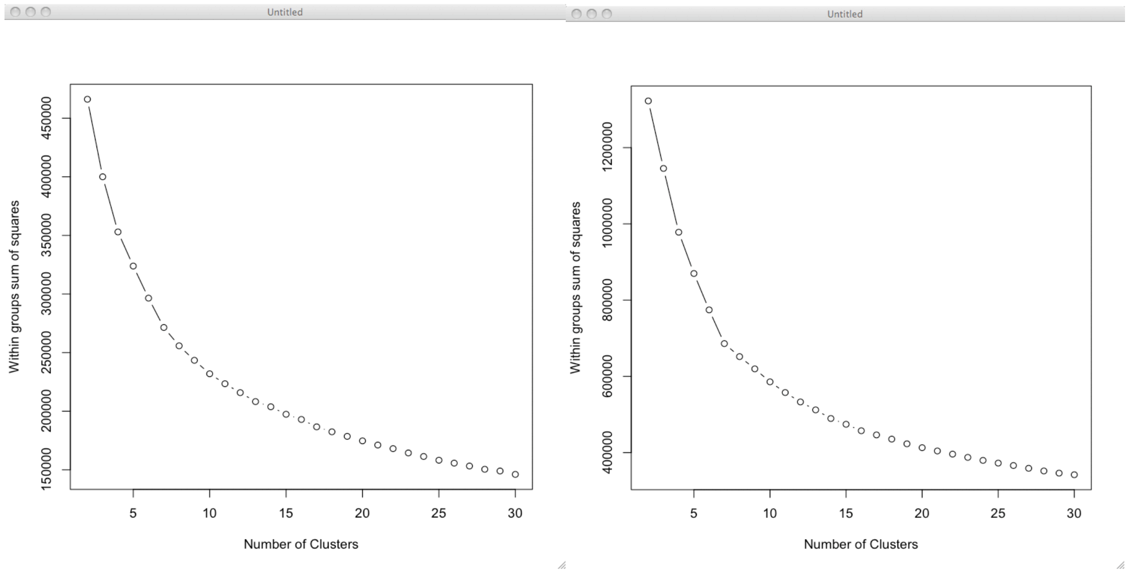


Figure 4: Withinss measure for H2 (left) and H2tri (right).

Qualitatively, both plots for H2 and H2tri are similar, and they seem to indicate that the right number of clusters might be between 2 and 10 (the knee of the curve appears to lie in that range). Several of the computationally expensive clustering procedures in the following sections require us to supply a range of clusters to explore, and we use this range as a baseline. We note that the withinss measure will never reduce to 0 (unless one has 200K clusters!), and beyond some number the falloff is gradual; hence this interpretation is somewhat subjective.

At this point in time, we don't know what the right value should be for the K-means procedure (and if there's a difference across datasets), so we chose the same value for both datasets and examine the results.

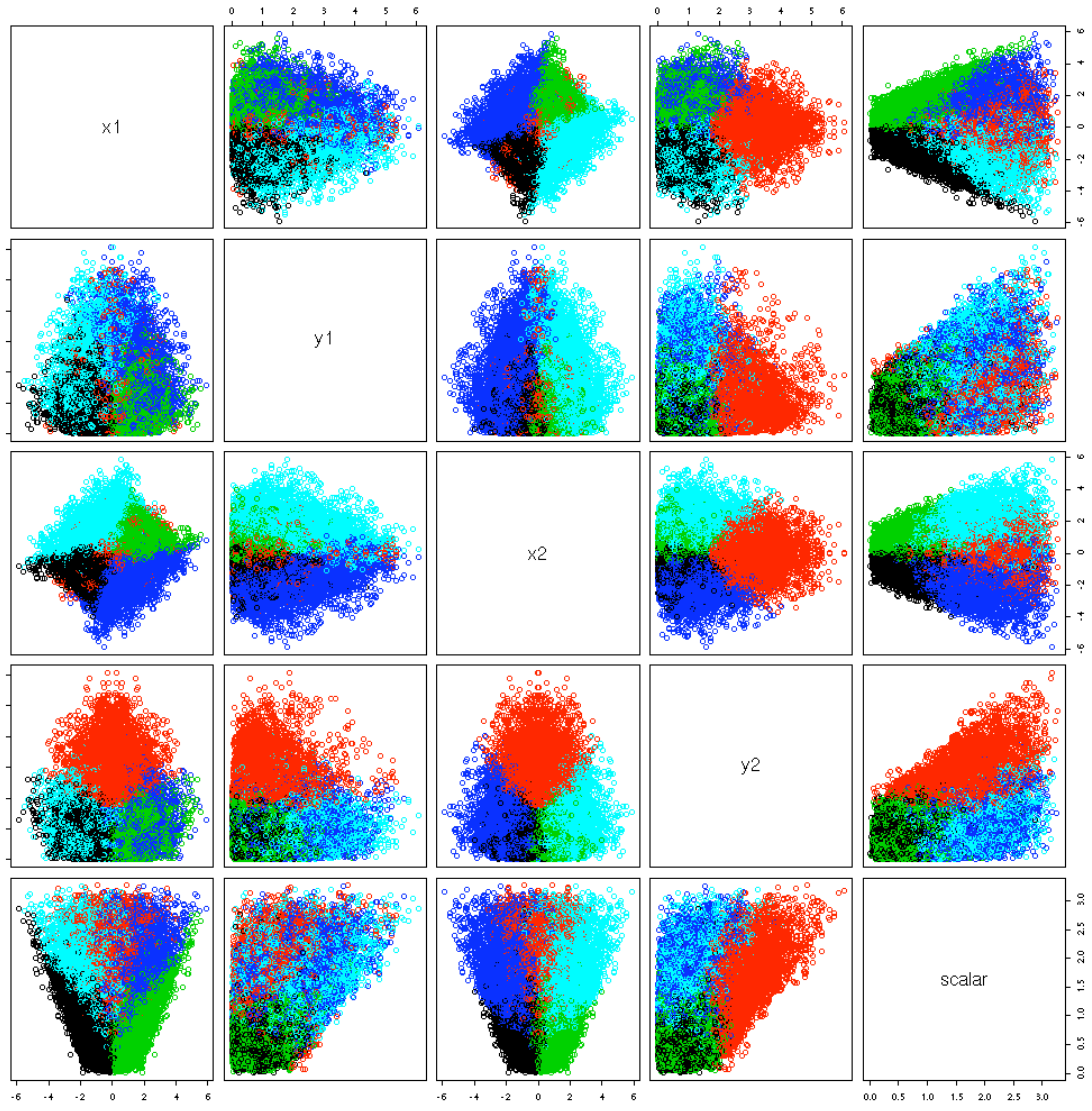


Figure 5: K-means clustering results for H2 for k=5

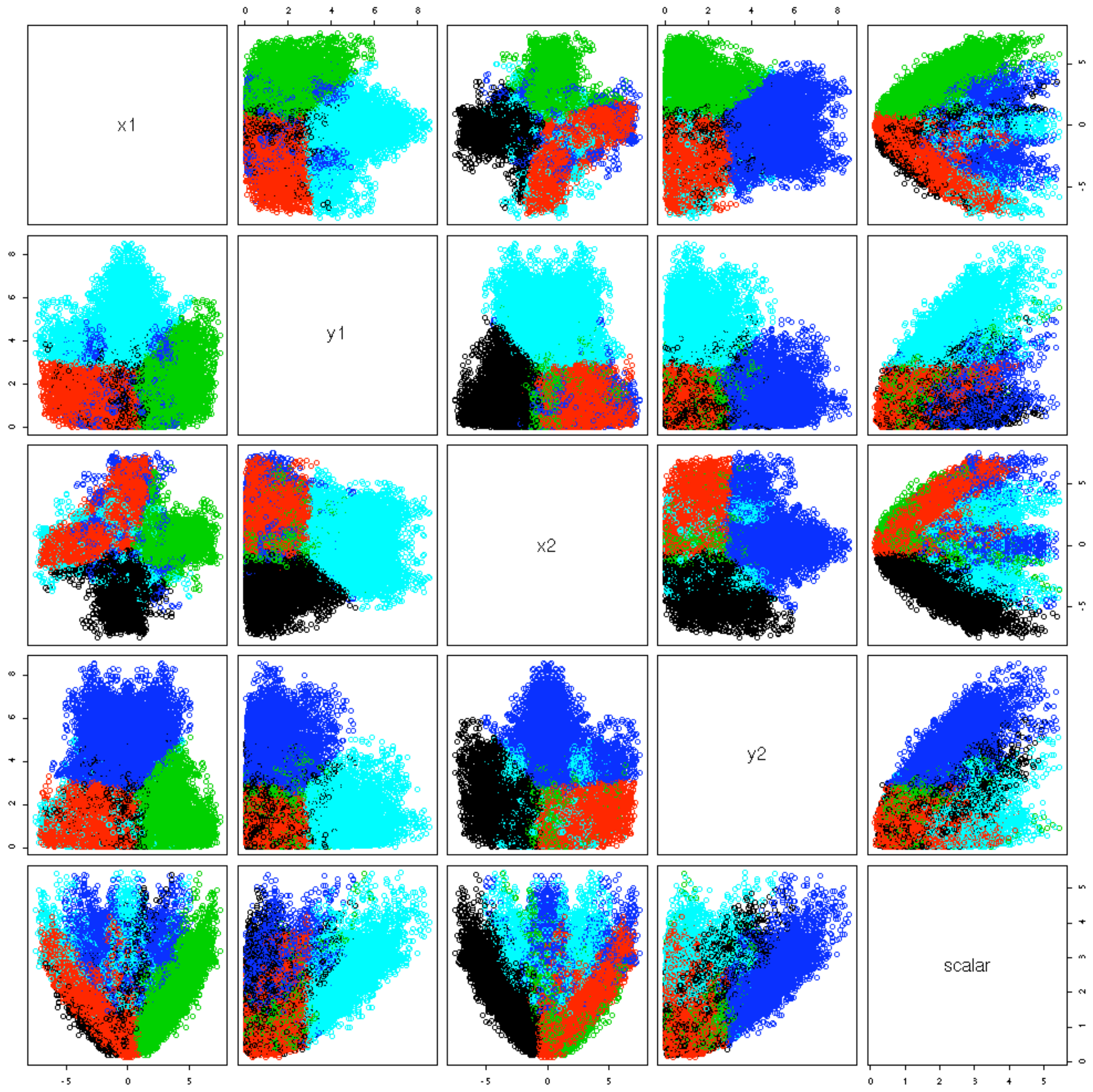


Figure 6: K-means clustering results for H2tri, for k=5.

We observe that for both datasets, we don't get a clean separation of the clusters. The imposed cluster boundaries look somewhat artificial and are an artefact of the multi-dimensional voronoi tessellation.

Subsequently, we tried splitting the dataset into multiple timesteps (with overlapping windows) and performing k-means on individual timesteps. This approach was intended to examine how the clustering would vary across timesteps.

We came across an interesting problem while visualizing results of temporal clustering. Typical clustering algorithms (like k-means) don't associate any importance to the cluster label, i.e. a label of "1" in timestep 10 is just as good as label "2" in timestep 11. If you perform k-means clustering across time-frames, each time frame is processed independently of another. This is problematic for visualization purposes, because one typically presents clusters with colors, and if one naively presents temporal clustering results, the distraction of colors changing every frame (even for the same cluster) overwhelms all other cues. In order to alleviate this problem, we devised and implemented the following post-processing technique: assuming that we have the same number of clusters in each time-frame, we compute pair-wise distances between clusters in the current frame and the next frame; sort the distances and assign a target cluster in the next frame to the closest source cluster in the current frame. Once this assignment is done, remove all instances of the target and source cluster from the sorted list, and repeat till all clusters in the next frame have a match. This seems to work quite well, and is demonstrated in some movies listed in the K-means section below:

http://vis.lbl.gov/~prabhat/MolVis/Results/H2-kmeans-8.mpg

http://vis.lbl.gov/~prabhat/MolVis/Results/H2tri-kmeans-8.mpg

http://vis.lbl.gov/~prabhat/MolVis/Results/H2-scalar-kmeans-8.mpg

http://vis.lbl.gov/~prabhat/MolVis/Results/H2tri-scalar-kmeans-5.mpg

The “scalar” version of the movies uses the “scalar” field alone for performing the k-means clustering.

In general, the temporal results of the k-means clustering were interesting: we observed an interesting odd/even split about the y-axis for the datasets (mostly even for H2 and odd for H2tri).

Partition across medoids (PAM) technique

After exploring the K-means method, we wanted to test some other clustering techniques. The PAM technique (implemented with the pamk methods in R fpc package) tries to optimize silhouette width for a given range of clusters. This seemed like a promising approach, however when we tried PAM on the complete 200K dataset, the technique failed to run, complaining of too many entries. We therefore tried two lines of attack (which we have used throughout our report):

- Sampling 25K points from the entire dataset and running pamk
- Splitting the dataset into frames (or windows). We set the size of the window to 20K points and randomly sample 5K points. We then "slide" the window by 5K, index into the next 20K points, and sample 5K points for the next frame. The specific choices of window size, window offset and number of samples have been used in the past for similar analysis and datasets.

It is important to note that our windows are overlapping. This provides some degree of robustness for our procedures.

Testing on 25K samples of the entire dataset

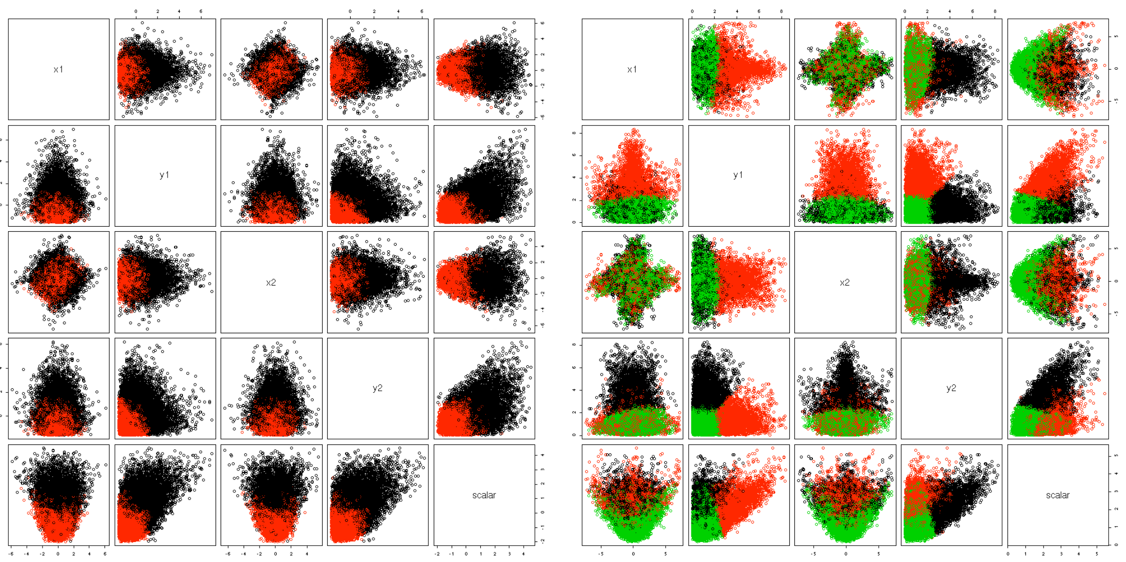


Figure 7: pamk results for the H2 (left) and H2tri (right) datasets.

