Supplemental Material for DRUID$_{JS}$ — A JavaScript Library for Dimensionality Reduction

Rene Cutura*
TU Wien, University of Stuttgart

Christoph Kralj†
University of Vienna

Michael Sedlmair‡
University of Stuttgart

Abstract

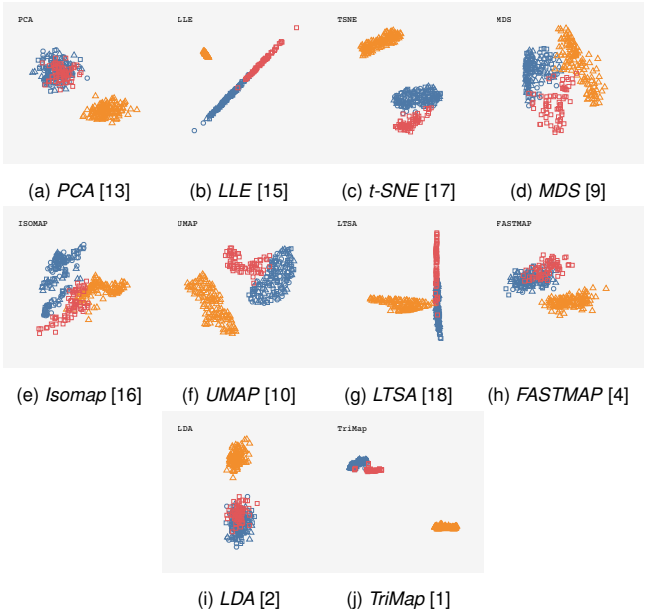
In the supplemental materials we provide example DR results using our use case dataset and the detailed results of our runtime evaluation. The performance analysis of each DR method is shown in a separate figure providing detailed information about the runtime in relation to the dataset size and the dimensionality of the dataset.

Index Terms: Software and its engineering—Software notations and tools—Software libraries and repositories

A Examples

Fig. 1 shows the projections of the *Palmer Penguin* [5] dataset with each of DRUID$_{JS}$'s DR method with its default parameters. For *LDA* [2] the species are used as classes.

```
new druid.DR(penguins).transform();
```



(a) *PCA* [13] (b) *LLE* [15] (c) *t-SNE* [17] (d) *MDS* [9]

(e) *Isomap* [16] (f) *UMAP* [10] (g) *LTSA* [18] (h) *FASTMAP* [4]

(i) *LDA* [2] (j) *TriMap* [1]

Figure 1: Visualizations of projections with each DR method in DRUID$_{JS}$ of the 4D *Palmer Penguin* dataset, which consists of 342 different specimens of penguins of the species ● Adelie, ● Gentoo, and ● Chinstrap, living on one of the islands ○ Torgersen, △ Biscoe, or □ Dream of the Antarctica.

*e-mail: rene.cutura@tuwien.ac.at
†e-mail: christoph.kralj@univie.ac.at
‡e-mail: michael.sedlmair@visus.uni-stuttgart.de

B Evaluation

We performed three evaluations to compare DRUID$_{JS}$ with state-of-the-art methods. First we did a runtime analyis of ● DRUID$_{JS}$ under different scales of data. Second, a runtime comparison to Scikit-learn [14] and UMAP-learn [11] (as ● `sklearn*`), and to the existing JavaScript DR libraries [6–8] (as ● `js`). Third, a case study comparing usability and runtime of DRUID$_{JS}$ to existing JavaScript DR libraries.

Due to the page limit, we put just six of the ten DRUID$_{JS}$ DR methods in the paper. Here in the supplemental material, we provide (a) the data of all 10 methods as well as the actual average runtimes (to enrich the color aggregations found in the paper).

The figures show a legend for the colorscale, then an overview which library is the fastest on which scale of data where each cell represents a dataset of N points and D dimensions. Each figure has such a table (appearing as line). For each dataset cell, we let the available ● `sklearn*`, ● `js` (the existing JavaScript libraries), and our ● DRUID$_{JS}$ implementations run 5 times each, and measured the runtimes. We selected the two fastest runs and color-coded them. This results in a single color cell if the two fastest were from the same implementation, and in stripes if they were from two different libraries. Below that, we show the average runtime of the five runs. Each cell shows in color (gray, white to red, and hatched) the mean time of the 5 runs of the respective DR method using a generated random dataset according N points (x-axis), and D dimensions (y-axis). The gray cells ■ indicate a runtime under 100ms, which allows for on-the-fly user interactions [3, 12].

