Supporting Information

Use of supervised classifiers in step I-II and cross validation.

*k*NN. Cross-validated predictive accuracies were calculated on data by *k*NN using the R library "*knnflex*". Data classification was conducted by applying the *k*NN (*k*=10).

PLS. Projection to Latent Structure (PLS) was applied on processed data for dimension reduction using the classical SIMPLS algorithm (1) as implemented in the R library "*plsgenomics*".

SVM. The Support Vector Machines (SVM) method (2) was used for data classification using the "*libsvm*" module of the R library "*e1071*". Data classification was conducted by applying the SVM with linear kernel on the first 5 components of PLS.

PCA-CA-*k*NN. Cross-validated predictive accuracies were calculated on data by combining established methods. Principal Component Analysis (PCA) was applied on data for dimension reduction using the standard R function "*prcomp*". Canonical Correlation analysis (CA) was conducted using the standard R function "*cancor*". The *k*-Nearest Neighbor (*k*NN) method was used for data classification using the R library "*knnflex*". The data were projected into a PCA subspace explaining 90.0% of the variance. The resulting PCA score matrix was projected into the CA subspace. Data classification was conducted by applying the *k*NN (*k*=10) on the components of the PCA-CA subspace.

Cross-validation is a statistical method of evaluating and comparing learning algorithms by dividing data into two segments: one used to learn or train a model and the other used to validate the model. In typical cross-validation, the training and validation sets must crossover in successive rounds such that each data point has a chance of being validated against. The basic form of cross-validation is *k*-fold cross-validation, and this is the form of cross-validation used in KODAMA.

In *k*-fold cross-validation the data is first partitioned into *k* equally (or nearly equally) sized segments or folds. Each of these *k* subsets serves in turn as a test set. For each of these *k* test sets, a classifier is trained on the remaining *k-1* folds (the training set). The trained classifier is then used to classify the samples in the test set, and the accuracy is calculated. The combined value of the accuracy over the *k* test sets, which is based on the prediction of all samples one time each, is the cross-validated estimate of that error. Leave-One-Out Cross-Validation (LOOCV) is a special case of *k*-fold cross-validation where *k* equals the number of samples in the data. In other words in each iteration, nearly all the data except for a single sample are used for training and the model is tested on that single sample. An accuracy estimate obtained using LOOCV is known to be almost unbiased but it has high variance, leading to unreliable estimates (3).

A large *k* is seemingly desiderable, since with a larger *k* there are more performance estimates, and the training set size is closer to the full data size. As *k* increases, however, the overlap between training sets also increases, leading to less precise, less fine-grained measurements of the performance metric. In the other hand small *k* speeds up the computation velocity of the cross-validation but the training sets are far to represent the full data.

In KODAMA, we implemented the 10-fold cross-validation because it makes classifications using 90% of the data, making it more likely to be generalizable to the full data; it has been shown to be sufficient to achieve stable values of the accuracy (4). These competing factors have all been considered and k=10 seems to be a good compromise.

Optimization of the φ, T, M, and ε parameters.

Except for the choice of the classifier to use, KODAMA has three different parameters that can slightly affect the analysis: the number of samples φ to select in the maximization of the cross-validated accuracy part, the number of Monte Carlo (MC) steps, T, and the number of iteration of steps I-III, M.

We optimized these three parameters on the basis of the results achieved on three different datasets using KODAMA with *k*NN as classifier. The first dataset was generated with 3 clusters and 5 dimensions. The number of data points for each cluster is 50. Each cluster is created from a different multivariate normal distribution with a different covariance matrix of the features (5). Each covariance matrix was randomly generated with values that range between 0 and 1. The second dataset is the Swiss-roll with 1000 data points described by the following parametric equations: $x=u\times cos(u)$, $z=u\times sin(u)$; where u varies between 1.5π and 4.5π, and y varies between 0 and 21. The third dataset is a spiral with 200 data points defined by the following parametric equation: $x=cos(u)\times(u+a)$, $y=sin(u)\times(u+a)$; where u varies between π and 4π, and a is a value from a Gaussian distribution with mean=0 and standard deviation =0.7.

The MC procedure optimizes the vector W by maximizing A_W through a defined number T of iterations. At each step A_W can increase or at least remain equal. The parameter T defines the number of loops that MC procedures does to optimize the cross-validated accuracy. In Fig. S4, we show how the accuracy A_W evolves during the MC procedure in the three different datasets. After 10 loops, the accuracy A_W achieves the maximum value and the characteristic of the dataset seems to not interfere with the maximization procedure. A good compromise between computational time and quality of maximization is T=20. In several cases the

MC procedure reaches 100% accuracy before completing *T* iterations.

The parameter *M* is the number of times that maximization of the cross-validated accuracy part is repeated. This part is repeated *M* times to average effects due to the randomness of the iterative procedure and sample selection. Larger *M* value provides a better description of the distribution of the data. We calculated the residual variance defined as $r^2(A_{200},A_X)$, where A_{200} is the KODAMA dissimilarity matrix obtained with M=200, and A_X is the KODAMA dissimilarity matrix obtained with the different *M* value tested. The differences between the results with M=100 and M=200 are low. We conclude that M=100 can be a good approximation a larger *M* value (Fig. S4).

A subset $N'=\varphi N$ of samples is selected at the beginning of the maximization of the cross-validated accuracy part. Each time that this part of the KODAMA is repeated a different subset of samples is used. The choice of the number of samples to be used can affect the analysis. The residual variance between the Euclidean distance of data points in the manifold and the KODAMA dissimilarity matrix is used to analyze the data in the Swiss-roll and the spiral datasets. Davies-Bouldin Index (DBI) was used to analyze the results in the 3-clusters dataset. With low φ values KODAMA cannot achieve a good representation of the manifold embedded, whilst KODAMA suffers from problems relative to the "short-circuits" in the neighborhood graph for φ approaching 1. These competing factors have all been considered and φ=0.75 seems to be a good compromise (Fig. S4).

The problem of "short-circuits" emerges also if occasional proximities are taken as meaningful while calculating the final KODAMA dissimilarity matrix using the Floyd's algorithm. We have found that short circuits can be removed by setting equal to zero all P_M values below a certain threshold ε. Empirically, a value of ε of 0.05 eliminates "short-circuits" without introducing fragmentation of the manifold.

Significance of the KODAMA result

The Shannon Entropy (*H*) (6) can be used to assess the significance of the KODAMA result on a high-dimensional dataset. *H* is given by:

$$H = -\sum_{i}^{N}\sum_{j}^{N} v(i,j) \times \log v(i,j)$$

where $v(i,j)$ is $p_M(i,j)$ divided by the sum of all the values in the matrix $P_M=\{p_M(i,j)\}$. *H* values were calculated using the function "*entropy*" in the R library "*entropy*".

In terms of hypothesis testing, we proposed testing the null hypothesis that the available result can be modeled as coming from a single multivariate Gaussian distribution. Our test statistic is the H of the P_M proximity matrix. Simulation of the test statistic is used to produce percentile-based p values. This test assumes that there is no difference in H value between two groups of P_M proximity matrices obtained from two randomly formed datasets. In this test, KODAMA is performed on data from a single multivariate Gaussian distribution generated with the same number of samples and variables, and the same covariance matrix of the original data. From each proximity matrix, a H value is obtained. By repeating the procedure G times, a null distribution of H values is obtained. H_0 is then defined as a distribution of H values that are expected to be insignificant. Statistical significance of KODAMA is then assessed by relating the H value of the KODAMA performed on the original data to the distribution of the H values obtained from KODAMA performed on the multivariate Gaussian distributions. The p value is calculated as the number of H values from the distribution of random data that are smaller or identical to the H value from the original data divided by G. The lower limit of the number G is dictated by the required statistical significance: for instance, to attain a p <0.01 at least G=100 is necessary but may not be sufficient for a proper sampling of the distributions tails.

We compared the H values obtained on 6 different datasets: Swiss-roll, Helicoid, Dini's surface, 3-cluster, and 2 different continuous distribution datasets (Test-1 and Test-2). The last two datasets are generated from a single multivariate Gaussian distribution and they differ for the degree of correlation between variables. In the first one, the variables are not correlated to each other, whilst the second one is generated using a covariance matrix of the variables. The covariance matrix was randomly generated with values that range between 0 and 1. Three hundreds samples were simulated with 10 variables for both datasets. In Fig. S5, we show the KODAMA proximity matrix and the visualization with Multi-Dimensional Scaling of the KODAMA dissimilarity matrix obtained for four of these datasets.

In Table S2, for each dataset we report the H values obtained form KODAMA proximity matrices and H values obtained from the simulation on 100 datasets generated from a Gaussian distribution with the same covariance matrix. Correctly KODAMA identify as not significant the results for Test-1 and Test-2 datasets, and significant the results for the other datasets.

Feature extraction methods

For each feature extraction methods, the number of dimensions in the output space was chosen *a priori* equal to number of classes minus one if the number of classes is major than two, two otherwise.

Multi-Dimensional Scaling (MDS) was performed using the function "*cmdscale*" in the R library "*stats*". Diffusion Maps (DM) (7) was performed using the function "*diffuse*" in the R library "*diffusionMap*".

The parameter "*eps.val*" controls the degree of localness in the diffusion weight matrix. We used the default function to optimize "*eps.val*". Isometric Feature Mapping Ordination (ISOMAP) (8) was performed using the function "*ISOMAP*" in the R library "*vegan*". We performed ISOMAP optimizing the suitable neighborhood size through estimating the "quality" of the corresponding mapping, *i.e.* how well the high-dimensional structure is represented in the embedded space, measured by the residual variance. Principal Component Analysis (PCA) (9) was performed using the function "*prcomp*" in the R library "*stats*". Locally Linear Embedding (LLE) (10) was performed using the function "*lle*" in the R library "*lle*". The optimal number of neighbours was calculates by using the algorithm proposed by Kayo (11) described in the function "*calc_k*" in the R library "*lle*". Random Forest (RF) (12) was performed using the function "*randomForest*" in the R library "*randomForest*". The number of trees was 2000. A higher number of trees should increase the performance of RF. We optimize this parameter *a posteriori* on the basis of the results obtained. MDS was applied on one minus the proximity matrix achieved by RF (13). Sammon's Non-Linear Mapping (Sammon) (14) was performed using the function "*sammon*" in the R library "*MASS*". We leaved unchanged the default parameters. Stochastic Proximity Embedding (SPE) (15) was performed using the function "*spe*" in the R library "*spe*". We leaved unchanged the default parameters. *t*-Distributed Stochastic Neighbor Embedding (*t*-SNE) (16) was performed using the function "*tsne*" in the R library "*tsne*". We defined the perplexity parameter on the basis of the number of data points. Tree preserving embedding (TPE) (17) was performed using the function "*tpe*" in the R library "*tpe*".