Running pamk() on a random sample of 25K points reveals the optimal number of clusters as 2 (for H2), and 3 (for H2tri). These cluster assignments are shown in Figure 7. These results led us to investigate the following question: "What would happen if we ran pamk() on the scalar field alone?" We tried that and obtained the following result.

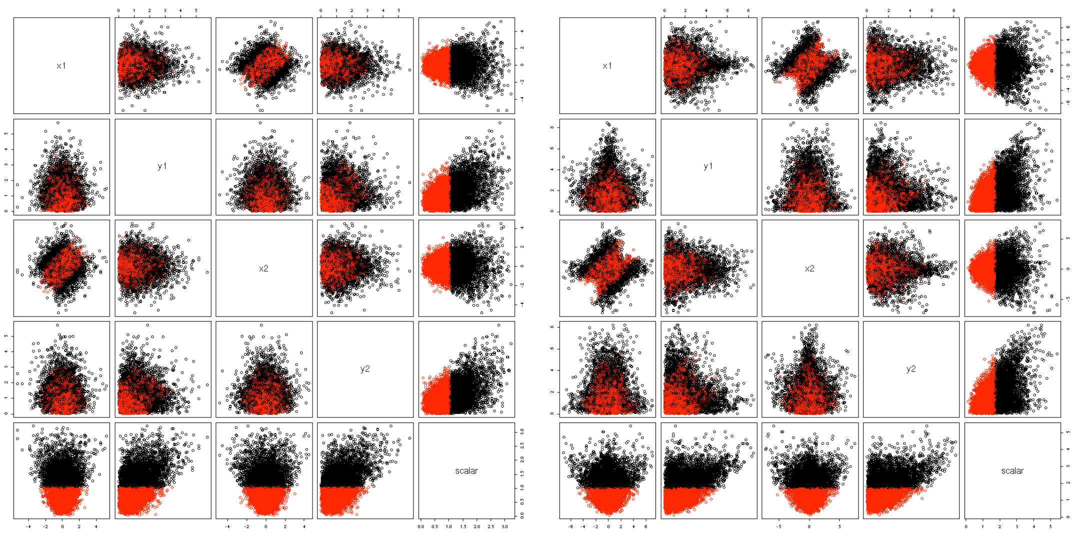


Figure 8: pamk results for the H2 (left) and H2tri (right) timeseries.

Figure 8 shows 2 clusters are returned for both H2 and H2tri. In retrospect, this was not a smart option to try out since scalar is just a 1D quantity, and pamk() partitions around mid-point of the distribution. We see this with the vertical and horizontal lines in the last row/column of the above figures.

Testing on the Time-series

Next, we applied pamk() clustering to each window frame of both datasets. The following figures describe our results.

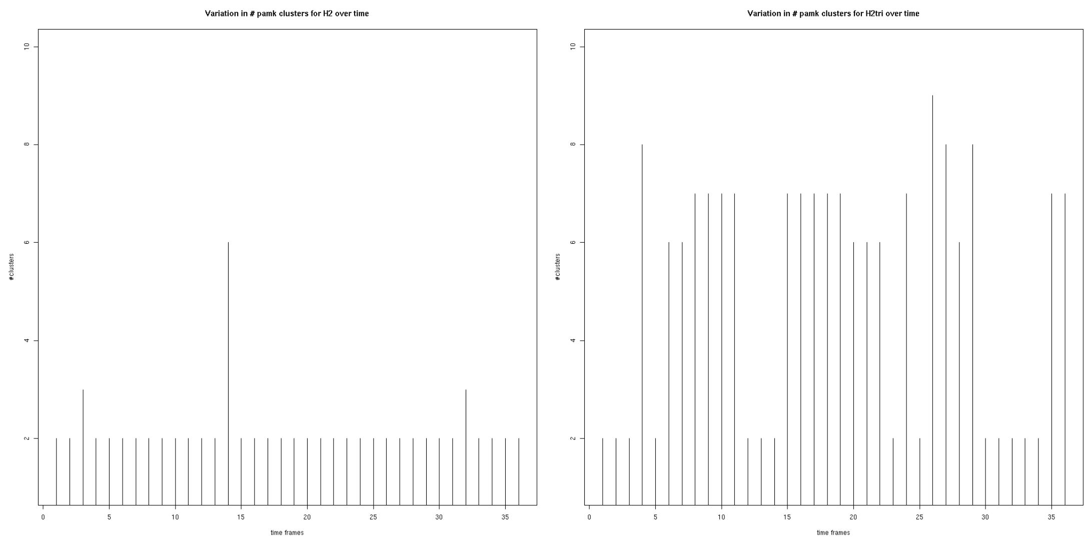


Figure 9: Number of optimal pamk clusters for H2 (left) and H2tri (right) over time frames.

We observe that the procedure is able to distinguish between the 2 datasets; it returns 2 clusters (most of the time) for H2, and roughly 7 clusters for H2tri. This is a promising result.

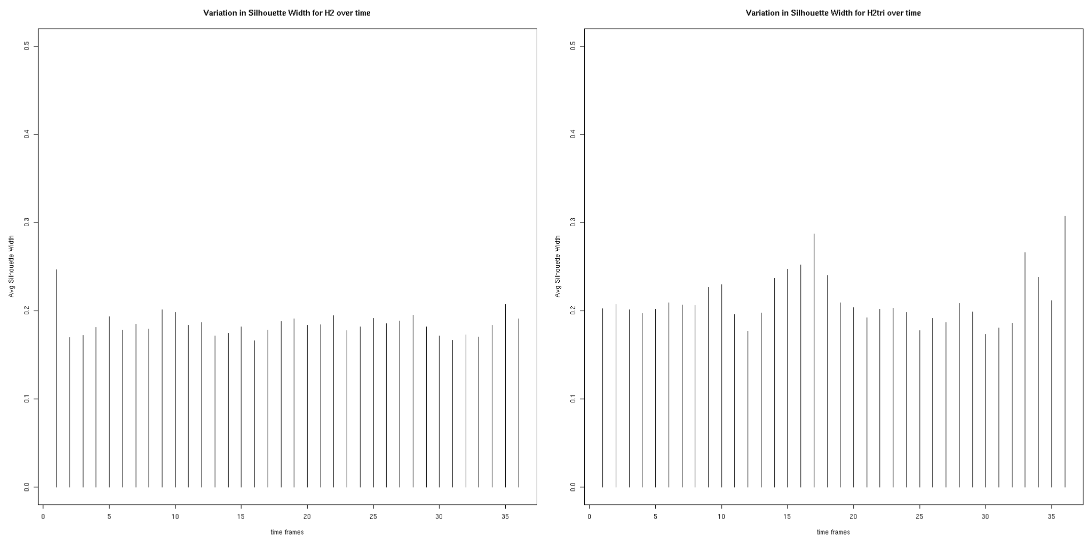


Figure 10: Silhouette width for H2 (left) and H2tri (right).

Generally we observe similar silhouette width across the entire time-series; which seems to indicate that the clustering is stable. The absolute magnitude of the silhouette width is comparable across the datasets.

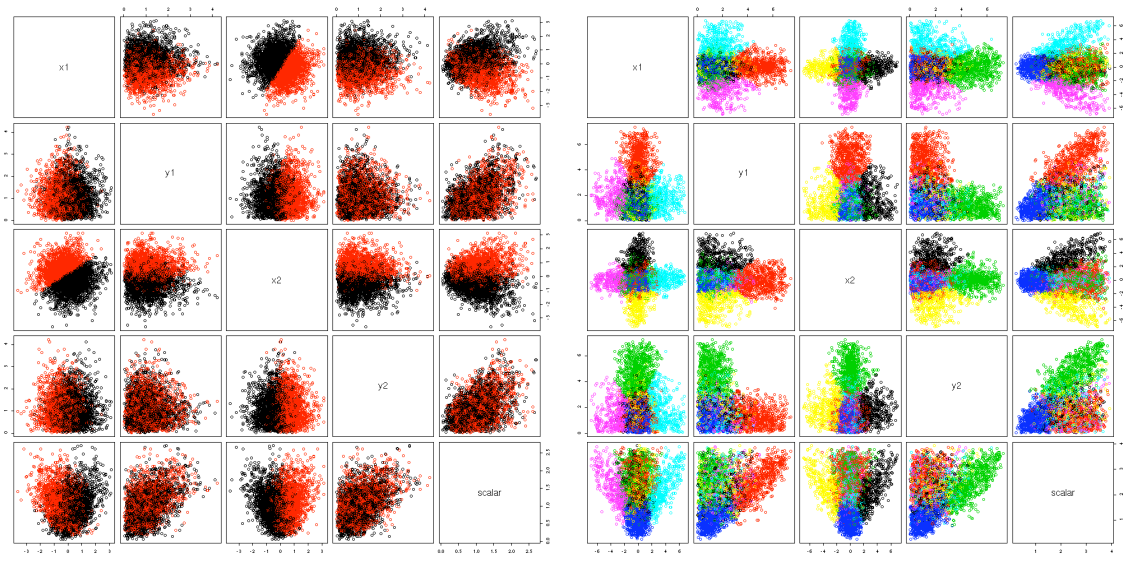


Figure 11: Representative frames with high silhouette widths. Time-frame #8 for H2 with 2 clusters is plotted on the left. Time-frame #17 for H2tri with 7 clusters is on the right.

Qualitatively, we see a dramatic difference in the shape of the clusters. While the 2 clusters for H2 are symmetric across the y-axis, the 7 clusters for H2tri seem to be localized in different regions. Looking at the (x1,y1) plot, the region in bottom center has multiple overlapping clusters, which is interesting.

Pamk clustering for Time-series without the scalar value

Upon examining at the pamk() clustering results, we next conducted the analysis for the spatial dimensions alone (x1,y1,x2,y2) without the scalar value. We now present the results of this analysis.

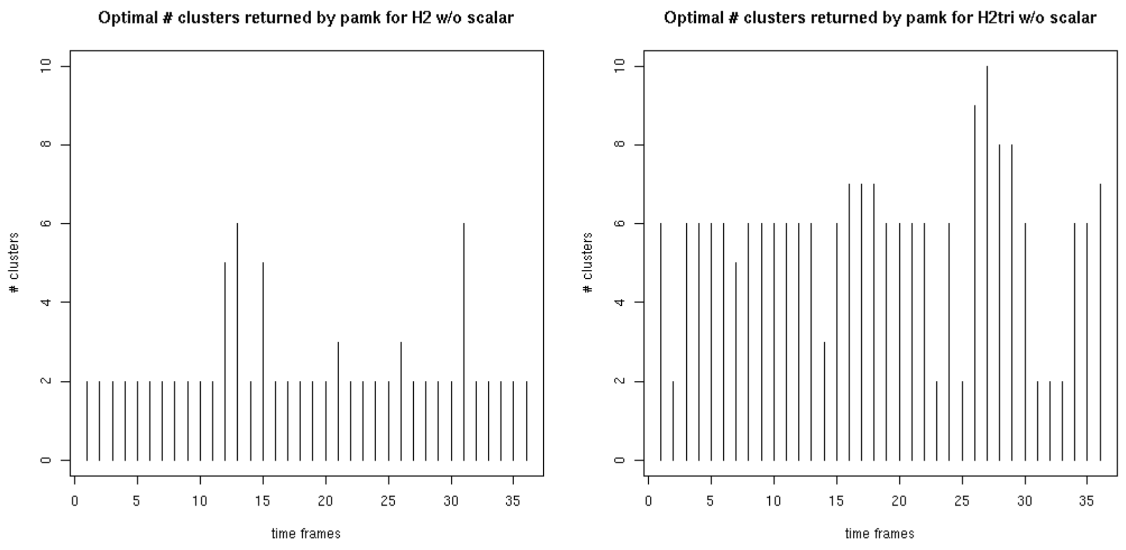


Figure 12: pamk results for time-series. H2 is on the left and H2tri on the right.

Similar to the earlier results for the entire dataset, we seem roughly 2 clusters being returned for H2, and roughly 6 clusters being returned for H2tri.

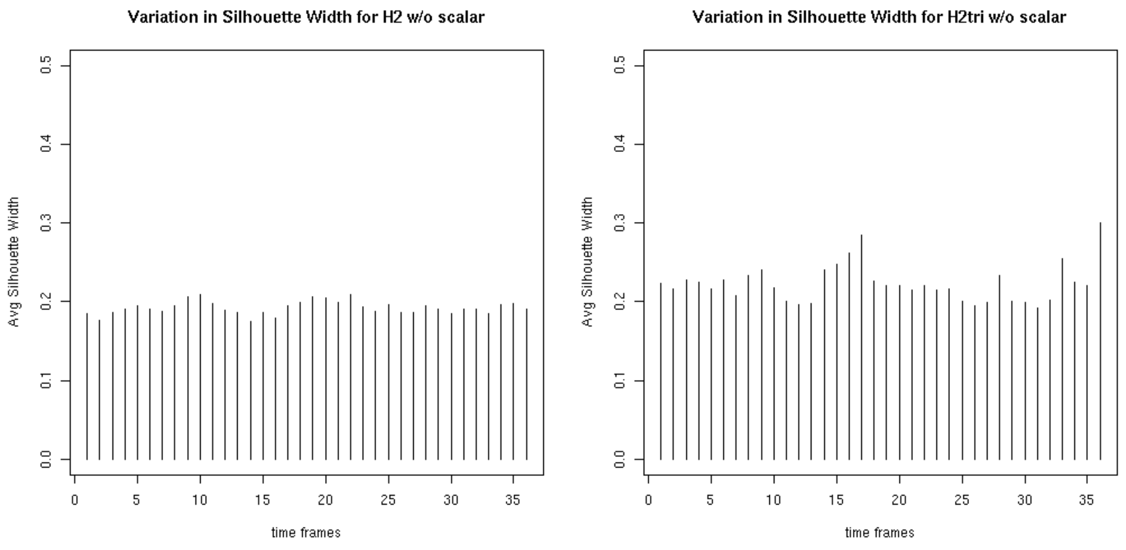


Figure 13: Silhouette Width for H2 (left) and H2tri (right).

The silhouette width also remains roughly same across the time frames, which indicates robustness to our choice of sampling parameters.

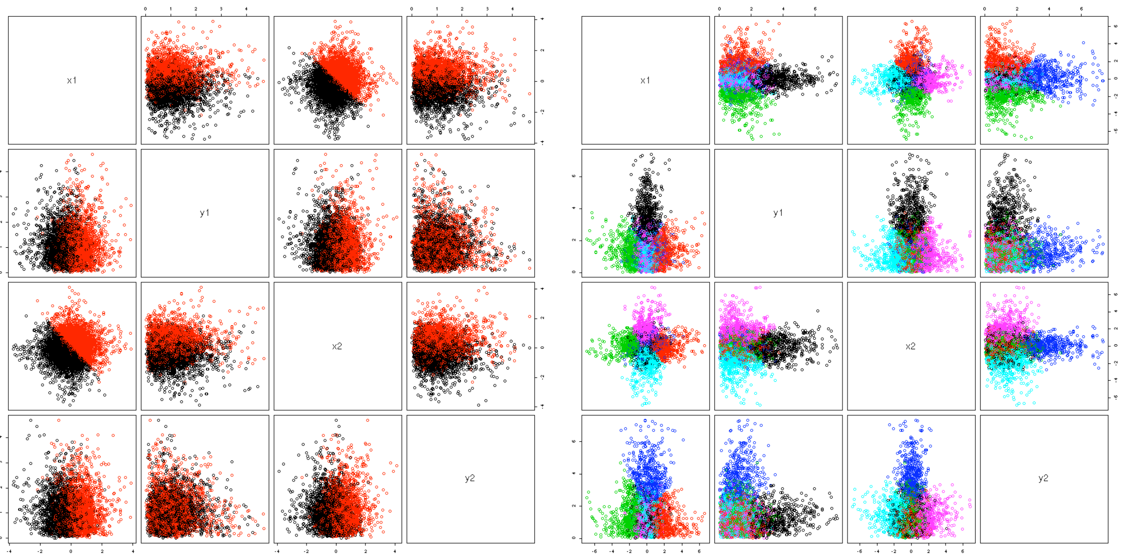


Figure 14: Representative frames. Timestep 20 for H2 is plotted on left. Timestep 20 for H2tri is plotted on right.