DR Method	Libraries	Figures
PCA [13]	● ● ●	1a, 2, 12b, 12c, 12d
LLE [15]	● ●	1b, 3, 12e, 12f
t-SNE [17]	● ● ●	1c, 4, 12g, 12h, 12i
MDS [9]	● ●	1d, 5, 12j, 12k
Isomap [16]	● ●	1e, 6, 12l, 12m
UMAP [10]	● ● ●	1f, 7, 12n, 12o, 12p
LTSA [18]	●	1g, 8, 12q
FASTMAP [4]	●	1h, 9, 12r
LDA [2]	●	1i, 10, 12s
TriMap [1]	●	1j, 11, 12t

Table 1: Table of evaluations with references to the Figures. Column with libraries is colorcoded: ● DRUID$_{JS}$, ● `js` [6–8], ● `sklearn*` [11, 14].

References

[1] E. Amid and M. K. Warmuth. TriMap: Large-scale Dimensionality Reduction Using Triplets, 2019.

[2] D. M. Blei, A. Y. Ng, and M. I. Jordan. Latent Dirichlet Allocation. *Journal of Machine Learning Research (JMLR)*, 3:993–1022, 2003.

[3] S. K. Card, G. G. Robertson, and J. D. Mackinlay. The information visualizer, an information workspace. In *ACM Conf. on Human Factors in Computing Systems (SIGCHI)*, pp. 181–186, 1991. doi: 10.1145/108844.108874

[4] C. Faloutsos and K.-I. D. Lin. FastMap: A Fast Algorithm for Indexing, Data-Mining and Visualization of Traditional and Multimedia Datasets. In *ACM Proc. on Management of data (SIGMOD)*, pp. 163–174, 1995. doi: 10.1145/223784.223812

[5] K. B. Gorman, T. D. Williams, and W. R. Fraser. Ecological sexual dimorphism and environmental variability within a community of antarctic penguins (genus pygoscelis). *PloS one*, 9(3):e90081, 2014. doi: 10.1371/journal.pone.0090081

[6] https://github.com/karpathy/tsnejs. tSNEjs.

[7] https://github.com/machinelearnjs/machinelearnjs. MachineLearn.js.

[8] https://github.com/PAIR-code/umap js. UMAP-js.

[9] J. B. Kruskal. Multidimensional scaling by optimizing goodness of fit to a nonmetric hypothesis. *Psychometrika*, 29(1):1–27, 1964. doi: 10.1007/BF02289565

[10] L. McInnes, J. Healy, and J. Melville. UMAP: Uniform Manifold Approximation and Projection for Dimension Reduction, 2018.

[11] L. McInnes, J. Healy, N. Saul, and L. Grossberger. UMAP: Uniform Manifold Approximation and Projection. *The Journal of Open Source Software (JOSS)*, 3(29):861, 2018. doi: 10.21105/joss.00861

[12] R. B. Miller. Response time in man-computer conversational transactions. In *Fall Joint Computing Conference (AFIPS)*, pp. 267–277, 1968. doi: 10.1145/1476589.1476628

[13] K. Pearson. LIII. On lines and planes of closest fit to systems of points in space. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, 2(11):559–572, 1901. doi: 10.1080/14786440109462720

[14] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, et al. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research (JMLR)*, 12:2825–2830, 2011.

[15] S. T. Roweis and L. K. Saul. Nonlinear Dimensionality Reduction by Locally Linear Embedding. *Science*, 290(5500):2323–2326, 2000. doi: 10.1126/science.290.5500.2323

[16] J. B. Tenenbaum, V. De Silva, and J. C. Langford. A global geometric framework for nonlinear dimensionality reduction. *Science*, 290(5500):2319–2323, 2000. doi: 10.1126/science.290.5500.2319

[17] L. J. P. van der Maaten and G. Hinton. Visualizing data using t-SNE. *Journal of Machine Learning Research (JMLR)*, 9:2579–2605, 2008.