The performance of each feature extraction method was analyzed by estimating the relative class overlap using the Davies-Bouldin Index (DBI) (18), a function of the ratio of the sum of within-cluster scatter to between-cluster separation, as implemented in the function "*DBIndex*" in the R library "*RDRToolbox*". DBI is defined as

$$DBI = \frac{1}{nc}\sum_{i=1, i\neq j}^{n} \max\left[\frac{\sigma_i + \sigma_j}{d(c_i, c_j)}\right]$$

where *nc* is the number of clusters, σ_i is the average distance of all samples in cluster *i* to their cluster center c_i, σ_j is the average distance of all samples in cluster *j* to their cluster center c_j, and $d(c_i,c_j)$ is the distance of cluster centers c_i and c_j. Small values of DBI correspond to clusters that are compact, and whose centers are far away from each other.

Clustering methods

High-Dimensional Data Clustering (HDDC) (19) was performed using the function "*hddc*" in the R library "*HDclassif*". Models and parameters were chosen on the base of the maximization of the BIC criterion.

Spectral Clustering (SC) (20) was performed using the function "*specc*" in the R library "*kernlab*". Radial Basis kernel function was used. Spectral clustering based on k-nearest neighbor graph (SCKNN) (21). Clustering was performed using the function "*specClust*" in the R library "*kknn*". The number of neighbors considered was 15. Hierarchical Clustering (HC) (22) was performed using Euclidean distance and Ward's method of aggregation. Clustering was performed using the function "*hclust*" in the R library "*stats*". *k*-medoids clustering (23) was performed using the function "*pam*" in the R library "*cluster*". *k*-means clustering was performed using the function "*kmeans*" in the R library "*stats*". The maximum number of iterations allowed was 100, and 100 random sets were chosen. Affinity Propagation (AP) (24) was performed using the function "apclusterK" in the R library "apcluster".

Adjusted Rand Index (ARI) (25) was used to compare the performances of different methods. ARI is frequently used in cluster validation since it is a measure of agreement between two partitions. ARI was calculated using the function "*adjustedRandIndex*" of the R library "*mclust*".

Non-linear datasets

The problem of non-linear dimensionality reduction is illustrated in Fig. 2A and in Fig. S6 for three-dimensional data sampled from two-dimensional manifolds. The Swiss-roll is described by the following parametric equations: $x=u\times cos(u)$, $z=u\times sin(u)$; where u varies between 1.5π and 4.5π, and y varies between 0 and 21. The Helicoid is described by the following parametric equation: $x=p\times cos(u)$; $y=p\times sin(u)$; $z=u$; where u varies between $-\pi$ and π, and p varies between -1 and 1. The Dini's surface is described by the following parametric equation: $x=cos(u)\times sin(v)$; $y=sin(u)\times sin(v)$; $z=cos(v)+log[tan(v/2)]+u/5$; with $0\leq u\leq 4\pi$ and $0.01\leq v\leq 1.00$ and constants $a=1.0$ and $b=0.2$. The spiral datasets described in Fig. 2B are defined by the following parametric equation: $x=cos(u)\times(u+a)$, $y=sin(u)\times(u+a)$; where u varies between π and 4π, and a is a value sampled from a Gaussian distribution with mean=0 and deviation standard between 0 and 2 (21 samplings). For each different Gaussian distribution, 100 different datasets were created. The parameter u describes the position of a point in the body of the spiral. Lower u values correspond to data points located in the center of the spiral whilst higher u values correspond to data points located in the external part. The coefficient of determination, r^2, between the first component of each method and the u values was used to evaluate the performance of each method. A higher r^2 means that the low dimensional embedding provides an accurate description of the original data.

Missing values

Real life experiments can often generate missing values. No really satisfactory solution exists for missing data, which is why it is important to try to maximize data collection. The main ways of handling missing data in analysis are: *i)* omitting variables which have many missing values; *ii)* omitting samples which do not have complete variables; and *iii)* estimating (imputing) what the missing value were. Obviously, estimating associations using an incomplete dataset remains less efficient (*i.e.*, imprecise), because part of the data is not available. Sometimes data are missing in a predictable way that does not depend on the missing value itself but which can be predicted from other data. KODAMA applies *k*NN imputation (26) on missing values in the initial step of the algorithm. For each feature with missing values, *k*NN imputation finds the *k*NN using a Euclidean metric, confined to the columns for which that feature is not missing. Each candidate neighbor might be missing some of the coordinates used to calculate the distance. In this case, *k*NN imputation averages the distance from the non-missing coordinates. Having found the *k*NN for a feature, *k*NN imputation imputes the missing values by averaging those non-missing elements of its neighbors.

To evaluate the performance of *k*NN imputation applied on KODAMA, missing values were randomly generated on simulated datasets. Different degrees of missing values were tested (*i.e.*, 5%, 10%, 15%, 20%, and 25%). We simulated 20 datasets for each degree of missing values. The datasets were generated with 3 clusters and 5 dimensions. The number of data points for each cluster is 30. Each cluster is created from a different multivariate normal distribution with a different covariance matrix of the variables. Each covariance matrix was randomly generated with values that range between 0 and 1. The performance of *k*NN imputation was analyzed by calculating the residual variance $r^2(K_I,K)$, where K_I is the KODAMA dissimilarity matrix of the data with the missing values estimated by *k*NN imputation, and K is the KODAMA matrix of the original data. Fig. S9 shows the results.

Computational time complexity

We measured computational time experimentally on a desktop machine with a 3.06 GHz Intel Core 2 Duo and with 4 GB of 1067 MHz RAM. We used the R version 3.0.2 (2013-09-25) -- "Frisbee Sailing".

The computational complexity of the KODAMA is dominated by the multiple iterations of the cross-validation procedure and, thus, it depends proportionally on the product of the number of cross-validations performed by the time complexity of the classifier used. Among the classifiers tested, the *k*NN classifier has the lowest time complexity. The simplest *k*NN algorithm has a time complexity of $O(n \times m \times f)$, where n and m are the number of data points of the training and test set, respectively, and f is the dimensionality of the dataset. In the case of a 10-fold cross-validation $n=0.9 \times N'$ and $m=0.1 \times N'$, where N' is the overall number of data points ($N'=\varphi N$), and the time complexity of the *k*NN classifier is therefore $O(0.09 \times N'^2 \times f)$. A 10-fold cross-validation performed with *k*NN classifier has thus a time complexity of $O(0.9 \times N'^2 \times f)$. KODAMA consequently has a time complexity at most of $O(0.9 \times M \times T \times N'^2 \times f)$, where M is the number of times that the maximization of the cross-validated accuracy is repeated, and *T* is the maximum number of MC iterations.

Running time (in seconds) of each classifier implemented in KODAMA (*i.e.*, *k*NN, SVM, and PCA-CA-*k*NN) was provided for different datasets, varying the number of samples and variables.

The time complexity of the SVM and PCA-CA-*k*NN classifiers is $O(N'^2 \times f \times nc)$, where *nc* is the number of classes present in the cross-validation.

We experimentally compare the time complexity of KODAMA with the other methods on datasets with 5 variables and different number of data points (*i.e.*, 50, 100, 200, 500, and 1000) and on datasets with 50 data points and a different number of variables (*i.e.*, 50, 100, 200, 500, and 1000), respectively, as shown in Table S3 and S4.

Running time of each classifier was also calculated for some of datasets tested (*i.e.*, Swiss-roll, Lymphoma, Metabolomic, Early-Type Galaxies, and The State of the Union). The results are shown in Table S5.

References

1. De Jong S (1993) SIMPLS: An alternative approach to partial least squares regression. . *Chemom Intell Lab Syst* 18:251-263.
2. Guyon I, Weston J, Barnhill S, & Vapnik V (2002) Gene Selection for Cancer Classification using Support Vector Machines. *Mach Learn* 46:389-422.
3. Efron B (1983) Estimating the error rate of a prediction rule: improvement on cross-validation. *J Am Stat Assoc* 78:316-331.
4. Molinaro AM, Simon R, & Pfeiffer RM (2005) Prediction error estimation: a comparison of resampling methods. *Bioinformatics* 21:3301-3307.
5. Ripley BD (1987) Stochastic Simulation. (Wiley), p 98.
6. Shannon CE (1948) A mathematical theory of communication. *Bell Syst Tech J* 27:379-423.
7. Coifman RR*, et al.* (2005) Geometric diffusions as a tool for harmonic analysis and structure definition of data: diffusion maps. *Proc Natl Acad Sci USA* 102(21):7426-7431.
8. Tenenbaum JB, de Silva V, & Langford JC (2000) A global geometric framework for nonlinear dimensionality reduction. *Science* 290(5500):2319-2323.
9. Ringner M (2008) What is principal component analysis? *Nature biotechnology* 26(3):303-304.
10. Roweis ST & Saul LK (2000) Nonlinear dimensionality reduction by locally linear embedding. *Science* 290(5500):2323-2326.
11. Kayo O (2006) Locally Linear Embedding Algorithm: Extensions and Application. (University of Oulu).
12. Breiman L (2001) Random Forests. *Mach Learn* 45:5-32.
13. Seligson DB*, et al.* (2005) Global histone modification patterns predict risk of prostate cancer recurrence. *Nature* 435(7046):1262-1266.
14. Sammon JWJ (1969) A nonlinear mapping for data structure analysis. *IEEE Trans Comp* C-18(5):401-409.
15. Agrafiotis DK (2003) Stochastic proximity embedding. *Journal of computational chemistry* 24(10):1215-1221.
16. van der Maaten LJP & Hinton GE (2008) Visualizing High-Dimensional Data Using t-SNE. *J Mach Learn Res* 9:2579-2605.
17. Shieh AD, Hashimoto TB, & Airoldi EM (2011) Tree preserving embedding. *Proc Natl Acad Sci USA* 108(41):16916-16921.
18. Davies DL & Bouldin DW (1979) A cluster separation measure. *IEEE Trans Pattern Anal Machine Intell* 1:224-227.
19. Bouveyron C, Girard S, & Schmid C (2006) High-Dimensional Data Clustering. *Comput Stat Data Anal* 52:502-519.
20. Ng AY, Jordan MI, & Weiss Y (2001) On Spectral Clustering: Analysis and an algorithm. *Adv Neural Inf Proc Syst* 14:849-856.
21. Malik J (2000) Normalized cuts and image segmentation. *IEEE Trans Patt Anal Mach Intell* 22:888-905.
22. Eisen MB, Spellman PT, Brown PO, & Botstein D (1998) Cluster analysis and display of genome-wide expression patterns. *Proc Natl Acad Sci USA* 95(25):14863-14868.
23. Kaufman L & Rousseeuw PJ (1990) *Finding groups in data: an introduction to cluster analysis* (Wiley, New York) pp xiv, 342 p.
24. Frey BJ & Dueck D (2007) Clustering by passing messages between data points. *Science* 315(5814):972-976.
25. Hurbet L & Arabie P (1985) Comparing partitions. *J Classif* 2(1):193-218.
26. Troyanskaya O*, et al.* (2001) Missing value estimation methods for DNA microarrays. *Bioinformatics* 17(6):520-525.