Again, we see a symmetric distribution about the y-axis for the left figure, and clusters of a very different shape on the right.

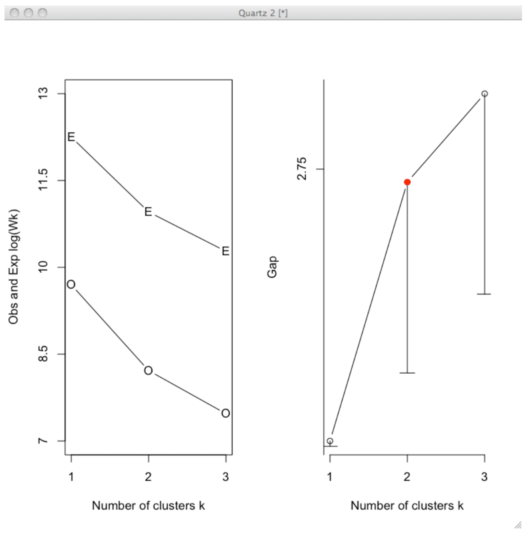
Gap Method

Next, we applied another technique: the Gap statistic (from the R lga library), for returning the optimal number of clusters. The gap() call takes either the tibshirani or the DandF criteria, and uses Linear Grouping Analysis to compute the number of clusters.

Initially, both techniques (tibshirani and DandF criteria) failed to run on the entire dataset. Even choosing a subset of 25K point failed, the lga() procedure complained that it could not converge in the default (20) number of iterations. We overwrote the default implementations of these procedures to increase the number of iterations to 500, after which we did not encounter convergence warnings.

Running with the DandF criteria failed completely. For 10K points from the entire dataset, DandF ran for 48 hours, did not complete execution. We then specified a single timeframe with 1000 points and 1 boot-iteration, the procedure ran for 24 hours and did not complete execution! Consequently, we had to abandon these criteria.

We had more success with the tibshirani criteria, at least in terms of getting the procedure to run. For a sample of 50K points from the entire dataset, we get the following results:



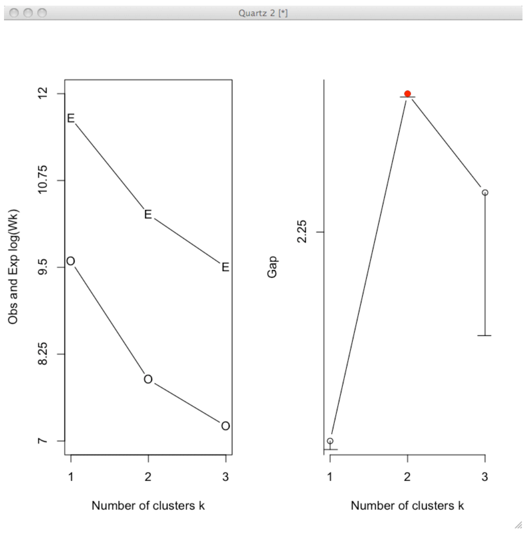


Figure 15: gap results for H2 (left) and H2tri (right).

Figure 15 shows the gap method returning 2 clusters for H2 (left) and H2tri (right). Hence, we are unable to distinguish between the two datasets based on this procedure.

Next, we applied the procedure to the time-series.

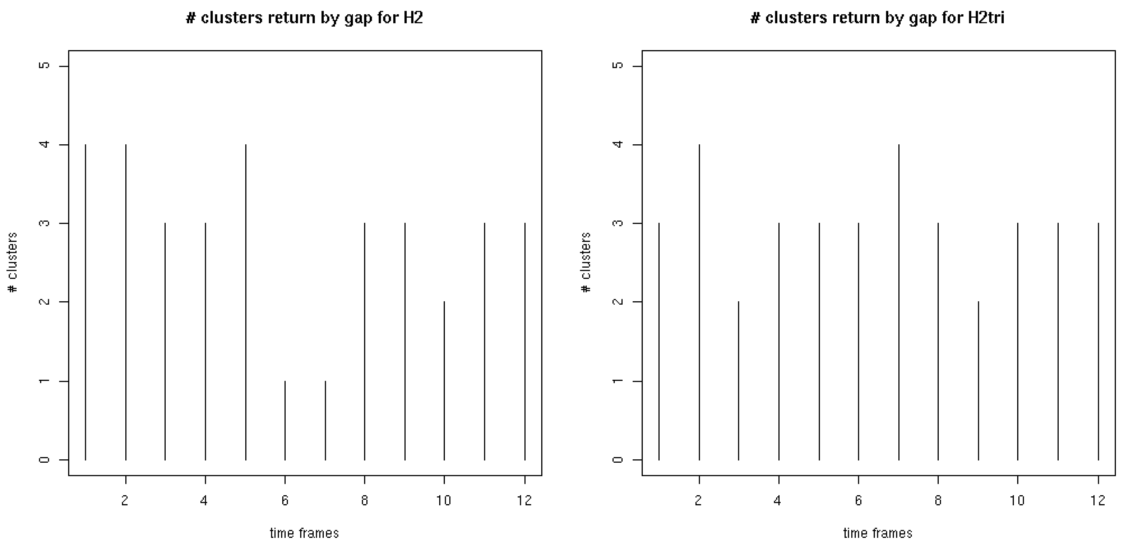


Figure 16: gap results for H2 timeseries (left) and H2tri timeseries (right).

Figure 16 shows that we obtain roughly 3 clusters for both the H2 and H2tri. Running the procedure on each frame was computationally very expensive, we had to increase the window offset to 15K to assure that the procedure completed in a reasonable amount of time (~48 hours).

Model based clustering

We tried to use model based clustering to return the optimal number of clusters. The procedure does hierarchical clustering on Gaussian mixture models and uses the BIC criteria to determine the number of clusters.

We ran for more than 48 hours to process 25K samples from the entire dataset. The procedure warned that the iterations still had not converged. First we tried the procedure with the default (2-8) number of clusters, and we got a warning: “optimal number of clusters occurs at max choice”. We then increased the number of clusters to 20, and still obtain the same warning message: “optimal number of clusters occurs at max choice” “best model: ellipsoidal, unconstrained with 20 components”. We believe that the procedure might have settled into a local optima (the loss function is non-convex), hence we have no reason to trust the clustering results.

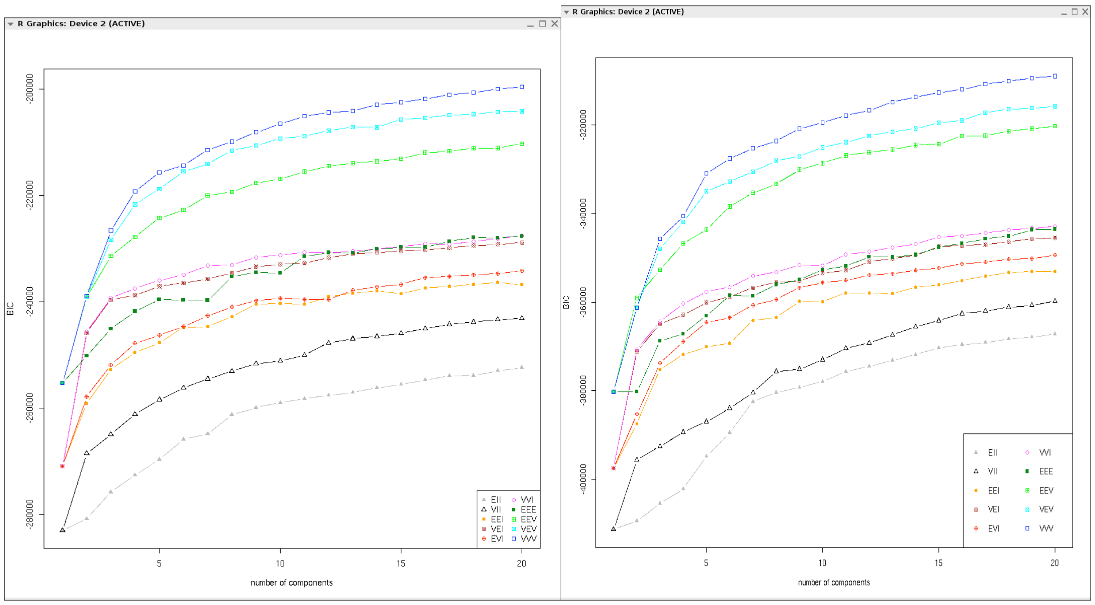


Figure 17: Model based clustering results for H2 (left) and H2tri (right). The BIC plots seem to indicate that the procedure has not converged.

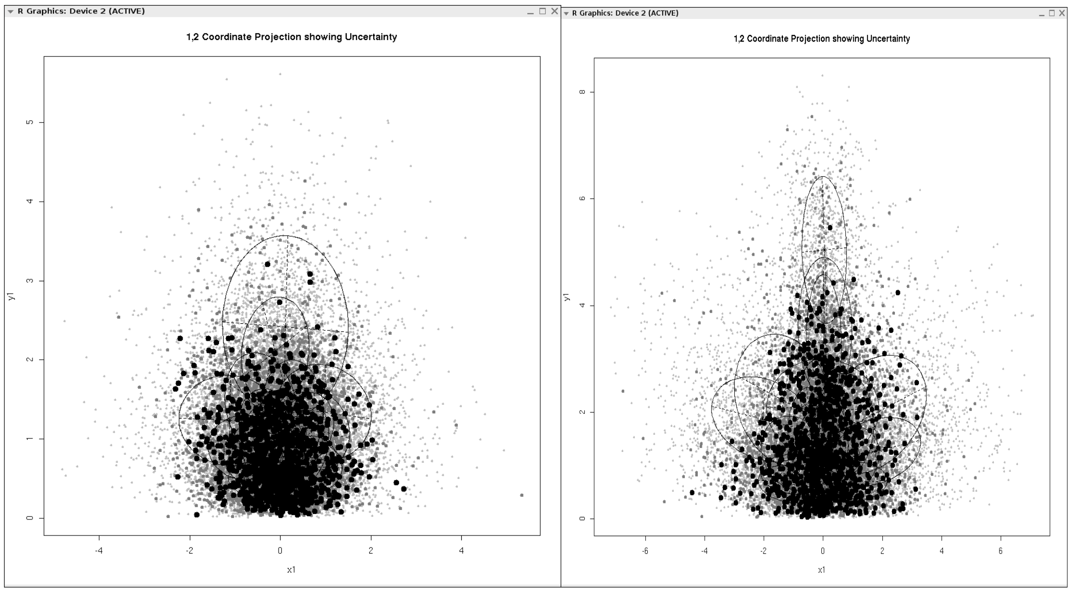


Figure 18: Uncertainty plots of H2(left) and H2tri(right). There are multiple overlapping clusters, which is not a good sign.

Hierarchical Clustering

Finally, we tried to use hierarchical clustering using the R pvclust library. Unfortunately, even after running for 48 hours, we could not obtain results for 50K points. Furthermore, the lack of interactive options to navigate the hierarchical clustering result made it hard for us to interpret the results. Ideally, we would want interactive capabilities to selectively move up and down the hierarchy tree.

Dimensionality Reduction Techniques

We now explore dimensionality reduction techniques (followed by clustering) to see if we can examine alternate patterns in the dataset.

PCA

The canonical technique for dimensionality reduction is Principal component analysis. We use the R princomp() function call. We were careful to make sure that we center the data, before doing PCA. At the same time, we did not scale the values to unit variance, this is important because all dimensions (x1,y1,x2,y2,scalar) are defined in the same space, and larger values of scalar are more important chemically, than smaller values of scalar.

Fortunately, PCA was one of the techniques that was able to process the entire dataset.

PCA Results for H2:

Importance of components:

	Comp.1	Comp.2	Comp.3	Comp.4	Comp.5
Standard deviation	1.309676	1.0077407	0.9921938	0.9772050	0.57428558
Proportion of Variance	0.343052	0.2031093	0.1968907	0.1909869	0.06596112
Cumulative Proportion	0.343052	0.5461613	0.7430520	0.9340389	1.00000000

Loadings:

	Comp.1	Comp.2	Comp.3	Comp.4	Comp.5
x1		0.707	0.707		
y1	0.508			0.707	0.492
x2		-0.707	0.707		
y2	0.508			-0.707	0.492
scalar	0.696				-0.719

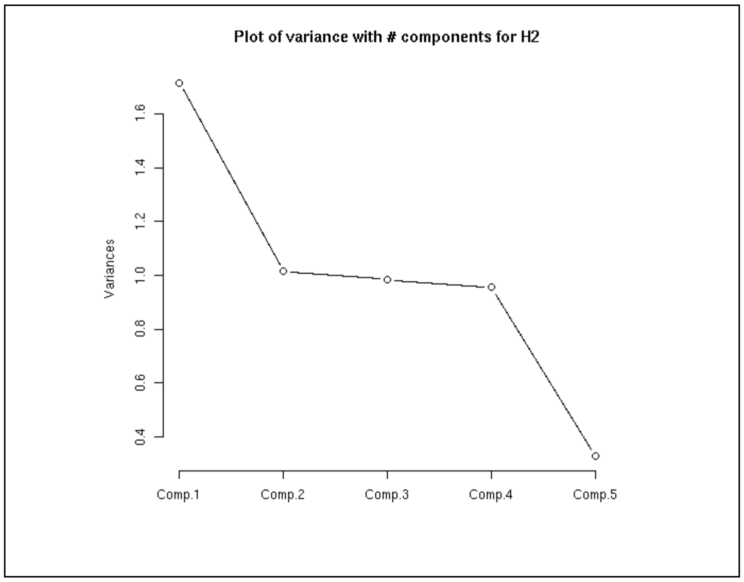


Figure 19: Variance plot for PCA on the H2 dataset

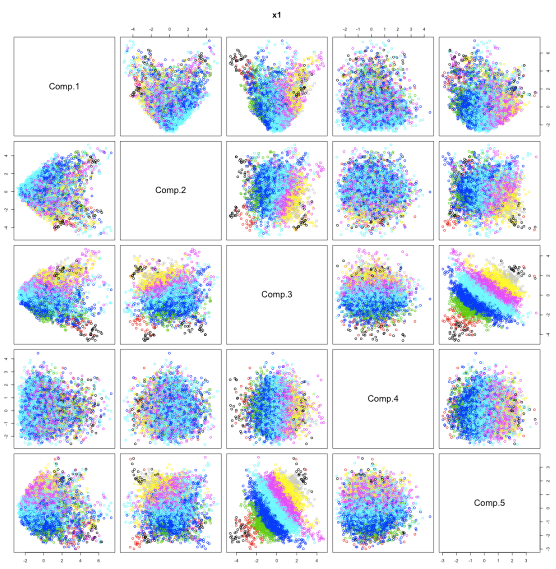
We observe that roughly 4-5 components are sufficient to capture the variance in the dataset. We observe a similar trend further on in the report. This seems to indicate that the dataset does not lie on a significantly lower dimensional subspace.