[18] Z. Zhang and H. Zha. Principal Manifolds and Nonlinear Dimensionality Reduction via Tangent Space Alignment. *SIAM Journal on Scientific Computing*, 26(1):313–338, 2004. doi: 10.1137/S1064827502419154

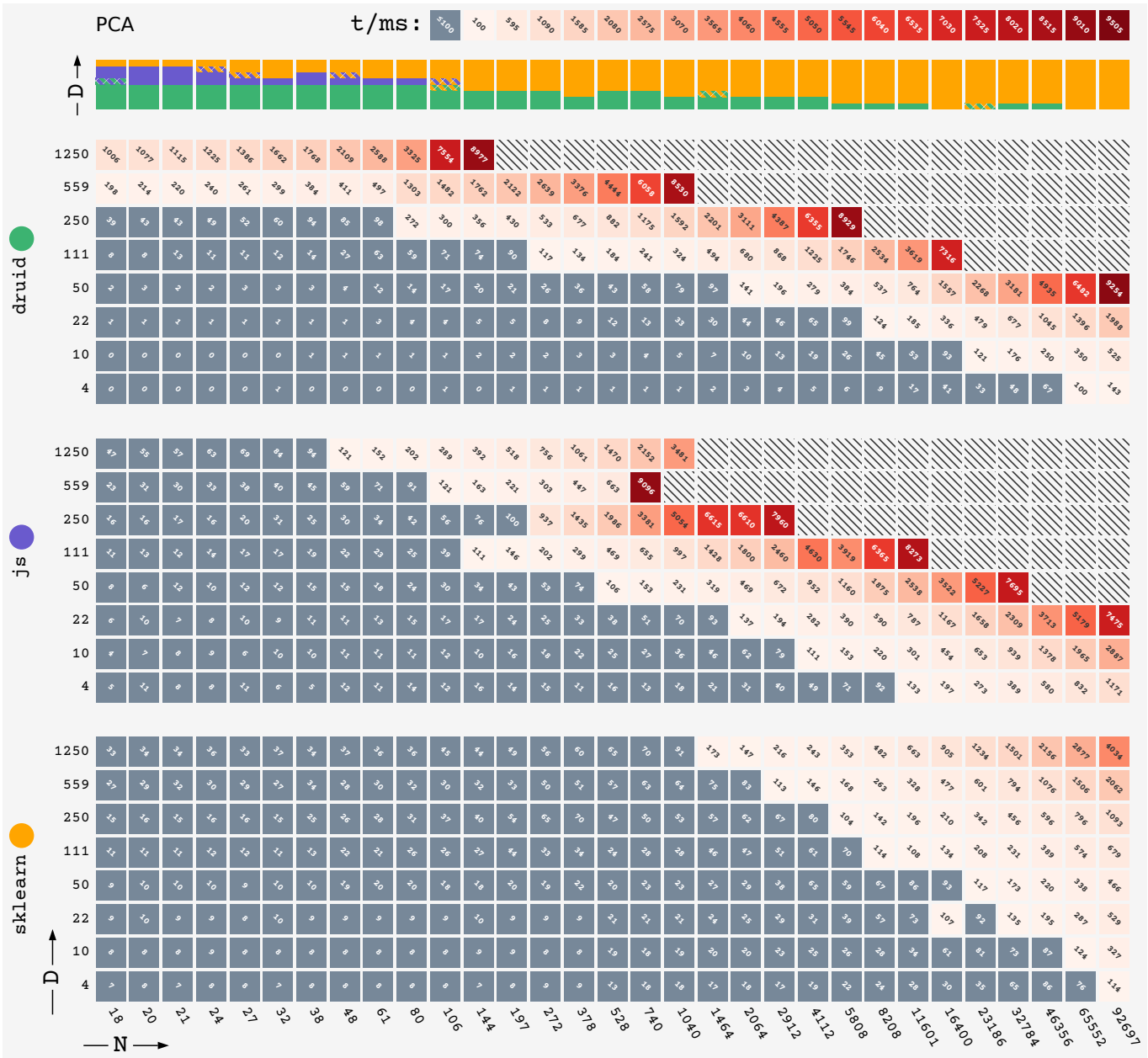


Figure 2: *PCA* [13] is a linear technique, which results in the linear projection with the highest variance of its points. DRUID$_{JS}$ uses the QR-algorithm to compute the needed eigenvectors, MachineLearn.js [7] uses the *SVD* algorithm of numeric.js to compute the principal components, as a required step the user needs to compute the projection with the principal components (Ex. 8 in the main document).