Table S1. KODAMA adjustable parameters with default values.

Parameter	Description	Default value
φ	Fraction of samples randomly taken at the beginning of step I	0.75
T	Maximum number of MC iterations	20
M	Repetitions of the iterative procedure in steps I-II-III	100
ε	Threshold for not significant proximities	0.05

Table S2. *H* value obtained with KODAMA performed with *k*NN on six different datasets: Swiss-roll, Helicoid, Dini's surface, 3-clusters datasets, and two different homogenous datasets with correlation and without correlation among the variables. Mean, standard deviation, and range of *H* values obtained with KODAMA performed with *k*NN on 100 homogenous datasets created with the same covariance matrix are reported. *p* values are also reported.

	H value	Mean	St. dev.	Min	Max	*p* value
Swiss-roll	11.391174	13.802700	1.38E-06	13.799552	13.805183	<0.01
Helicoid	10.942682	11.683730	3.44E-06	11.683730	11.683750	<0.01
Dini's surface	10.921598	11.654730	3.75E-06	11.654720	11.654740	<0.01
3-clusters	11.260296	11.369976	2.76E-05	11.359756	11.384299	<0.01
Test-1	11.384031	11.376065	1.11E-04	11.345257	11.396324	0.73
Test-2	11.397000	11.394322	2.70E-05	11.380627	11.405440	0.69

Table S3. Running times (seconds) of each method performed on homogenous Gaussian datasets with 5 variables and different numbers of data points.

	Number of data points				
	50	**100**	**200**	**500**	**1000**
KODAMA (kNN classifier)	109.199	244.892	512.025	1434.554	3563.039
KODAMA (SVM classifier)	152.425	179.740	276.022	605.145	1482.877
KODAMA (PCA-CA-kNN classifier)	120.799	246.311	497.037	1327.146	3162.133
DM	0.099	0.121	0.153	0.387	1.738
ISOMAP	0.451	1.289	6.713	85.160	783.752
PCA	0.002	0.001	0.004	0.002	0.001
LLE	2.421	5.441	12.950	62.539	309.378
RF	0.159	0.362	0.863	3.178	10.428
SAMMON	0.007	0.030	0.102	0.394	3.751
SPE	6.127	6.223	6.184	6.333	6.705
t-SNE	6.791	14.551	33.465	182.407	2453.518

Table S4. Running times (seconds) of each method performed on homogenous Gaussian datasets with 50 data points and different numbers of variables.

	Number of variables				
	50	**100**	**200**	**500**	**1000**
KODAMA (kNN classifier)	64.388	74.547	64.607	62.373	66.270
KODAMA (SVM classifier)	162.551	182.134	194.097	254.174	264.176
KODAMA (PCA-CA-*k*NN classifier)	102.248	79.182	76.313	62.370	98.220
DM	0.096	0.097	0.099	0.101	0.102
ISOMAP	0.318	0.321	0.318	0.324	0.324
PCA	0.003	0.004	0.006	0.011	0.019
LLE	3.342	3.832	4.701	7.715	11.898
RF	0.661	1.173	2.173	5.285	10.490
SAMMON	0.005	0.014	0.011	0.014	0.004
SPE	13.412	23.446	53.697	163.317	319.925
***t*-SNE**	6.943	6.940	6.864	7.758	14.091

Table S5. Running times (seconds) of each method applied to Swiss-roll, Lymphoma, Metabolomic, ETGs and State of the Union datasets.

	Swiss-roll	**Lymphoma**	**Metabolomic**	**ETGs**	**State of the Union**
Data points	1000	62	873	260	84
Variables	3	4026	416	4	834
KODAMA (kNN classifier)	3662.340	123.787	4133.893	690.069	164.217
KODAMA (SVM classifier)	1019.067	1224.028	1922.688	297.947	421.516
KODAMA (PCA-CA-*k*NN classifier)	3144.494	176.856	4560.366	659.273	193.041
DM	1.600	0.064	277.738	0.304	0.069
ISOMAP	519.844	0.486	414.901	12.816	0.871
PCA	0.006	0.113	1.261	0.001	0.048
LLE	311.742	57.809	1015.864	24.812	23.301
RF	8.111	61.781	229.911	1.131	18.357
SAMMON	2.850	0.008	2.295	0.133	0.024
SPE	6.439	1394.867	382.616	6.343	270.833
***t*-SNE**	477.211	407.68	391.755	47.790	15.816

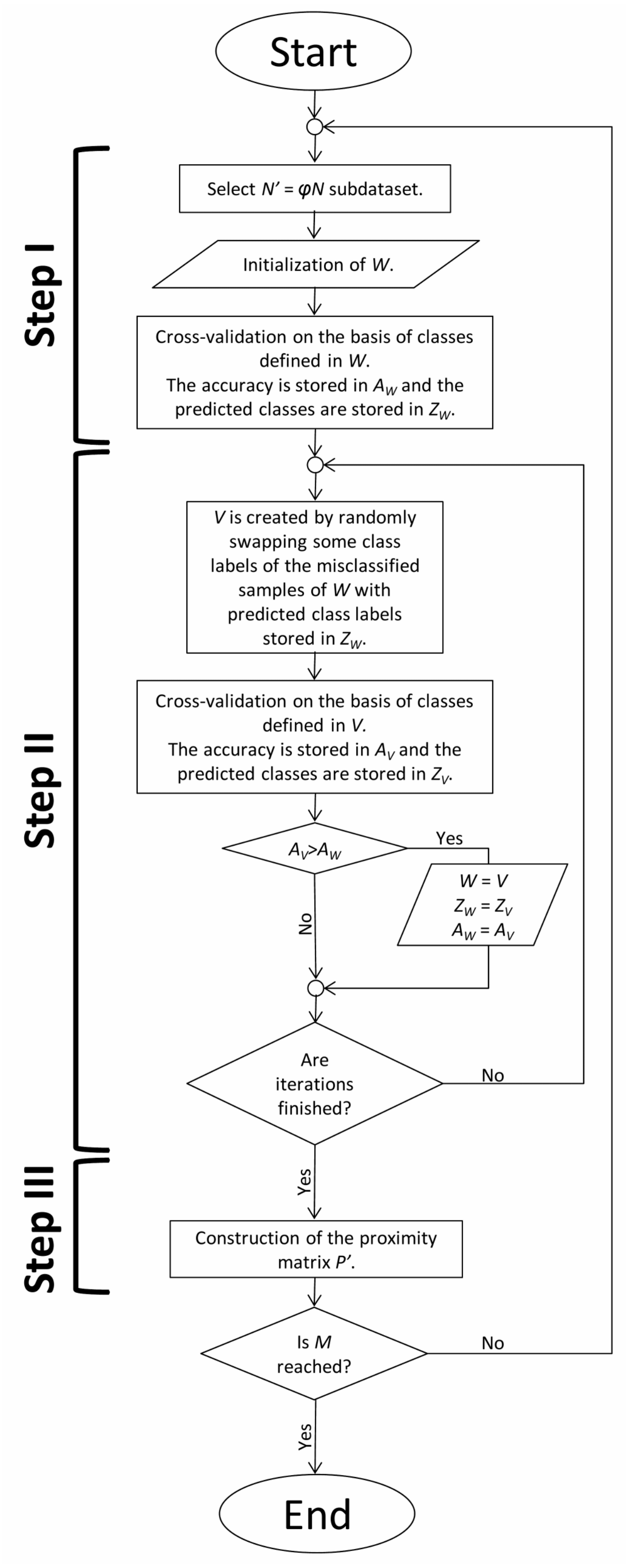
Table S6. Comparison among DBIs obtained with different feature extraction methods.

	Lymphoma	Metabolomics	ETGs (ε_e)	ETGs ($\varepsilon_{e/2}$)	State of the Union
KODAMA	0.142	0.131	0.913	0.981	0.237
DM	0.246	0.612	0.934	0.993	0.580
ISOMAP	0.784	0.408	0.902	0.988	0.356
PCA	0.307	0.521	0.942	1.134	2.260
LLE	0.186	0.349	1.256	1.247	0.750
RF	0.370	0.411	1.475	1.825	0.688
SAMMON	0.570	0.522	0.947	1.139	1.090
SPE	0.531	0.529	0.959	1.142	1.090
***t*-SNE**	1.367	0.331	3.584	3.112	2.423

Table S7. ARI values for different clustering methods applied to the Metabolomic data set.