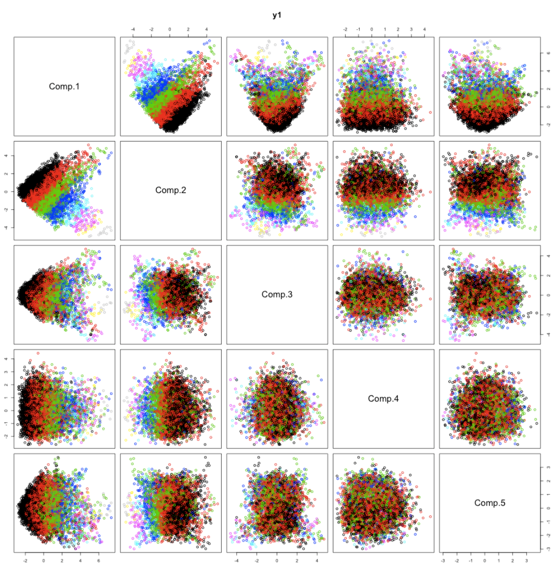
To facilitate the effect of the PCA on the dataset, we will henceforth present a set of 5 plots. These plots (while beautifully rendered) can be a little hard to understand, so we provide an explanation. Generally, we find that using 5 components (either from the PCA or Kernel PCA techniques) is sufficient to explain the dataset. Therefore we create a new dataframe with pairs of "Comp 1", "Comp 2", "Comp 3", "Comp 4", "Comp 5", which correspond to the principal components.

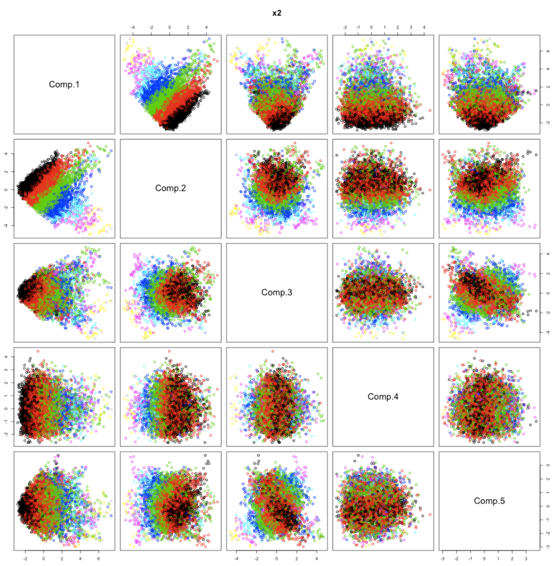
These pair wise plots span the new space. We can now project our entire dataset into this space and we would see black dots indicating the distributions. But we are interesting in observing what effect the new basis has on the original values (x1,y1,x2,y2,scalar). So we choose to *color each set of pair wise plots with the original dimensions x1, y1, x2, y2 and scalar.* This results in 5 plots.

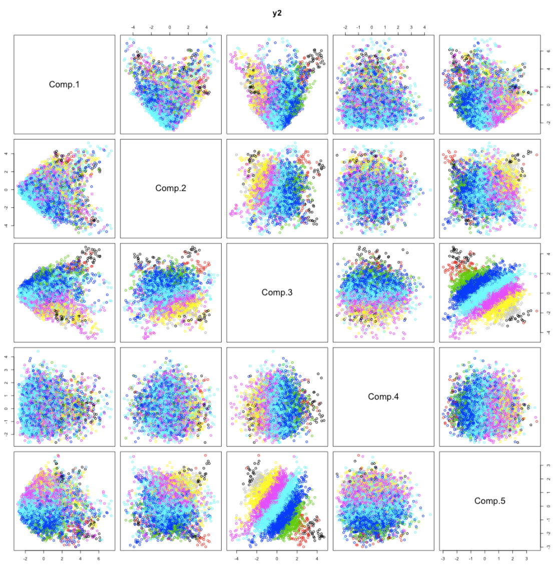
It can be a little hard to interpret a different 5D space (if it wasn't already hard enough to understand the original 5D space) . But we can observe a few patterns. For instance in the following image, Plot #1, showing x1 projected onto the new basis, has a diagonal strip for Comp 3 and Comp 5. This seems to indicate that the new basis has been successful in separating clusters in that dimension. For the same plot, if we see Comp1 and Comp 4, all the clusters are jumbled up, which means that specific basis was not able to separate the clusters.

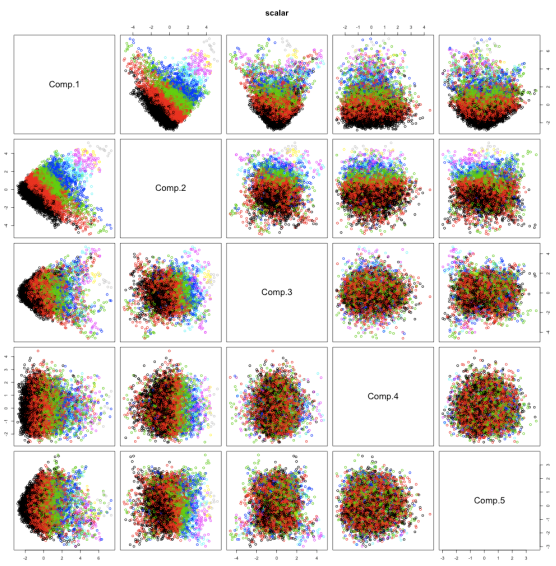
We would like to make a minor note regarding R plotting routines. For some reason, R assumes that if you want to color by some value, that value should be in the range (1 to ∞). However, our values (x1 for instance) are from -3 to +3. Therefore, by default most of our data was plotted in white, which mislead us until we addressed the problem by offsetting the range.











(x1,y1,x2,y2,scalar) projected on the new basis.

PCA Results for H2tri

Importance of components:

	Comp.1	Comp.2	Comp.3	Comp.4	Comp.5
Standard deviation	1.2641043	1.0540908	1.0089498	0.9909643	0.5393682
Proportion of Variance	0.3195935	0.2222226	0.2035970	0.1964030	0.0581839
Cumulative Proportion	0.3195935	0.5418161	0.7454131	0.9418161	1.0000000

Loadings:

	Comp.1	Comp.2	Comp.3	Comp.4	Comp.5
x1			0.707	0.707	
y1	0.478	0.707			-0.521
x2			-0.707	0.707	
y2	0.478	-0.707			-0.521
scalar	0.737				0.676

	Comp.1	Comp.2	Comp.3	Comp.4	Comp.5
SS loadings	1.0	1.0	1.0	1.0	1.0
Proportion Var	0.2	0.2	0.2	0.2	0.2
Cumulative Var	0.2	0.4	0.6	0.8	1.0

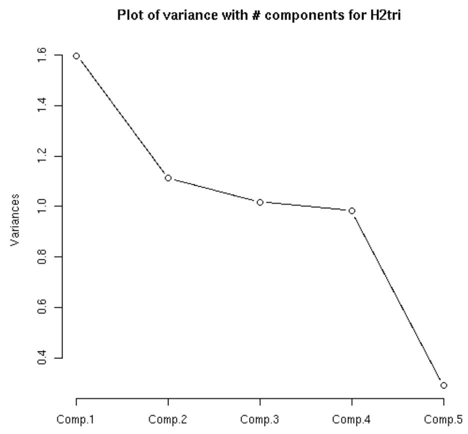
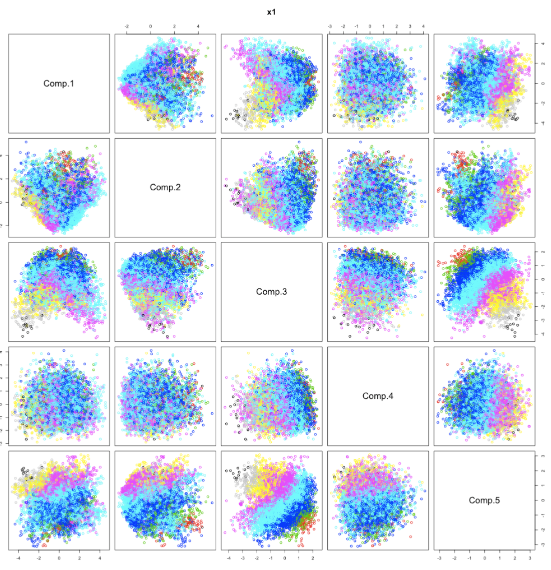
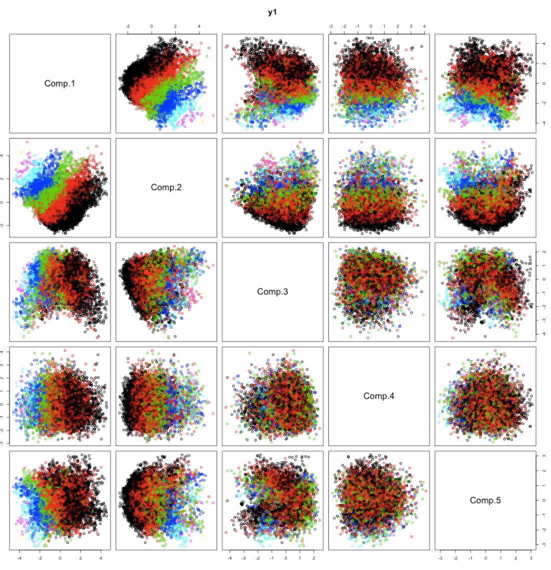
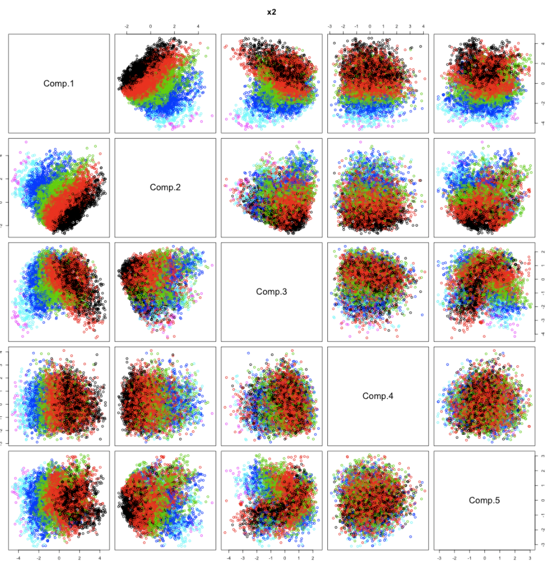
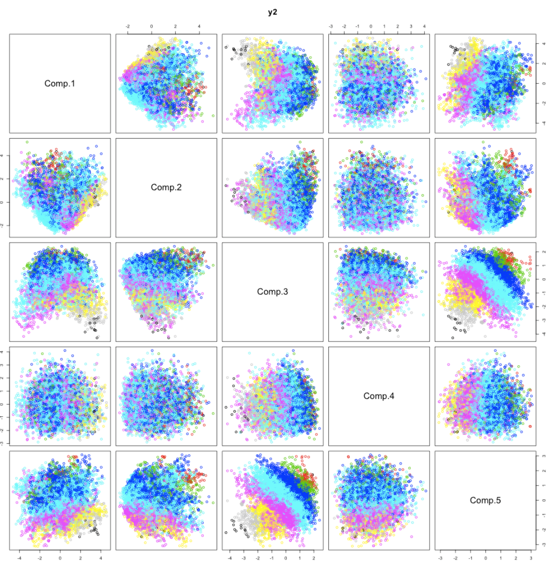


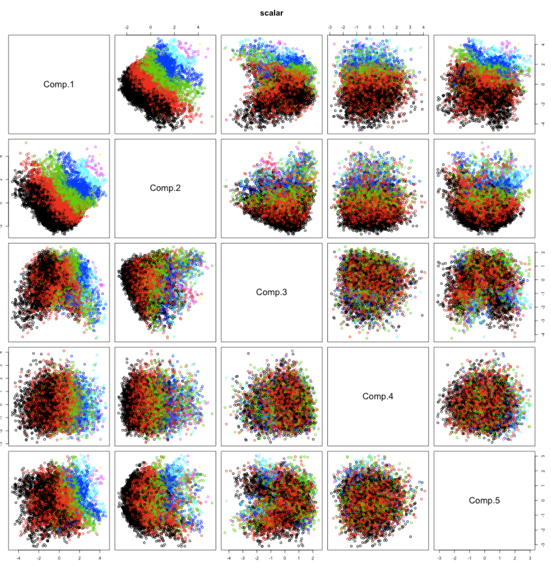
Figure 20: Variance plot for PCA on the H2tri dataset











(x1,x2,y1,y2,scalar) projected onto the new basis.

For both H2 and H2tri, we observe interesting diagonal patterns and some horizontal/vertical patterns in the preceding plots. The goal of the PCA analysis was to let us explore clustering in another basis space, and this is what we do next. We tried clustering using pamk() analysis on this new basis. We applied the analysis on the timeseries and got the following results:

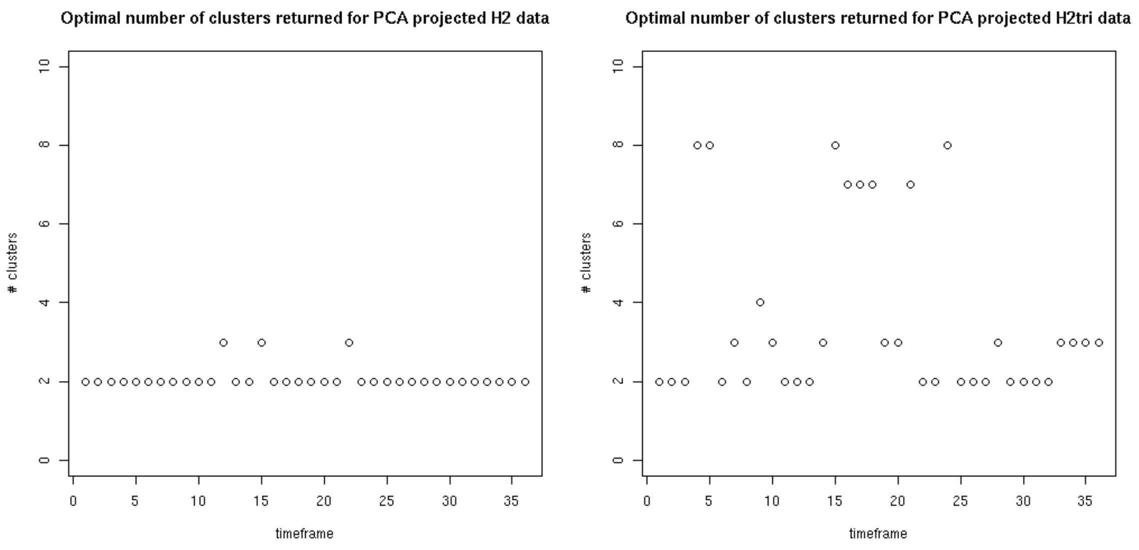


Figure 21: pamk results for H2 timeseries (left) and H2tri timeseries (right) on the new basis.

We again observe that pamk() is able to distinguish between the H2 (left) and H2tri (right) datasets based on the number of clusters over the time-series. The number of clusters for H2 is 2 and higher than 2 (maybe 3 or 7) for H2tri. We choose a representative frame (e.g. #17) for both datasets to further examine the shape of the clusters.

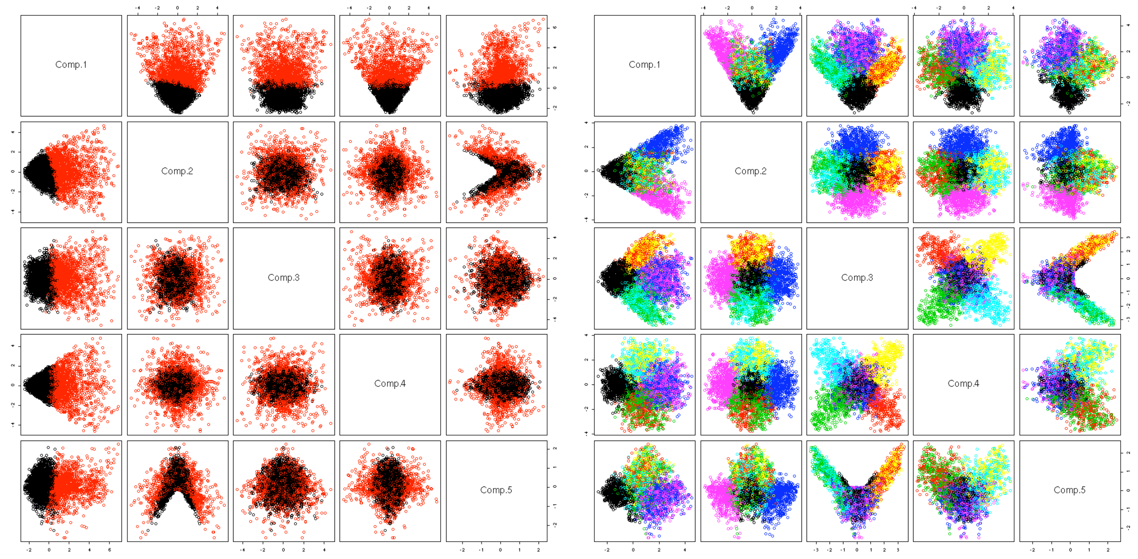


Figure 22: Clusters returned for Frame 17 for H2(left) and H2tri(right) in the new PCA basis.

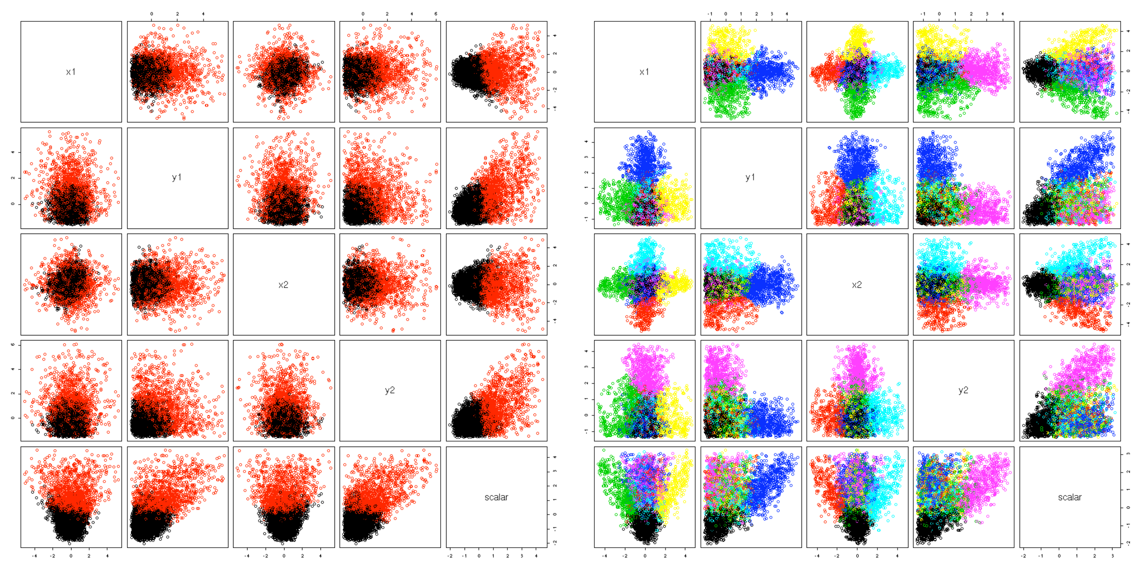


Figure 23: Corresponding clusters in Frame 17 for H2(left) and H2tri (right) in the original space.

We closely examined these clusters, especially in contrast with the vanilla pamk() clusters. It appears that this set of clusters is structurally more interesting, they seem to be located in the right regions. They are able to "diffuse" more with each other, which is possible since this is a noisy simulation. The boundaries of these clusters are also more feasible. From a chemistry point of view, silhouettes can correspond to active chemical surfaces, which is an interesting link. But trying to maximize *all* silhouettes at the same time (which is what the pamk procedure does) is something that we need to examine in more detail.

Note that we did not perform PCA (or other dimensionality reduction analysis) on the timeseries data. We could have easily done so, the major issue was how we would integrate the (possibly) different basis sets across the multiple timeframes. Morevoer, PCA was able to handle the entire dataset at once, which is what was we wanted to do.

MDS

We tried to apply Multi-dimensional scaling using the R cmdscale() function call. This version failed with an out of memory error for 200K points. We reduced the number of points to 50K, but the version still failed. Upon looking at the implementation, we realized that MDS internally creates an O(n^2) matrix, which makes this method intractable for our dataset.

Kernel PCA

Kernel techniques offer a powerful extension of classification/clustering methods by adding non-linear combinations of input dimensions. We wanted to explore Kernel PCA techniques for our dataset, but we faced the following challenges:

- We had to manually explore the parameter space for different kernel parameters. This is in stark contrast to the classification case, where Kernel SVM techniques have some labeled/training data, and implementations are able to automatically determine parameters for different kernels (polynomial degree, rbf sigma, etc) by bootstrapping. We can't do that for our clustering problem.
- Kernel PCA techniques fail outright for 200K points, in fact they barely work for 10K points. We had to explicitly create feature maps (essentially augmenting the non-linear terms to the original dataset) and perform regular PCA.

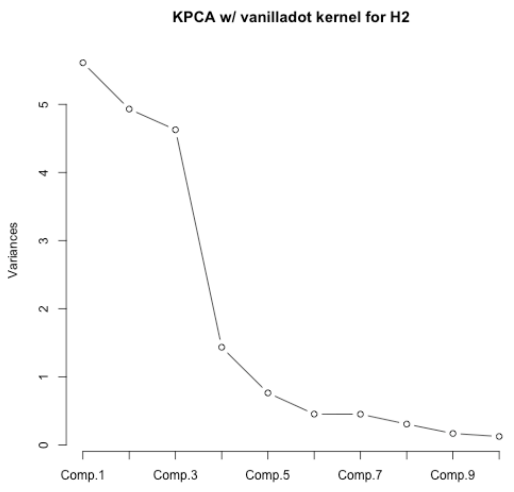
We ended up randomly sampling 5K points from the entire dataset, and manually exploring different kernel PCA techniques (rbfdot, polydot, vanilladot, tanhdot, laplacedot, besseldot, anovadot and splinedot from the kpca() function in the library kernlab). Our criteria for determining a 'good' kernel (and associated parameters), was one that would result in a good separation of the original dimensions.

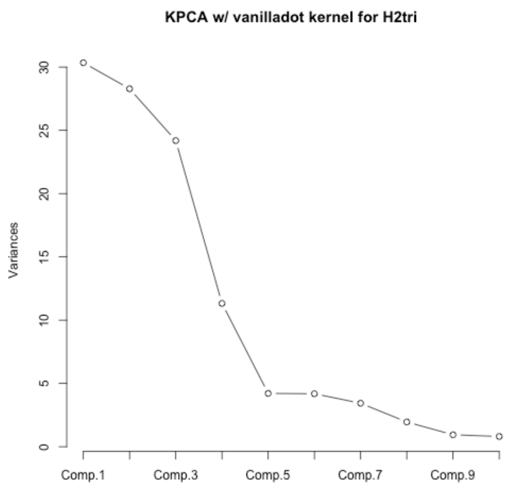
Of course, we stood the risk of not choosing a representative sample of the dataset and determining the non-optimal set of parameters, but we didn't have much choice. One can envision more computationally extensive/elaborate techniques to make the procedure less subjective.

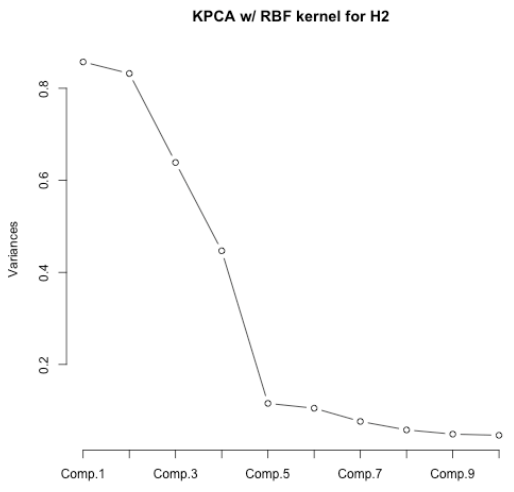
From our explorations, we found that an rbfdot kernel (sigma=0.1), vanilladot and polydot (degree 2 polynomial) result in interesting patterns, and we chose to pursue those more carefully. The splinedot kernel was the most spectacular in terms of its failure, it ran for ~24 hours and produced rather poor results.

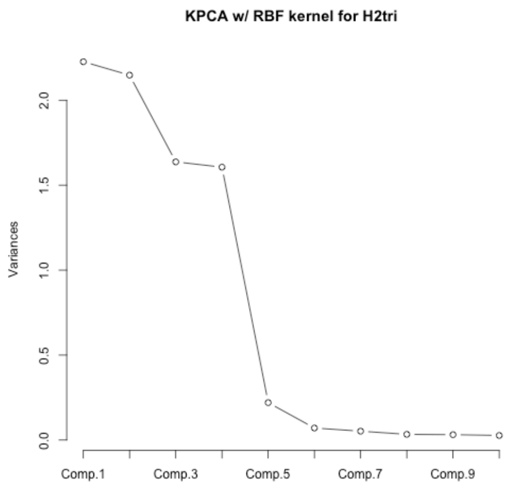
For the three chosen kernels, we explicitly constructed the new feature vectors using some C++ code and augmenting the data files. We then ran conventional PCA analysis. The results are presented in a fashion similar to the last section.