Method	ARI
KODAMA	0.769
k-means	0.330
high-Dimensional Data Clustering	0.358
Spectral Clustering	0.336
Spectral clustering (k-nearest neighbor graph based)	0.439
Hierarchical Clustering	0.305
k-medoids	0.317
Affinity Propagation	0.212

Fig. S1. Flowchart of the first part (steps I-III) of KODAMA.



Start
Step I
Select $N' = \varphi N$ subdataset.
Initialization of W.
Cross-validation on the basis of classes defined in W. The accuracy is stored in A_W and the predicted classes are stored in Z_W.
Step II
V is created by randomly swapping some class labels of the misclassified samples of W with predicted class labels stored in Z_W.
Cross-validation on the basis of classes defined in V. The accuracy is stored in A_V and the predicted classes are stored in Z_V.
$A_V > A_W$
Yes
$W = V$
$Z_W = Z_V$
$A_W = A_V$
No
Are iterations finished?
No
Yes
Step III
Construction of the proximity matrix P'.
Is M reached?
No
Yes
End

Fig. S2. The KODAMA accuracy maximization procedure. Each point is colored according to the cluster it belongs to; the circle represents the distance to the second nearest neighbor. A_W values show how the relative cross-validated accuracy increases during the iterative step. Vector W indicates the class. Vector Z_W indicates the predicted values of the classifier built on the base of the vector W.

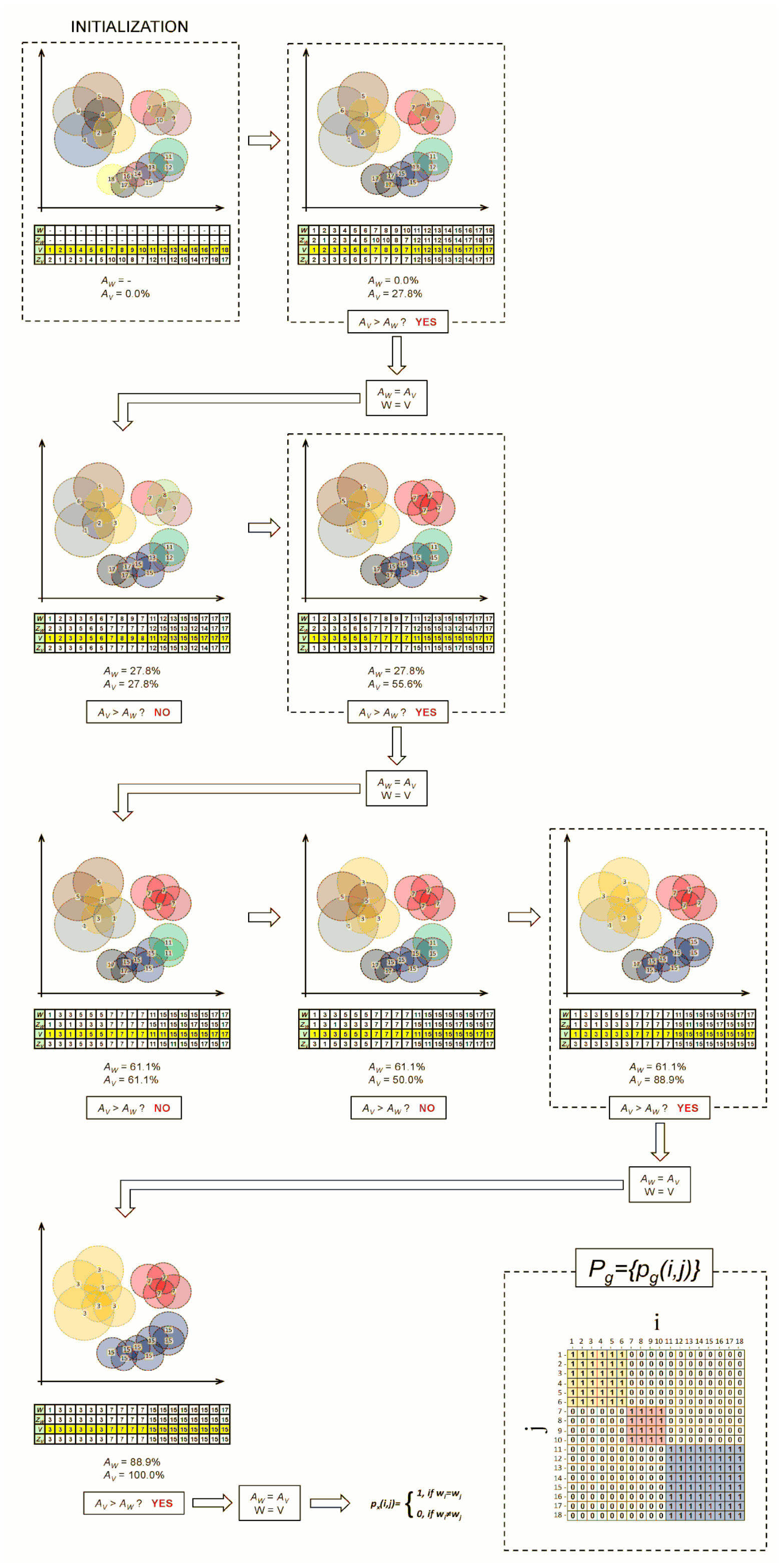


Fig. S3. The averaging of each element *p(i,j)* of the *M* proximity matrices *P* is performed with the following formula $p_M(i,j)=\sum_{g=1}^{M} p_g(i,j) \Big/ m(i,j)$ where *m(i,j)* indicates the number of times that samples x_i and x_j are present together in the same subdataset generated in step I. Thus, the resulting elements of the matrix $P_M=\{p_M(i,j)\}$ ($N \times N$) are averages ranging from 0 to 1.

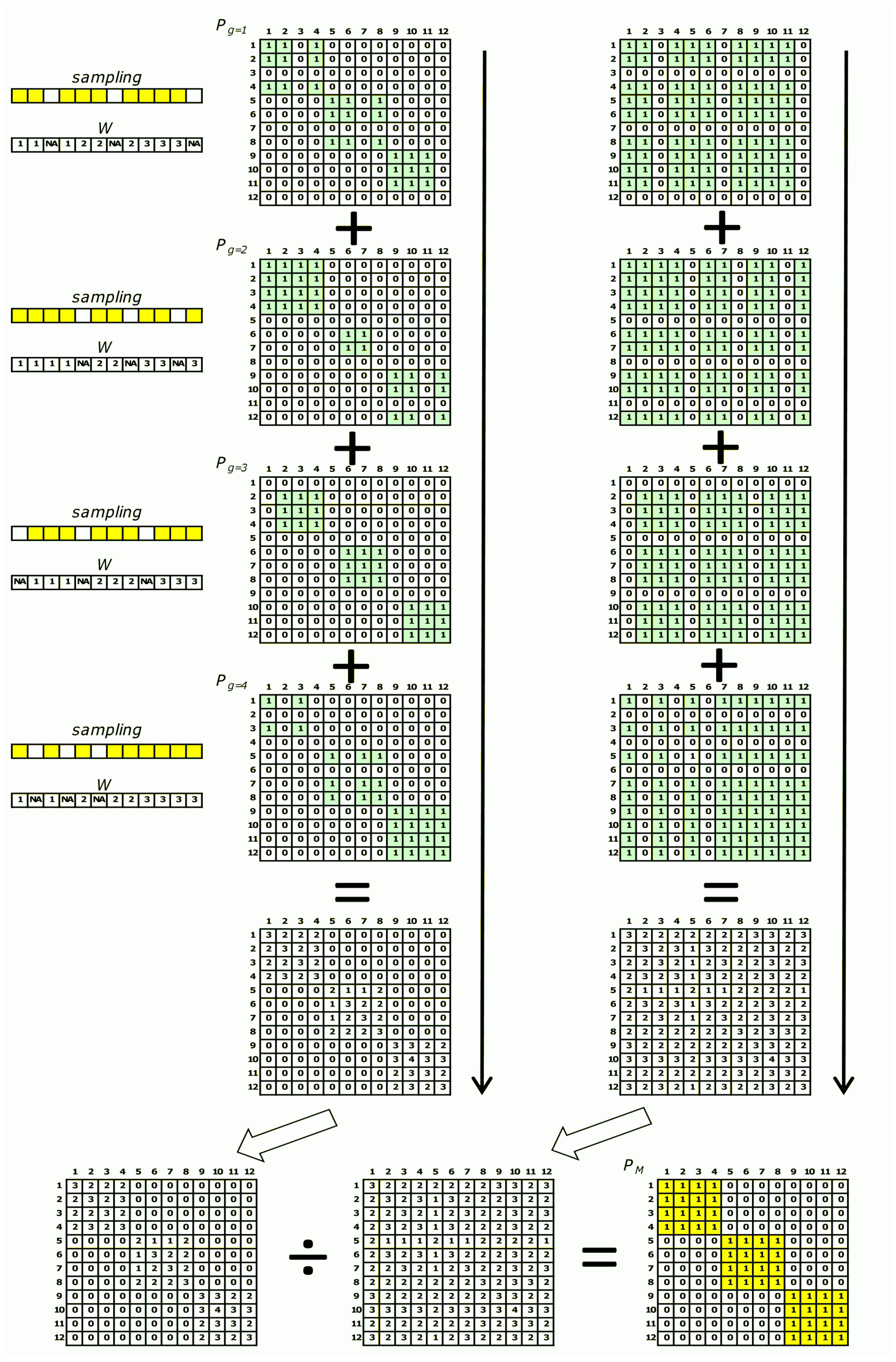


Fig. S4. Parameter optimization. KODAMA was tested on three different datasets to optimize the values of *T*, *M* and *φ* parameters. The increase in accuracy with increasing *T* or *M*, respectively, is shown. The best value of the *φ* parameter is more dependent on the type of data.