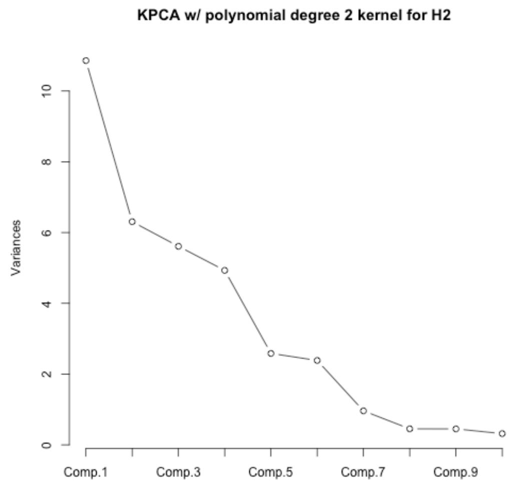
Plots of variance for multiple kernels

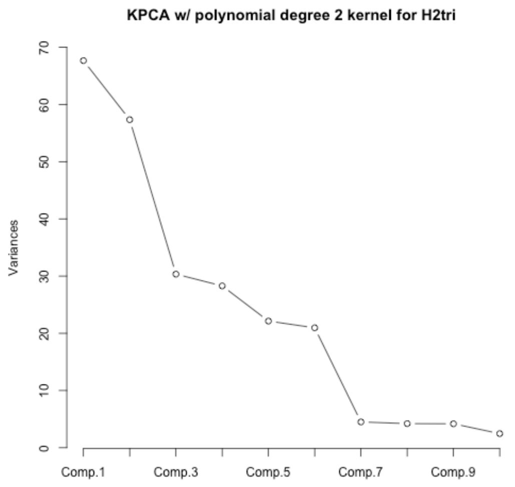




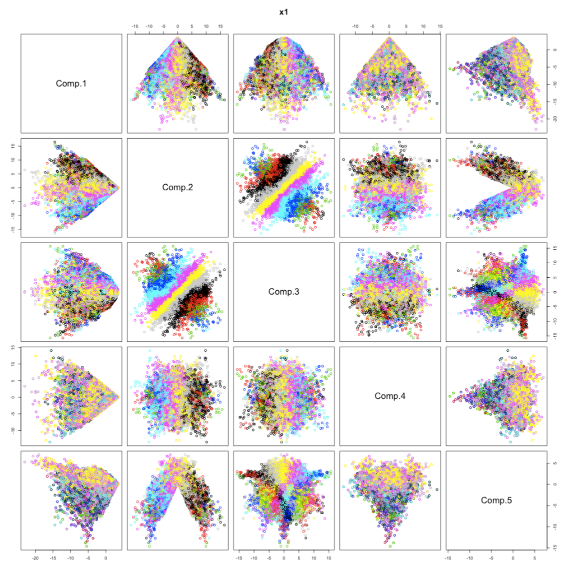


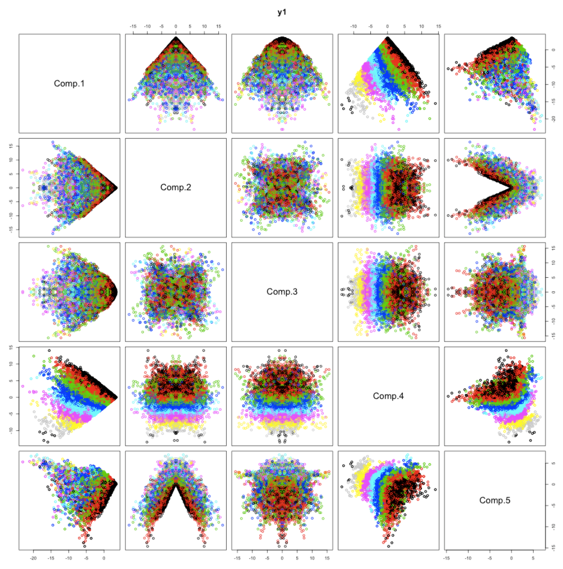


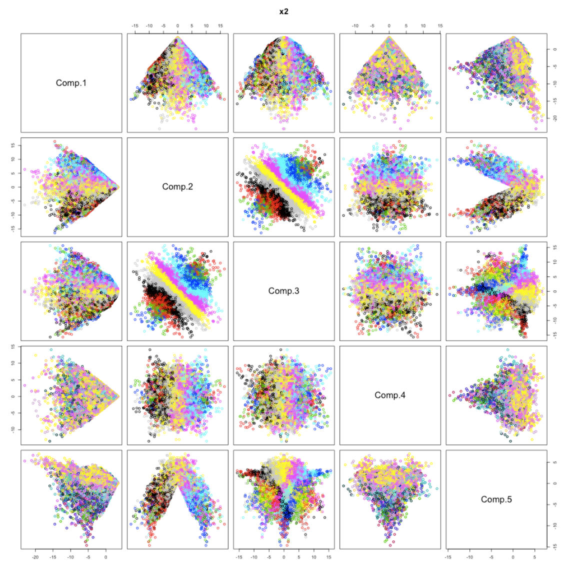


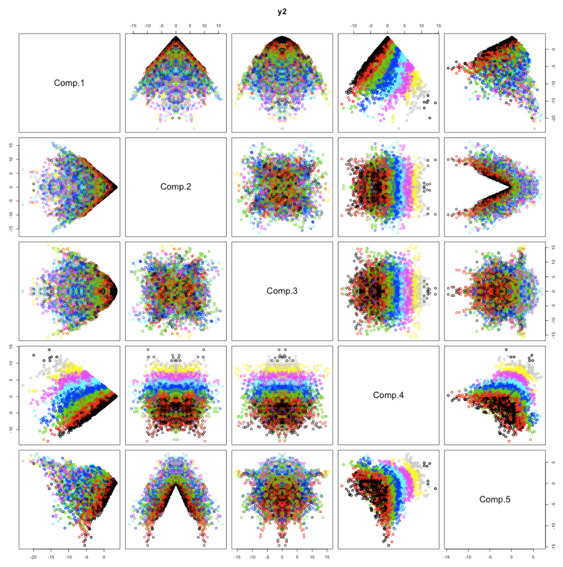


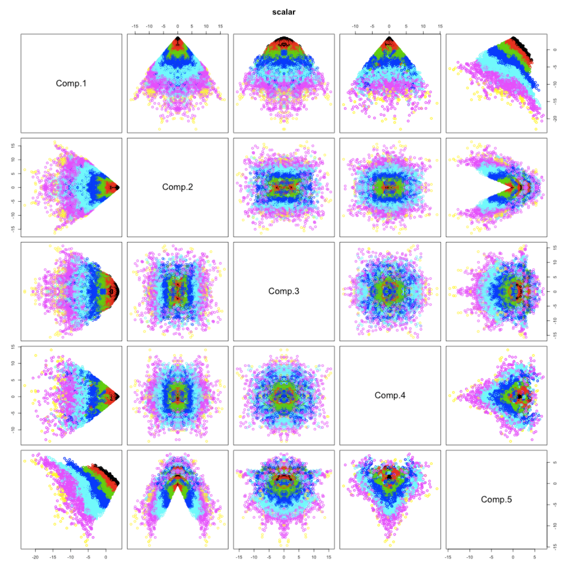
For all of the kernels, we generally find that 5 components are sufficient to explain the variance in the augmented dataset. We therefore use the first 5 components and present the results from projecting the (x1,y1,x2,y2,scalar) values on the new basis.



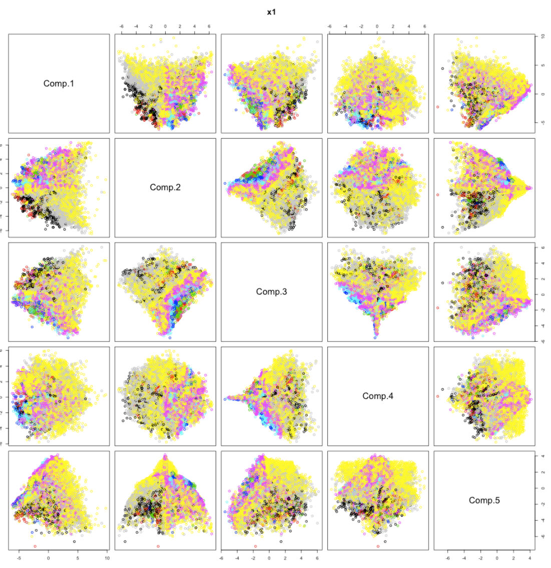


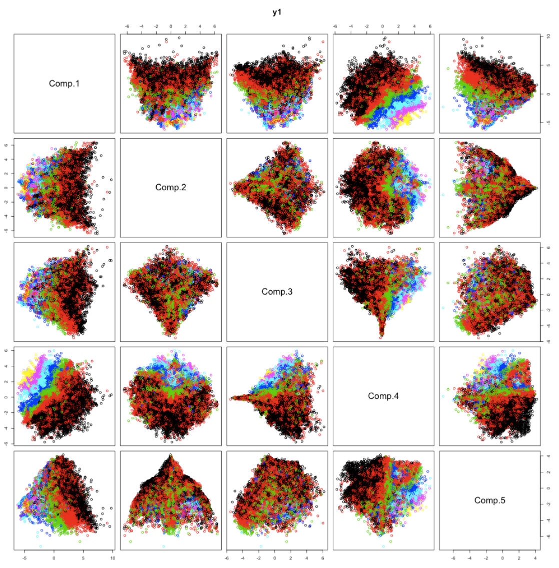


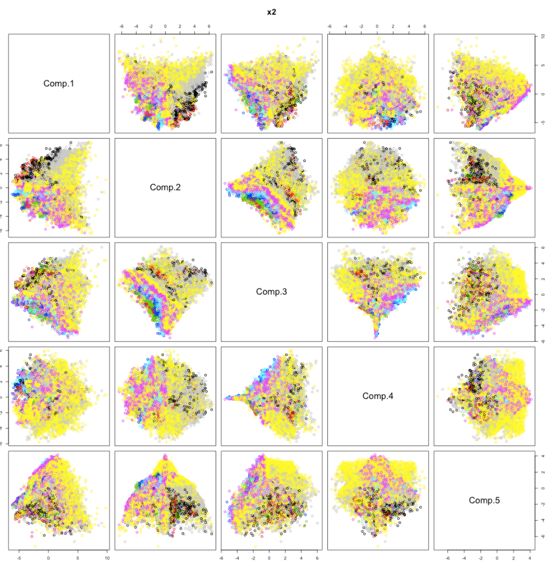


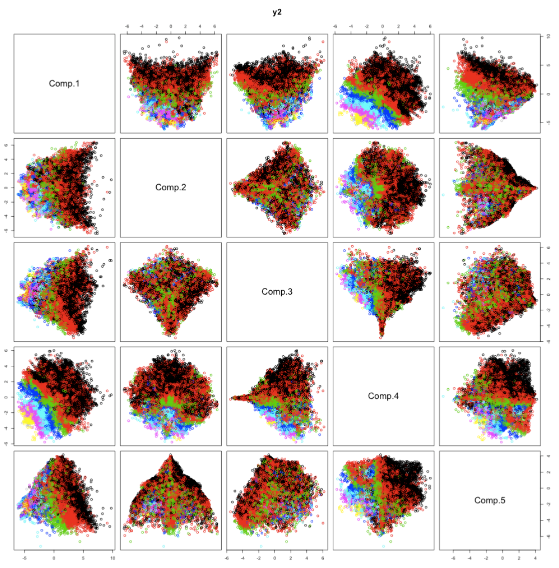


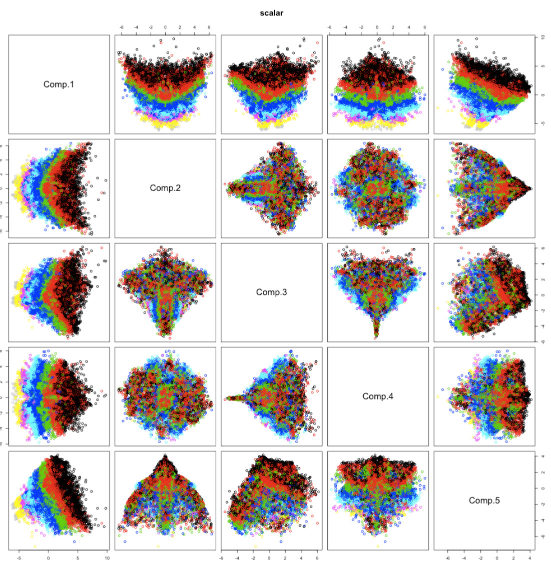
KPCA for H2, RBF, sigma=0.1



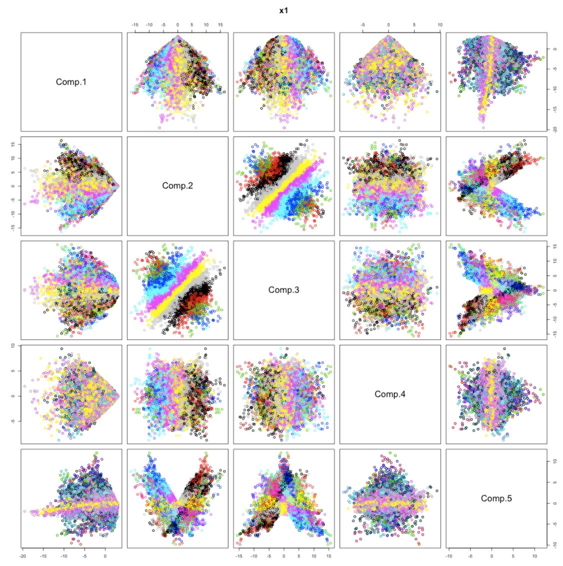


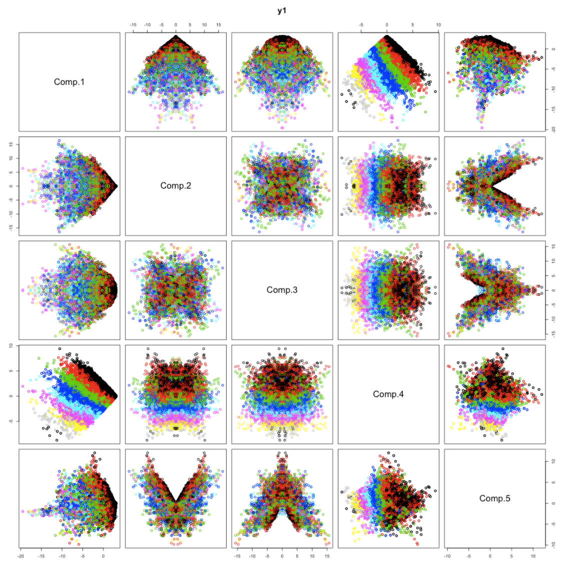


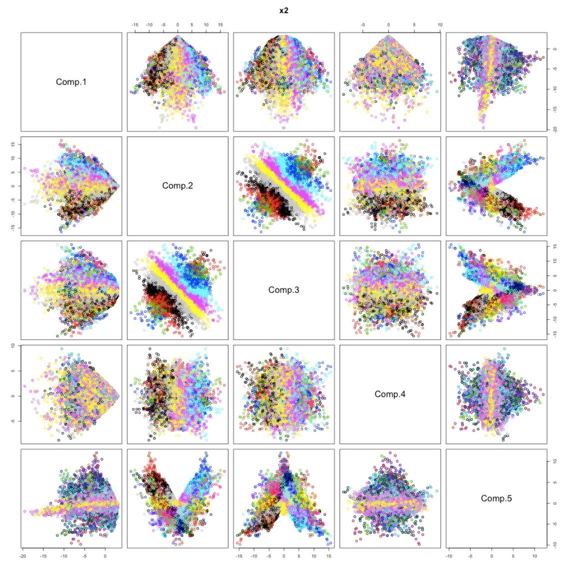


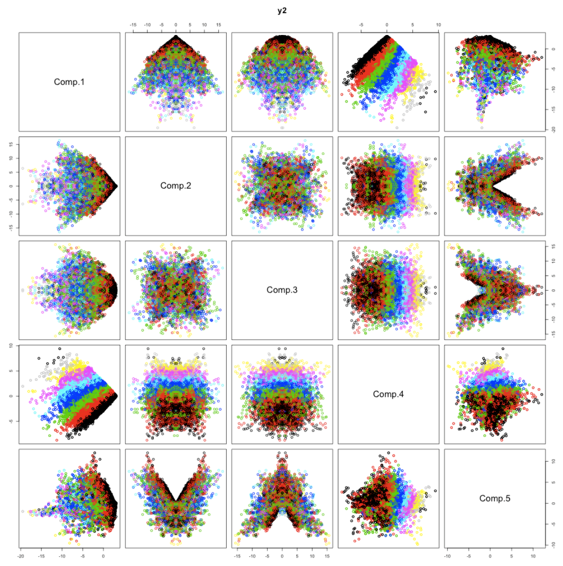


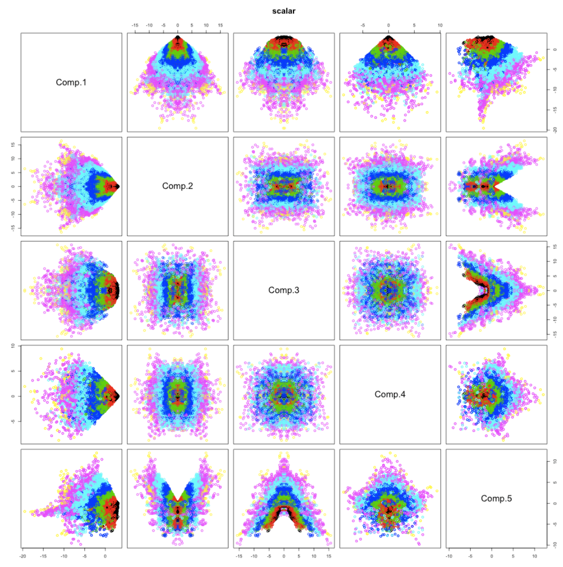
KPCA for H2tri, RBF kernel, sigma=0.1



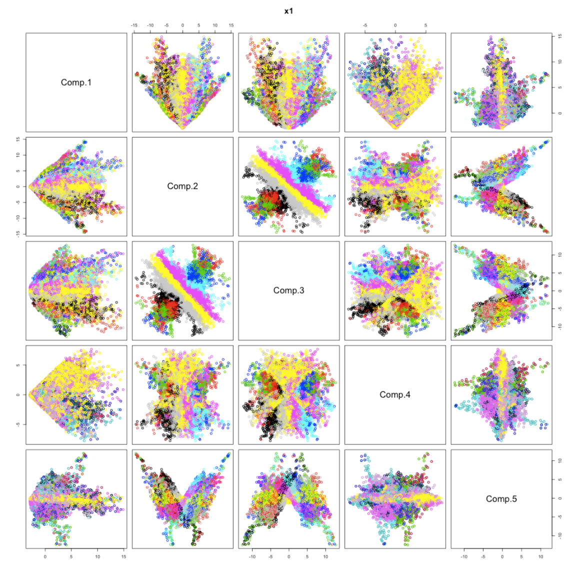


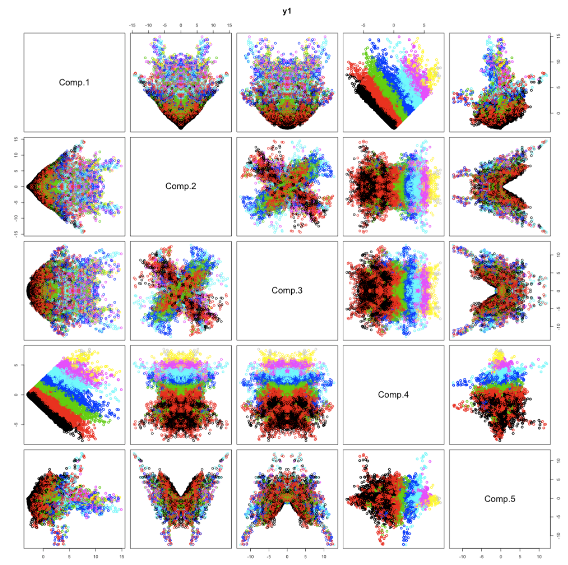


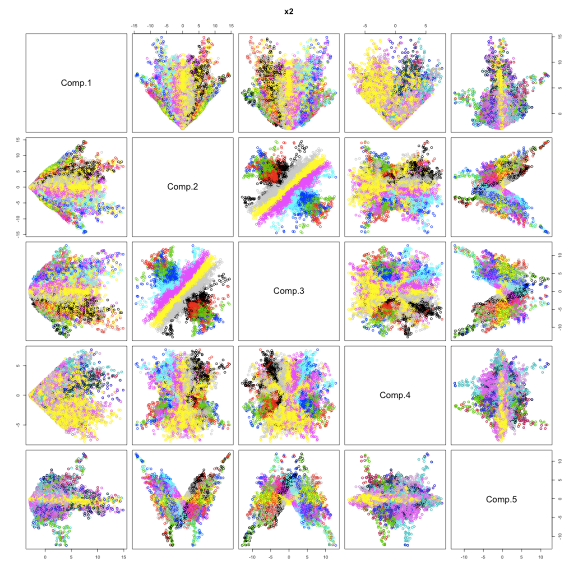


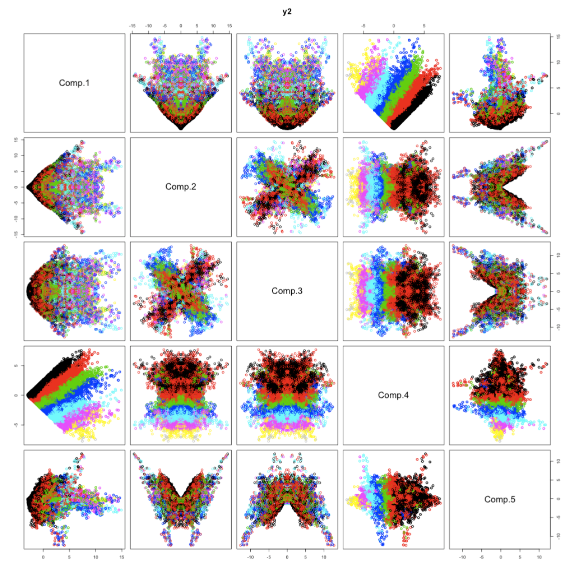


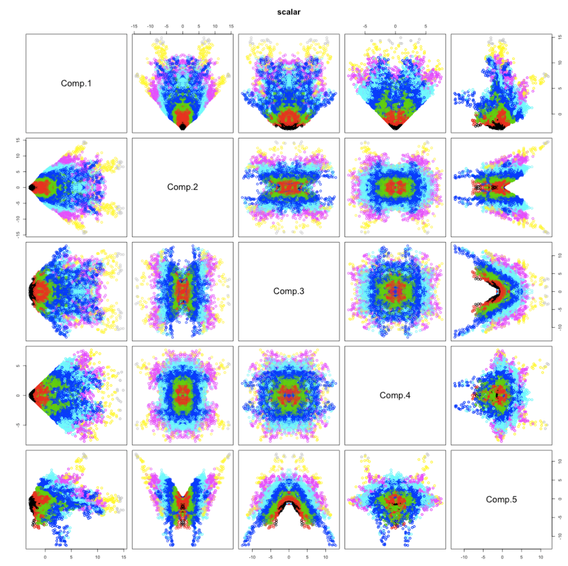
KPCA for H2, vanilladot kernel



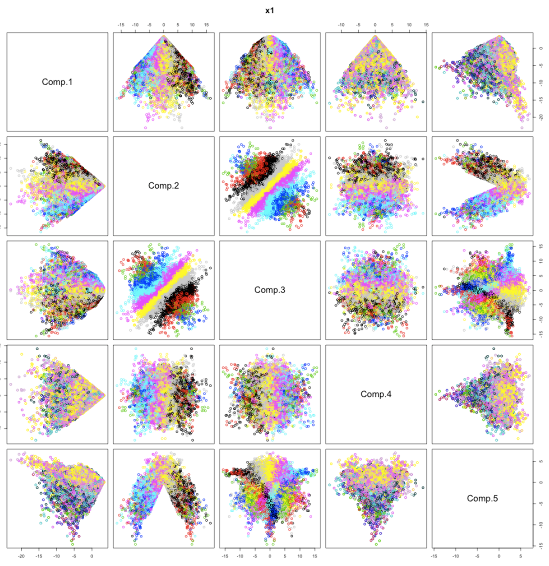


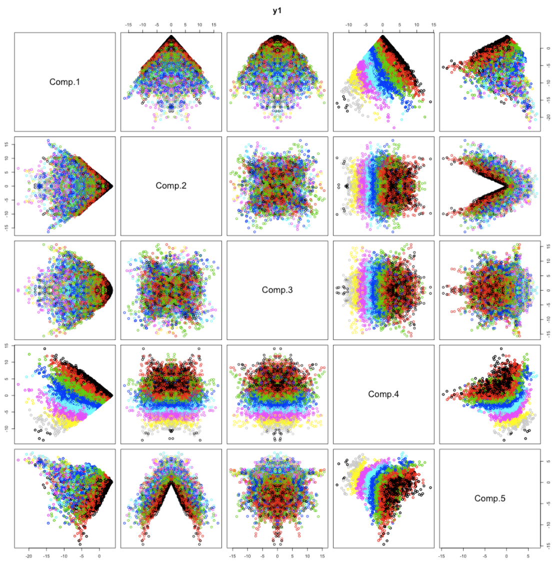


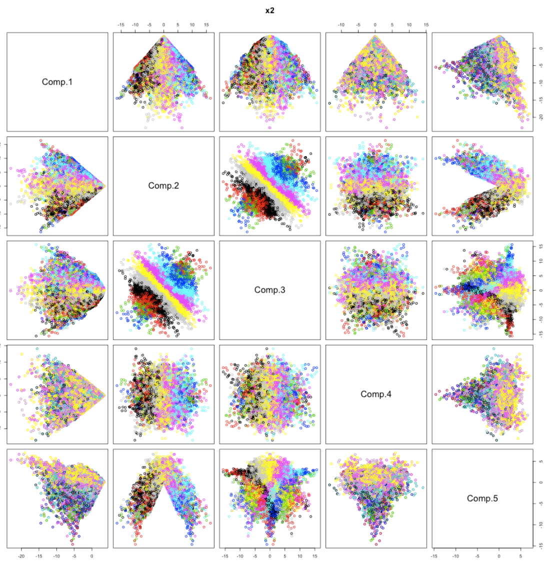


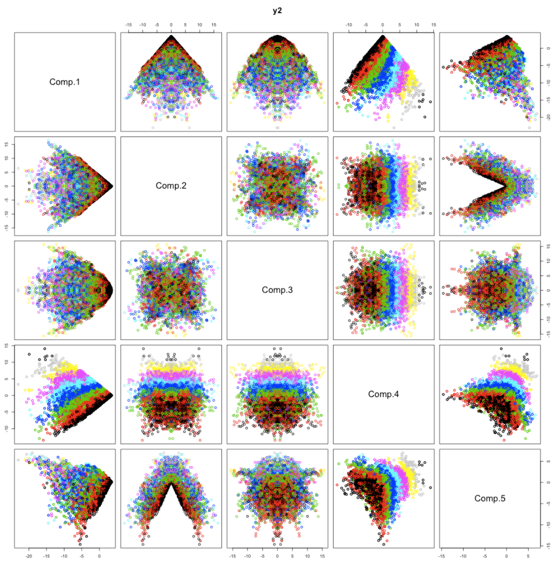


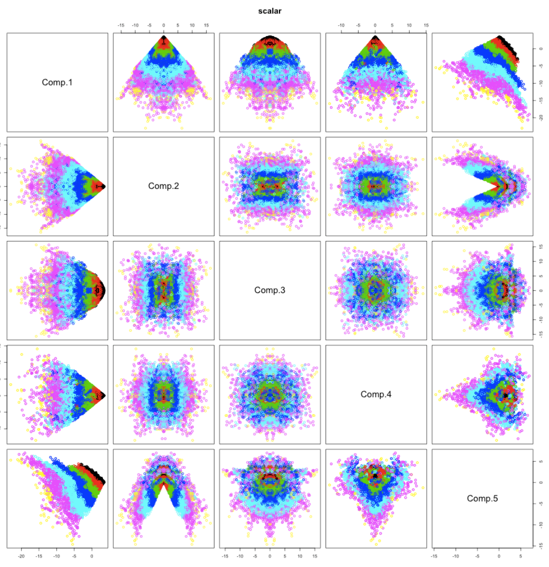
KPCA for H2tri, vanilladot kernel



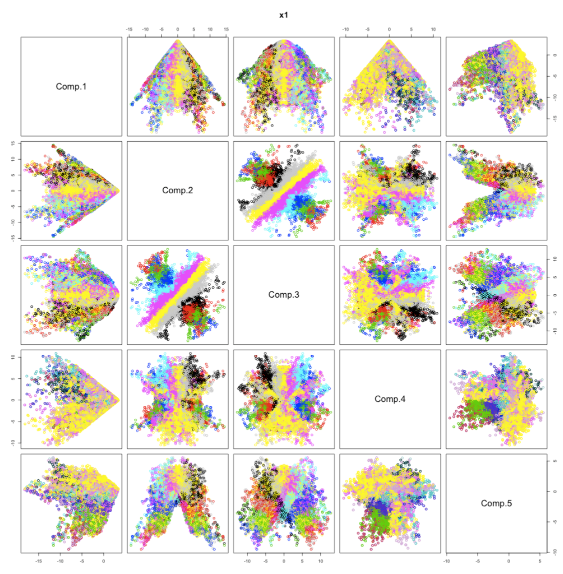


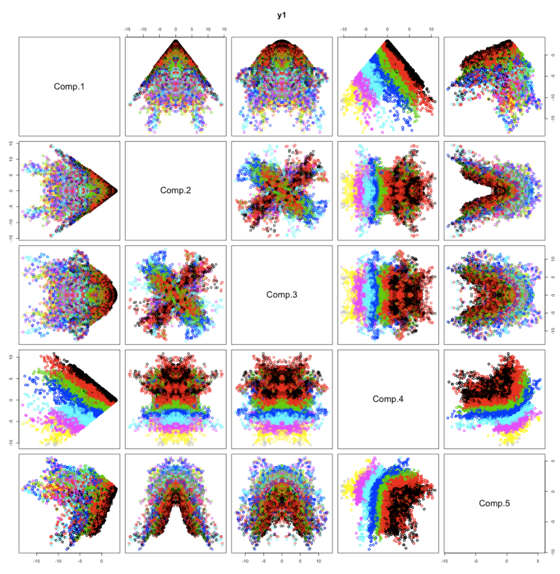


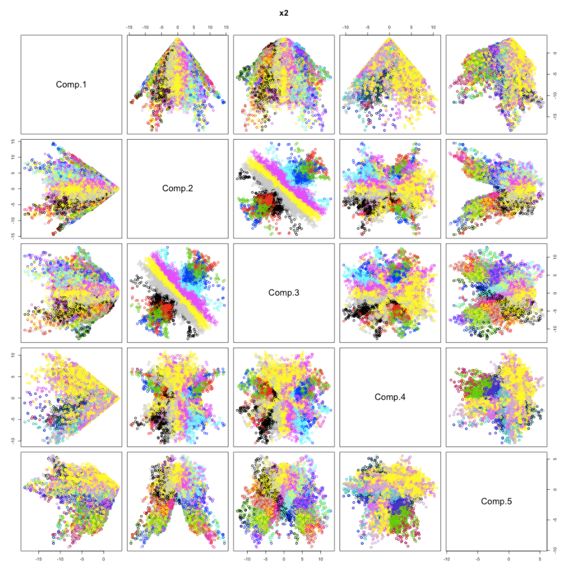


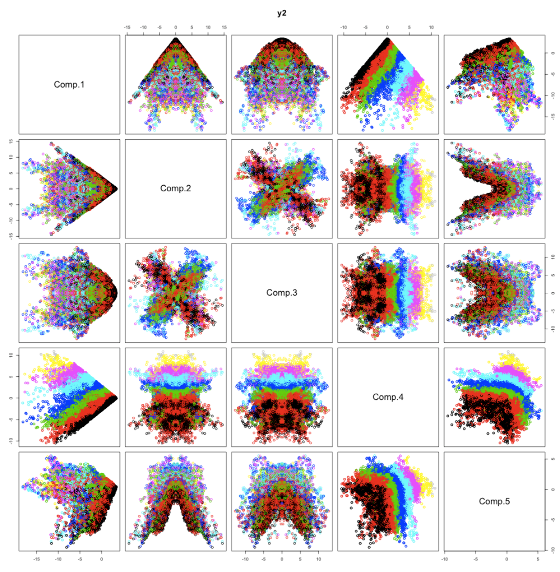


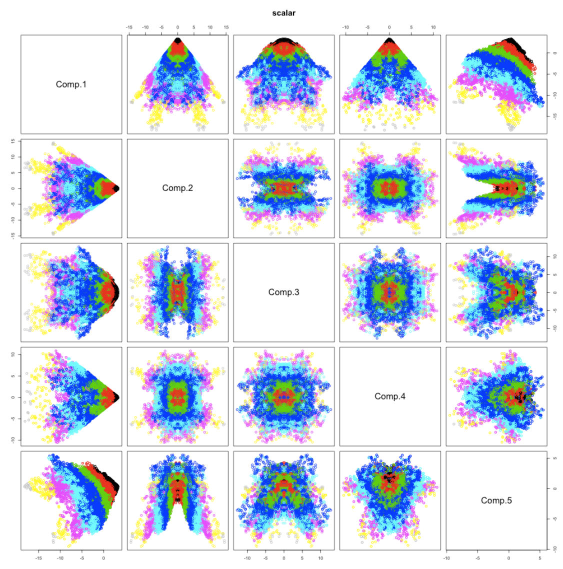
KPCA for H2, Polynomial degree 2 kernel











KPCA H2tri: polynomial degree 2 kernel

In general, we observe that the plots reveal intricate patterns that are more complex than the PCA case. We tend to see somewhat similar patterns across the three kernels, which is probably because they share similar inner products. Apart from diagonal strips, which had emerged in the PCA case, we see concentric patterns and strips that fan out radially (exemplified in the H2tri scalar projection for Comp1 and Comp 2).

Unfortunately, understanding an alternate linear basis is complex enough, understanding a non-linear basis is a non-trivial task that requires mapping the new patterns back to a scientifically relevant phenomena or property. We are currently working on this.

Followed by the KPCA projections, we applied pamk() analysis to determine the optimal number of clusters for the time-series of the 2 systems.