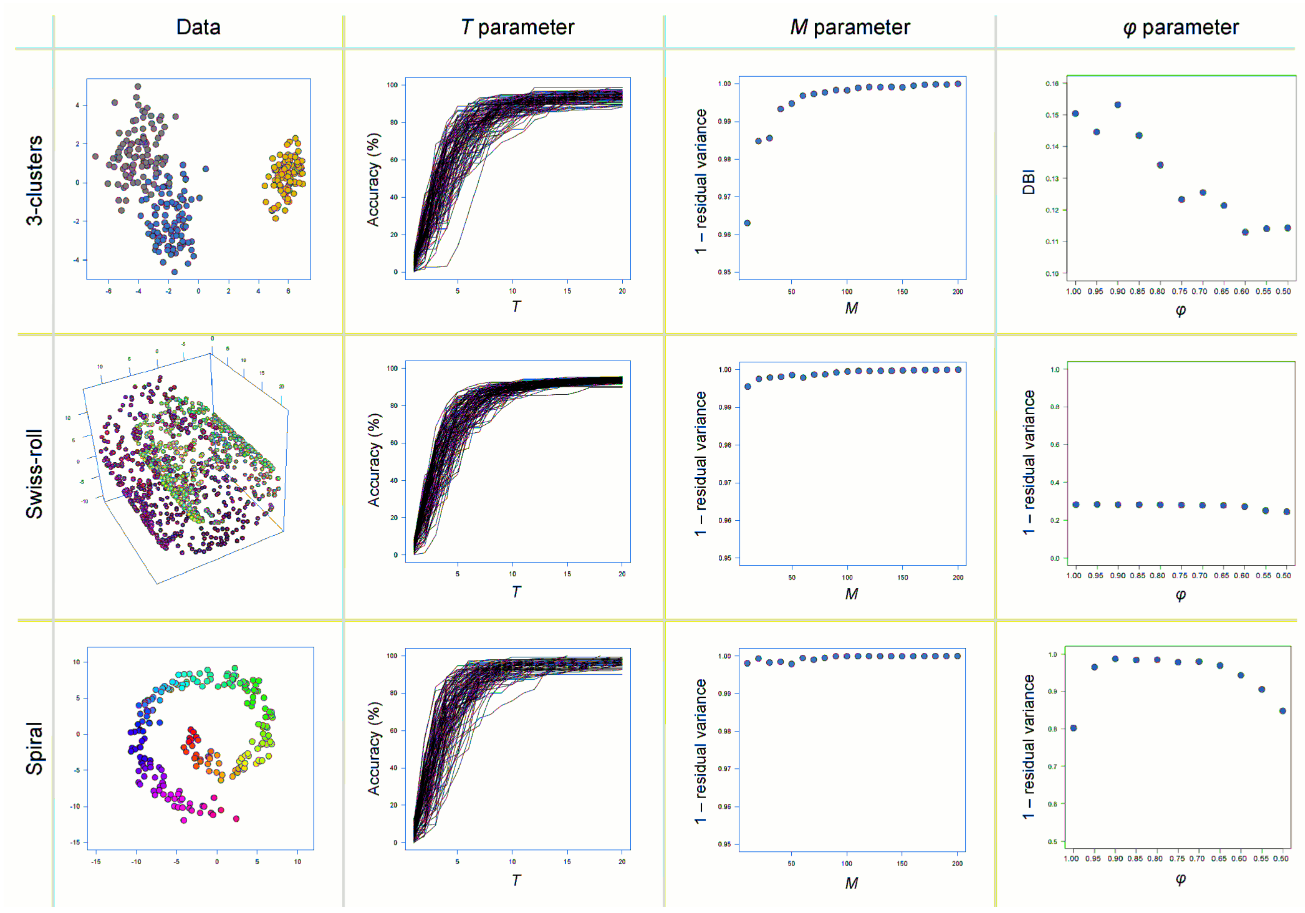


Fig. S5. KODAMA was tested on four different datasets. The first two datasets (*i.e*, Swiss-roll and 3-clusters) present a clear "organization" in the distribution of the data points. The last two datasets are continuous distributions with correlations (*i.e.*, Test-1) and without correlations (*i.e.*, Test-2) among the variables. In the first column the KODAMA proximity matrices obtained on the respective dataset are reported. In the second column the MDS plot of the KODAMA dissimilarity matrix is shown.

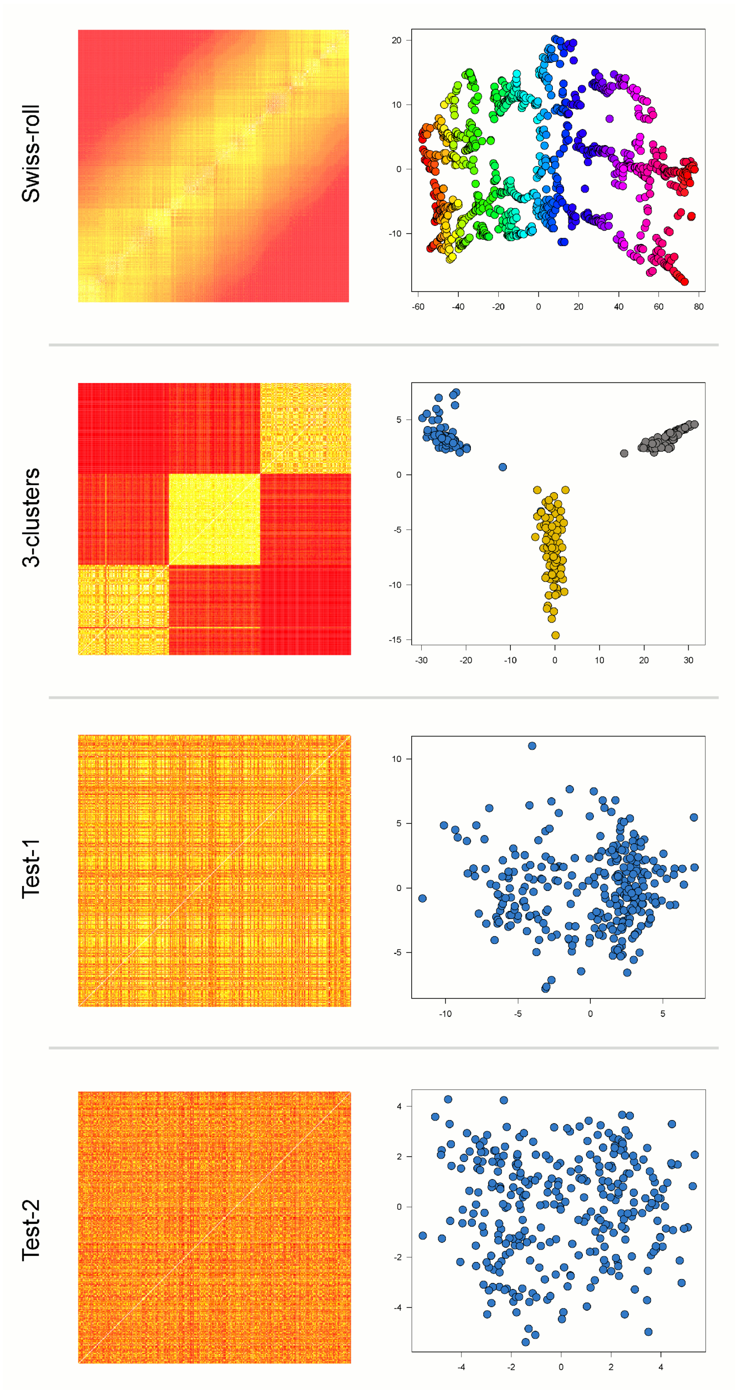


Fig. S6. Comparison between different feature extraction methods on the Swiss-roll, Helicoid, and Dini's surface. The methods shown are DM, ISOMAP, PCA, LLE, RF, Sammon, SPE, and *t*-SNE. The color-coding reveals how the data are embedded in two dimensions.