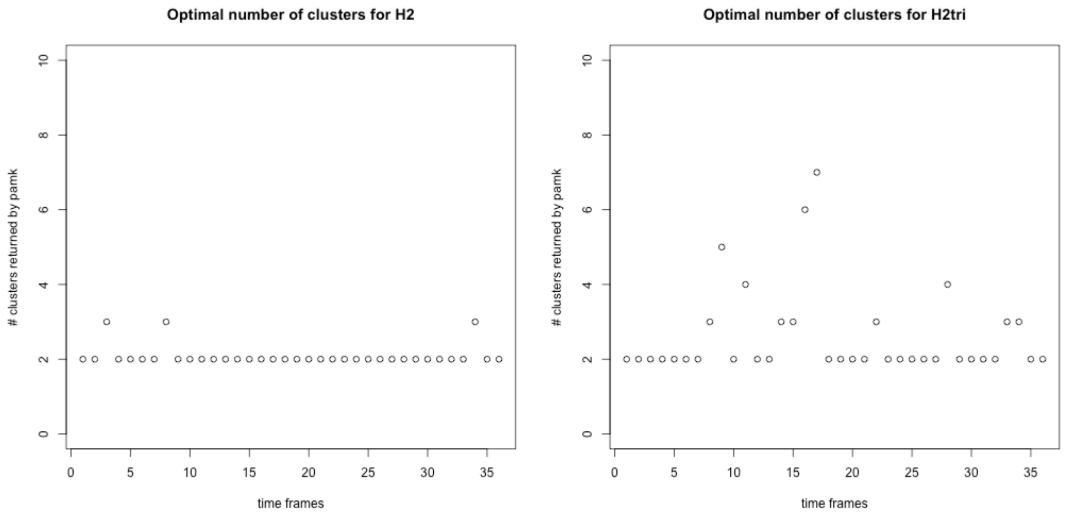


Figure 24: pamk results for H2 timeseries (left) and H2tri timeseries (right) on KPCA basis.

For the 2nd degree polynomial kernel, we get 2 clusters for H2. There appear to be more than 2 clusters for H2tri, and while it's hard to quantify exactly how many more clusters there are (maybe 3 or 4), the fact that there is a difference is interesting.

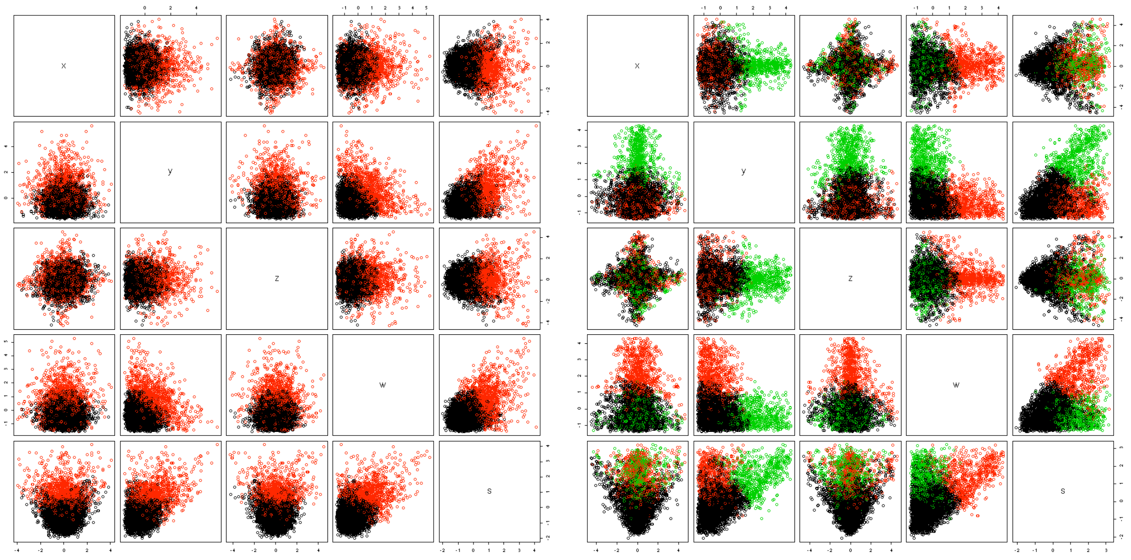


Figure 25: pamk clusters for H2 (left) for timestep=20 and H2tri (right) for timestep=15.

Manifold Learning Techniques

Finally, we explored some manifold learning techniques for the datasets. We were unable to find adequate implementations of the packages in R, and eventually used a Toolbox for Dimensionality Reduction in MATLAB.
http://homepage.tudelft.nl/19j49/Matlab_Toolbox_for_Dimensionality_Reduction.html

First, we tried some techniques to determine the intrinsic dimensionality of the system. Most techniques had computational problems in analyzing the entire dataset; we randomly sampled 50K points from the dataset and running analysis on those points. We obtained the following results:

	H2	H2tri
Eigenvalue	4	4
Maximum Likelihood Estimate	5	5
Correlation Dimension	4	4
Geodesic Minimum Spanning Tree	4	4

These results seem to be consistent with our earlier tests with PCA/KPCA techniques.

Thereafter, we applied some manifold learning techniques (Locally Linear Embedding, Hessian LLE and IsoMAP). Again, we faced computational problems with all methods. None of the methods was able to handle 200K points. We were able to get LLE to run on 2000 points, but it is unclear if one can generalize to the complete dataset even if one is able to learn a manifold from such a sparse sample.

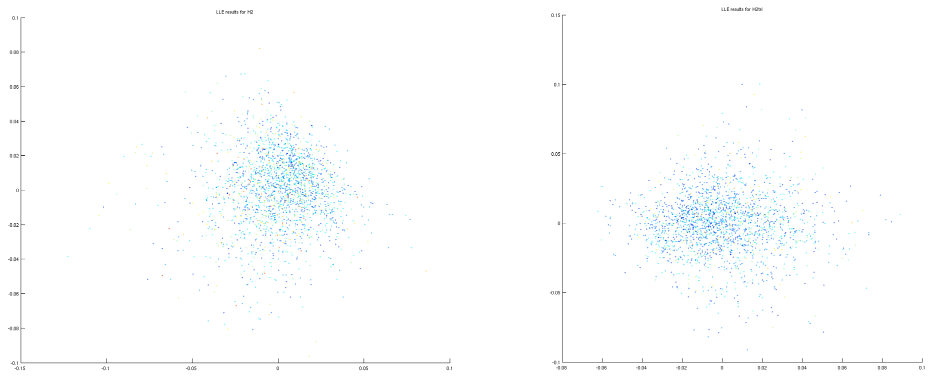


Figure 26: LLE embedding for H2(left) and H2tri(right). Images are colored by the scalar value.

Images are colored by the scalar value of the original points. We do not observe any interesting pattern in the embedding wrt the scalar values. This form of visualization in fact made us realize that such images are useful only if one could attach an "icon" to the original data point. So things like small thumbnails for images, or letters, or text words greatly help in understanding the projection. But if one has raw multi-dimensional data, such embeddings can be very hard (if not impossible) to understand.

The Hessian LLE implementation in the toolbox failed to run on 2000 points with a LU factorization error. It could be we could avoid this failure by adjusting some thresholds or parameters within the Matlab toolbox code, or that there is some other fundamental problem with applying the Hessian LLE to this dataset (which at best has a 4D manifold). Finally, we tried to apply the IsoMAP technique to 2000 points. Following a pattern, which we had seen numerous times by now, the code tried to allocate a O(n^2) matrix and the implementation was unable to handle the large internal data structures.

Scalability issues

The greatest hurdle that we faced in the current project was the lack of scalability of current R libraries for clustering and dimensionality reduction. Apart from the two notable examples of K-means clustering and vanilla PCA, all other techniques simply failed to load the entire dataset (200K points). The modes of failure ranged from procedures complaining of vectors being too long, running for extended periods of time (48-72+ hours) without completing execution and failing to converge. While we explicitly ran on a machine with 32GB of memory [thereby preventing memory allocation issues], most techniques merely used up 2-6GB of memory; which indicates that internal 32-bit addressing was not a constraint. While it would have been nice to have access to a multi-core or parallel version of R (R+MPI), we simply ran out of time to explore that in this project.

We overcame these issues by resorting to sampling a subset of the particles, or splitting the dataset into overlapping frames followed by sampling, both of which are adhoc solutions. We could have taken multiple samples (for analysis) in every timeframe; this would have provided us with some error bars for our various estimates. But even single iterations of analysis on individual samples took exceedingly long times (~30-45 minutes) and it was impractical to do multiple runs. We do note however, that in many areas in science, error bars are almost as important as the observation itself, and we do plan on running more tests in future work.

Summary of Results

This project was intended to address the following scientific questions:

1) Do high-dimensional phase spaces characterizing electronic structure of molecules tend to cluster in any natural way? Do we see a change in clustering patterns as we explore different electronic states of the same molecule?

2) Since it is hard to understand the high-dimensional space of trajectories, can we project these trajectories to a lower dimensional subspace to gain a better understanding of patterns?

3) Do trajectories inherently lie in a lower-dimensional manifold? Can we recover that manifold?

After extensive statistical analysis, we are now in a better position to respond to these questions.

1) We definitely see clustering patterns, and differences between the H2 and H2tri datasets. These are revealed by the pamk method in a fairly reliable manner and can potentially be used to distinguish bonded and non-bonded systems and get insight into the nature of bonding.

2) Projecting to a lower dimensional subspace (~4-5) using PCA or Kernel PCA reveals interesting patterns in the distribution of scalar values, which can be related to the existing descriptors of electronic structure of molecules. Also, these results can be immediately used to develop robust tools for analysis of noisy data obtained during QMC simulations

3) All dimensionality reduction and estimation techniques that we tried seem to indicate that one needs 4 or 5 components to account for most of the variance in the data, hence this 5D dataset does not necessarily lie on a well-defined, low dimensional manifold.

In terms of specific clustering techniques, K-means was generally useful in exploring the dataset. The partition around medoids (pam) technique produced the most definitive results for our data showing distinctive patterns for both a sample of the complete data and time-series. The gap statistic with tibshirani criteria did not provide any distinction across the 2 dataset. The gap statistic w/ DandF criteria, Model based clustering and hierarchical modeling simply failed to run on our datasets.

Thankfully, the vanilla PCA technique was successful in handling our entire dataset. PCA revealed some interesting patterns for the scalar value distribution. Kernel PCA techniques (vanilladot, RBF, Polynomial) and MDS failed to run on the entire dataset, or even a significant fraction of the dataset, and we resorted to creating an explicit feature map followed by conventional PCA. Clustering using K-means and PAM in the new basis set seems to produce promising results. Understanding the new basis set in the scientific context of the problem is challenging, and we are currently working to further examine and interpret the results.

Science Impact

We have outlined a set of statistical techniques suitable for qualitative and quantitative analysis of the results of stochastic simulations of N-particle fermionic systems. Thus far, the existing forms of analysis require assessment of the relevant noisy scalar fields by visual inspection. Our work is an important step towards automation of the respective analytical techniques. Our current results support the initial assumption that bonding phenomena can be related to the rearrangements of high-dimensional phase-spaces of N-electron systems. These relations can be further explored by looking into quantum-chemical nature of the identified parts of the studied phase-spaces. This opens up a new avenue for the generalization of the theory of chemical bonding.

Future Work

We would like to address the following important questions in our future endeavors:

- Can the pam clustering technique resolve bonded vs. non-bonded states for other molecules?
- Why do the gap statistic and model based clustering techniques fail on this dataset? Is there an underlying scientific reason?
- Is there a chemically relevant interpretation for the features computed by various clustering techniques?
- Can we design a clustering criteria based on known quantum chemical concepts and descriptors?

Finally, we were greatly hampered by the lack of scalable R implementations for most of the clustering and dimensionality reduction techniques. We have simulation data with dozens of coupled electrons and 10M+ timesteps. We would like to explore parallel R or other C++/Matlab code and test if our results still hold at scale.

Acknowledgement

We would like to thank Prof. Jon McAuliffe and Sharmodeep Bhattacharyya for their technical assistance on a number of topics related to the project. We would also like to thank Jiayin Zhong for testing R codes on the datasets.

This work was supported by the Director, Office of Advanced Scientific Computing Research, Office of Science, of the U.S. Department of Energy under Contract No. DE-AC02-05CH11231.

Appendix

In this section, we list a few representative codes that we ran for the different methods.

```
H2 = read.table("/d/visguests/prabhat/NERSC/Zubarev/new/H2_ScF_disp.data", header=TRUE)

library("cluster")

fits=list()

for (i in 2:20) {

   fits[[i]]=kmeans(H2, i,iter.max=1000)

}

############################

library("mclust")

num_models=1:20

fit=Mclust(H2,G=num_models)

############################

split = sample(nrow(H2),10000)

H2_set = t(H2[split,])

pvfit = pvclust(H2_set, method.dist="euclidean", nboot=10)

############################

H2 = scale(H2, center=TRUE)

H2fit = princomp(H2)

summary(H2fit) # print variance accounted for

loadings(H2fit) # pc loadings

H2fit$scores # the principal components

############################

library(lga)

split<-sample(nrow(H2),10000)

H2_gap_tibshirani=gap(H2[split,], K=10, B=2, niter=500)

H2_gap_dandf =gap(H2[split,],K=10,B=2,criteria="DandF",niter=500)
```

```
kpc1<-kpca(~.,data=H2_train_sc[,-6],kernel="rbfdot",kpar=list(sigma=0.1),features=3)

##besseldot:

kpc6<-kpca(~.,data=H2_train_sc[,-6],kernel="besseldot",kpar=list(sigma=0.1),features=2)

##ANOVA:

kpc7<-kpca(~.,data=H2_train_sc[,-6],kernel="anovadot",kpar=list(sigma=0.1,degree=10),features=2)

############################



######## load the pre-computed rbf features directly for H2, sigma = 0.1

H2 =read.table("/d/visguests/prabhat/NERSC/Zubarev/new/H2_ScF_rbf.data",header=TRUE)

H2 = scale(H2, center=TRUE)

H2_proj_data = as.matrix(H2) %*% H2_fit$loadings[,1:5]

H2_kfit = kmeans(H2_proj_data, 7, iter.max=5000)

# pamk for the new basis

H2_clusters = list()

H2_nclusters = list()

H2tri_clusters  = list()

H2tri_nclusters = list()

offset_size = 5000

frame_size = 20000

sample_size = 5000

start_frame = 1

end_frame = 36

for (i in start_frame:end_frame) {

        print(i)

        offset = (i-1)*offset_size

        start_range = offset

        end_range = start_range + frame_size

        split = sample(start_range:end_range, sample_size)

        pamk_fit = pamk(H2tri_proj_data[split,], krange=2:10)

        H2tri_clusters[[i]] = pamk_fit
        H2tri_nclusters[[i]] = pamk_fit$nc
        print(pamk_fit$nc)
}

#########################################
# MATLAB codes for intrinsic dimensionality estimation and manifold learning
#########################################
addpath(genpath('/gpfs/home/prabhat/Zubarev/drtoolbox'))

H2 = load('H2_ScF_disp.data');
H2tri = load('H2tri_ScF_disp.data');

num_points = 50000;
shuffle=randperm(200000);
indices=shuffle(1:num_points);

no_dims = round(intrinsic_dim(H2(indices,:), 'EigValue'))
no_dims = round(intrinsic_dim(H2(indices,:), 'MLE'))
no_dims = round(intrinsic_dim(H2(indices,:), 'CorrDim'))
no_dims = round(intrinsic_dim(H2(indices,:), 'GMST'))



num_points = 1000;
shuffle=randperm(200000);
indices=shuffle(1:num_points);

H2_labels = H2(indices,5);
mappedH2 = compute_mapping(H2(indices,:), 'LLE', 5);
mappedH2 = compute_mapping(H2(indices,:), 'HessianLLE', 5);
mappedH2 = compute_mapping(H2(indices,:), 'IsoMAP', 5);
```