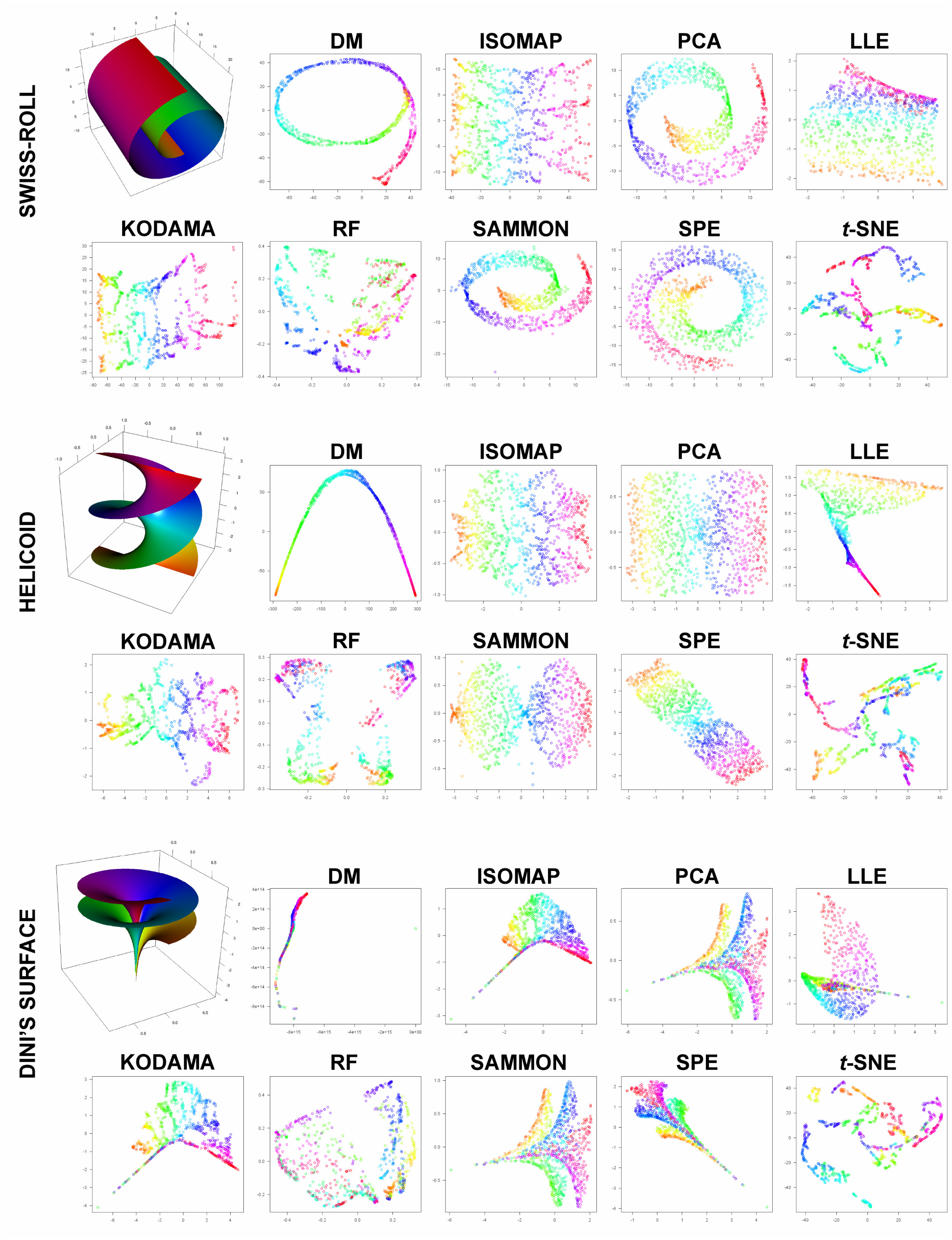


Fig. S7. Performance of achieving a low-dimensional representation from a manifold embedded in high dimensional space as a function of the noise in the Swiss-roll and Helicoid datasets of 500 data points each. KODAMA (in blue), ISOMAP (in yellow), and LLE (in red) were applied to Swiss-roll and Helicoid datasets. The Swiss-roll is described by the following parametric equations in three dimensions: $x=(u+a_1)\times cos(u)$, $z=(u+a_2)\times sin(u)$; where u varies between 1.5π and 4.5π, and y varies between 0 and 10. The values a_1 and a_2 are from a Gaussian distribution with mean=0 and standard deviation between 0 and 2 (21 samplings). For each Gaussian distribution, we created 100 different datasets. The Helicoid is described by the following parametric equation: $x=p\times cos(u)$; $y=p\times sin(u)$; $z=u+a$; where u varies between $-\pi$ and π, and p varies between -1 and 1. The value a is from a Gaussian distribution with mean=0 and standard deviation between 0 and 1 (21 samplings). For each Gaussian distribution, we created 100 different datasets.

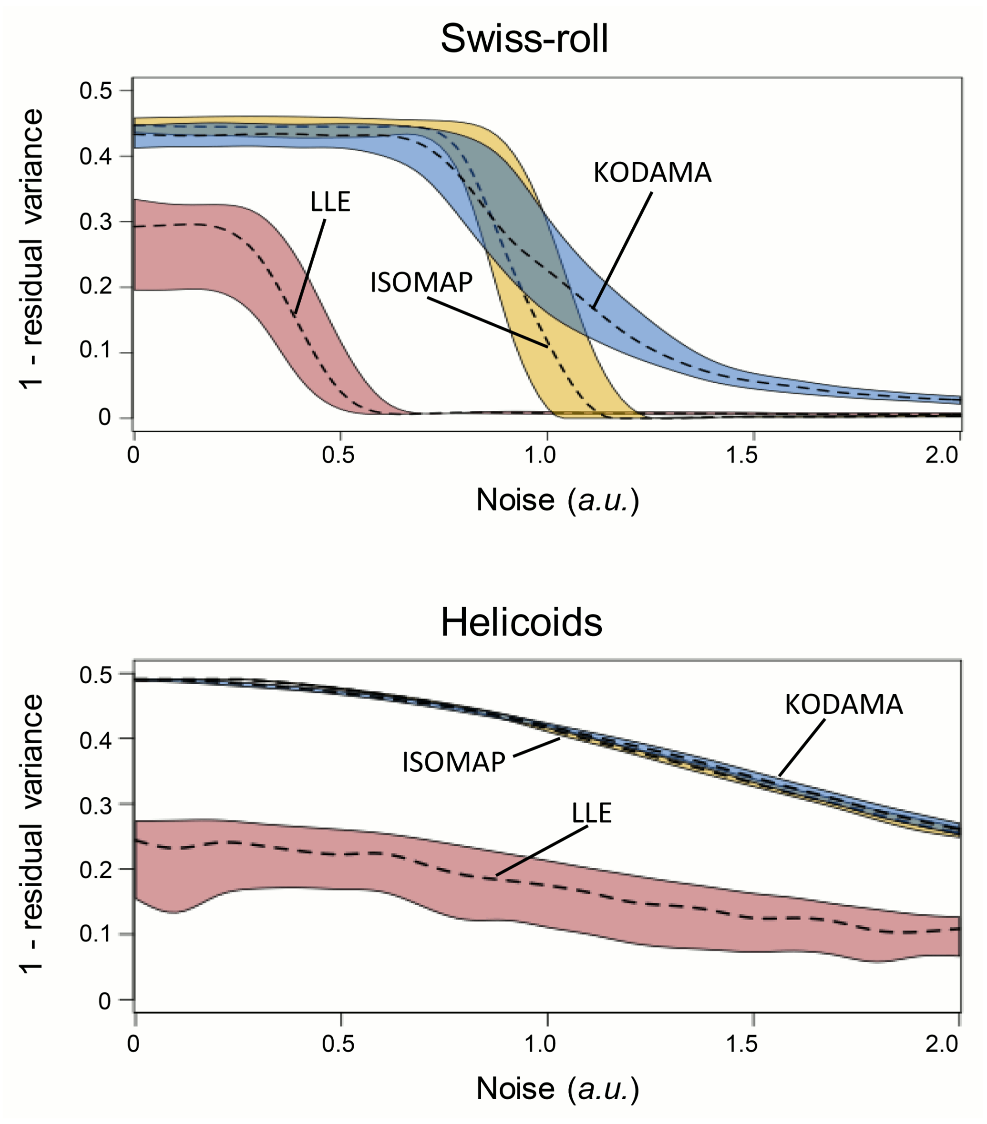


Fig. S8. Results of different methods applied to datasets with different degrees of separation among clusters.

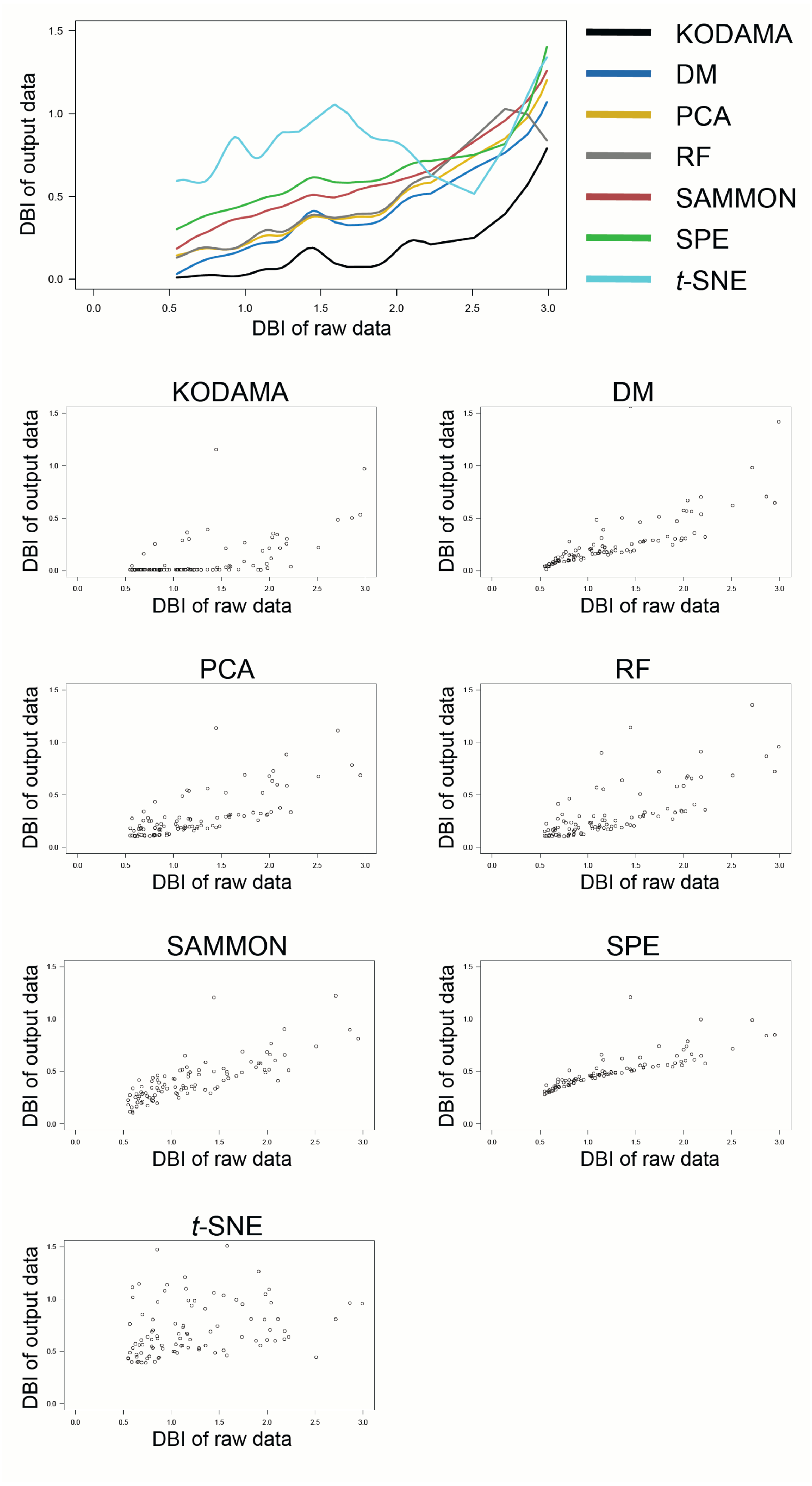


Fig. S9. Results of different methods applied to datasets with different degrees of missing values.

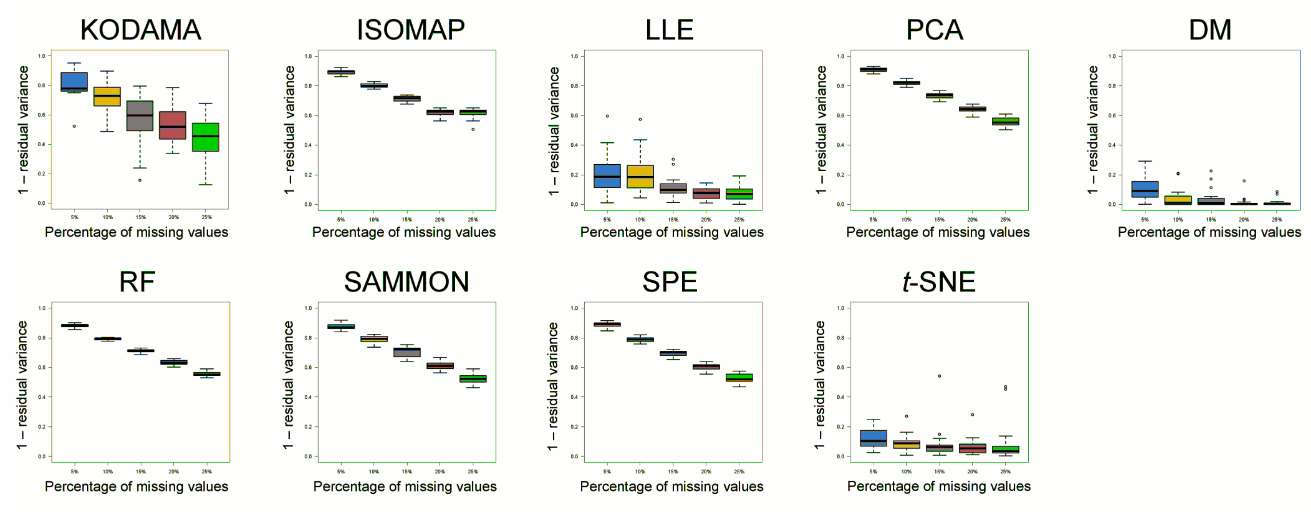


Fig. S10. Comparison between DM, RF, Sammon, SPE, and *t*-SNE on the Lymphoma, Metabolomic, and ETGs datasets. Data points are colored by their class.

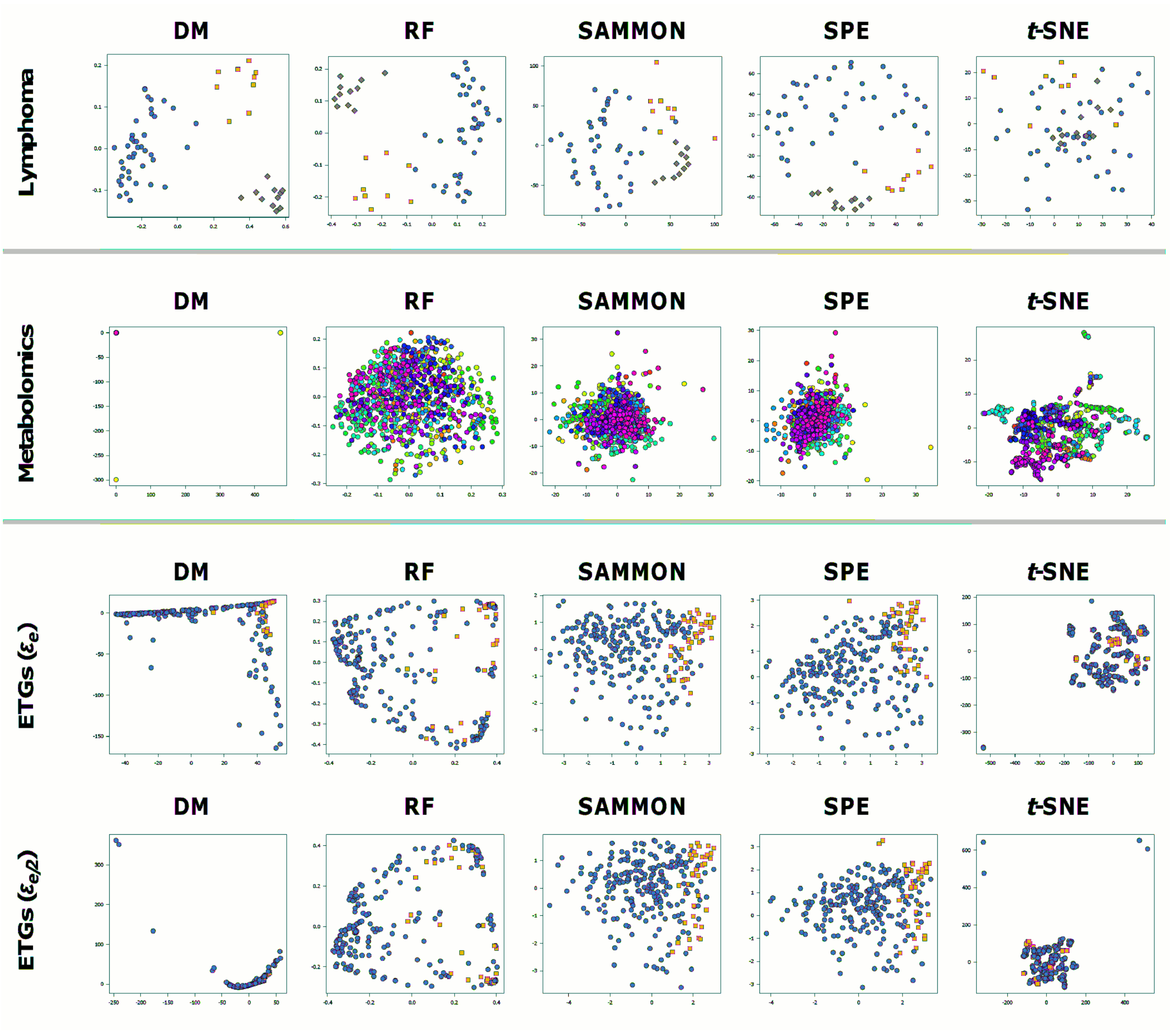


Fig. S11. Metabolomic dataset. Different visualization of KODAMA dissimilarity matrix with MDS, *t*-SNE, and TPE compared to visualization with Euclidean distance matrix with MDS, *t*-SNE, and TPE.

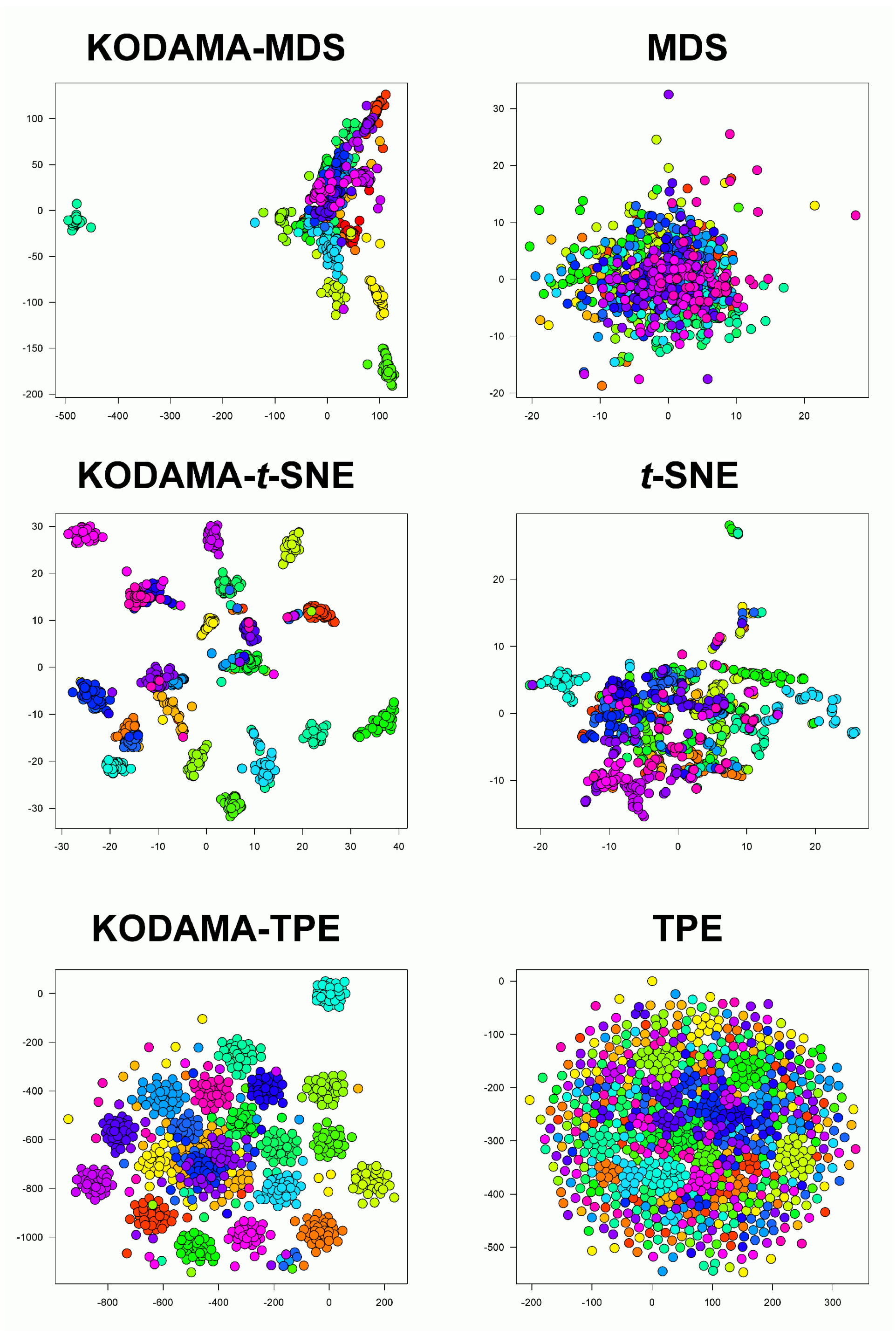


Fig. S12. Metabolomic dataset. Semi-supervised PCA-CA and KODAMA showing unsupervised gender discrimination. PCA-CA-*k*NN classifier for KODAMA was selected by minimizing the *H* value. MDS was used to visualize the results of KODAMA dissimilarity matrix.

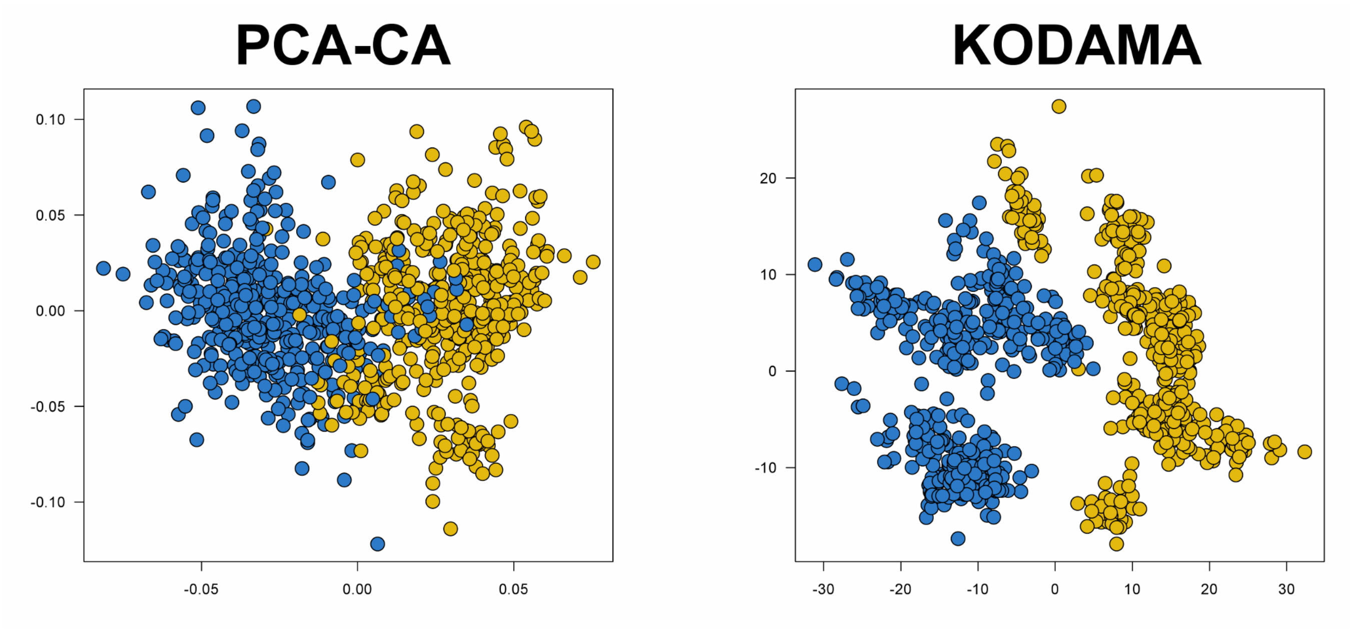


Fig. S13. ETGs dataset. Comparison between KODAMA performed with *k*NN, PCA, ISOMAP, and LLE. Color-coding indicates samples from the same class. The results of the other methods are shown in Fig. S10.

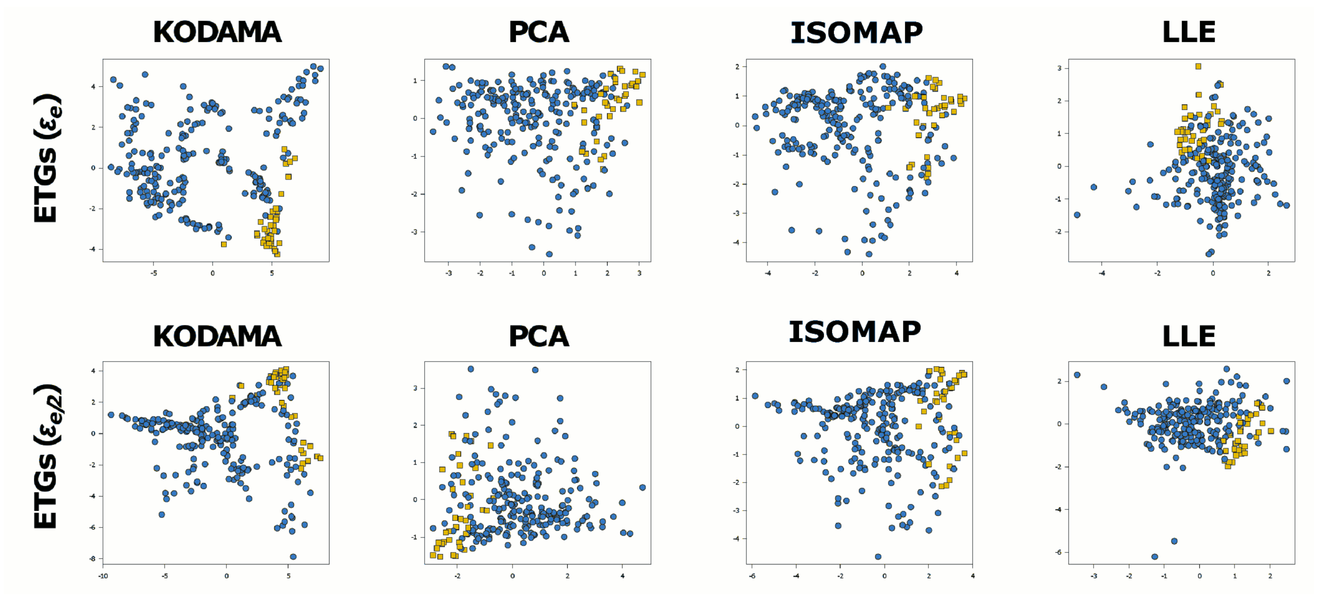
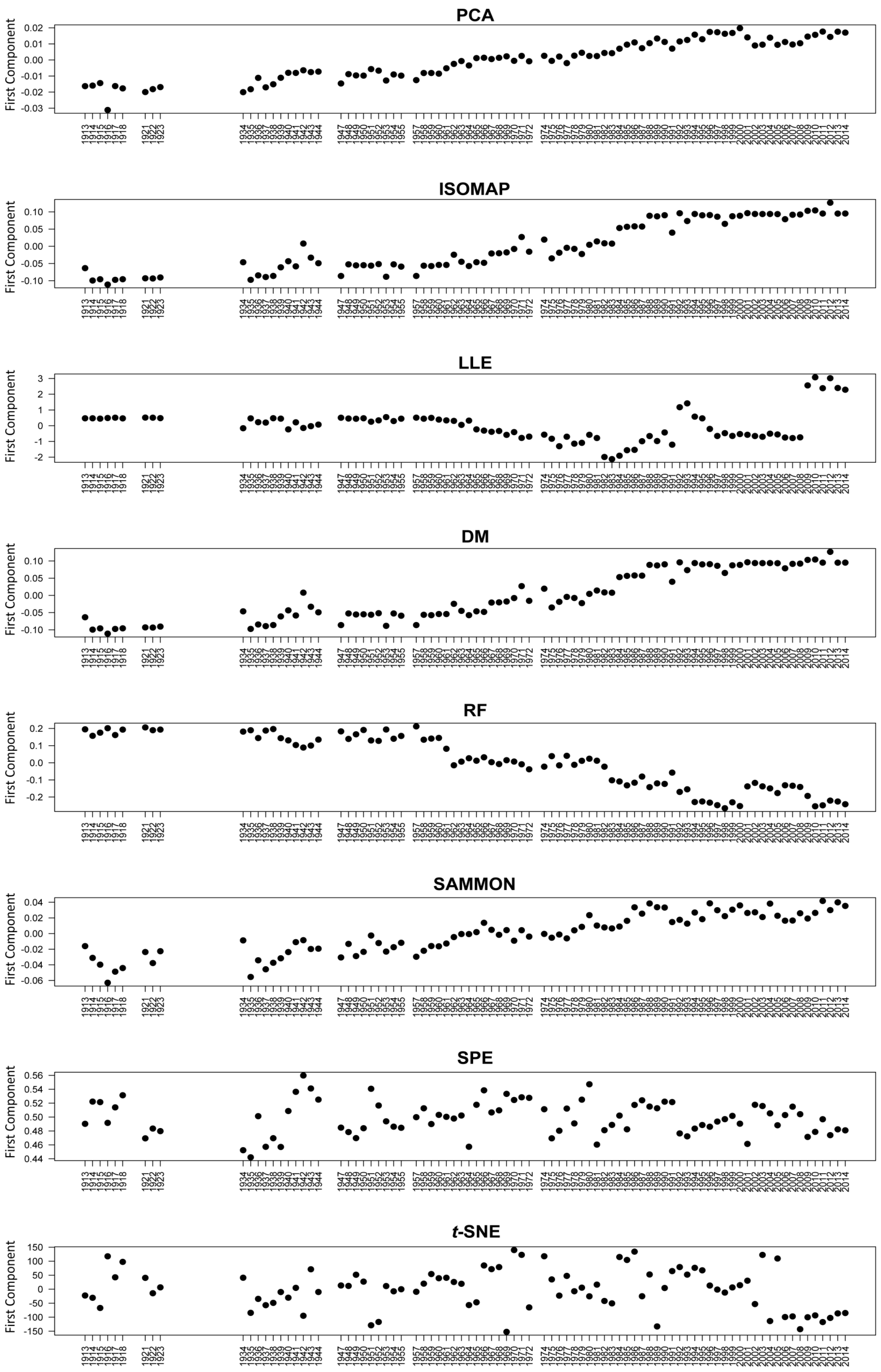


Fig. S14. First component of DM, ISOMAP, PCA, LLE, RF, Sammon, SPE, and *t*-SNE applied to the selected addresses of American presidents, in chronological order.


PCA
ISOMAP
LLE
DM
RF
SAMMON
SPE
t-SNE
First Component