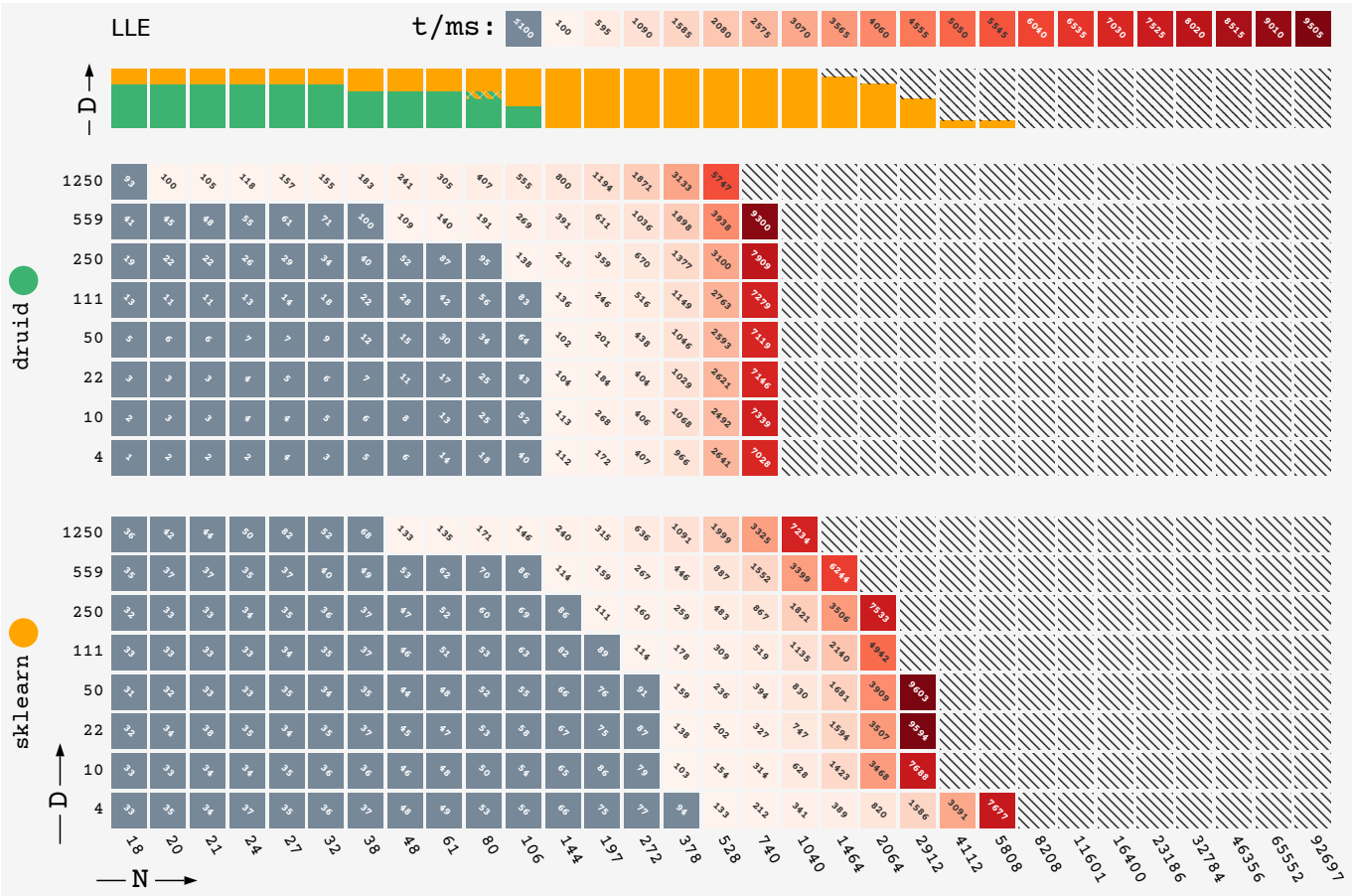


Figure 3: *LLE* [15] creates a sparse matrix, where each datapoint's entries are the coefficients needed to linearly reconstruct the datapoint. The d smallest eigenvectors of the sparse matrix represent then the projection.

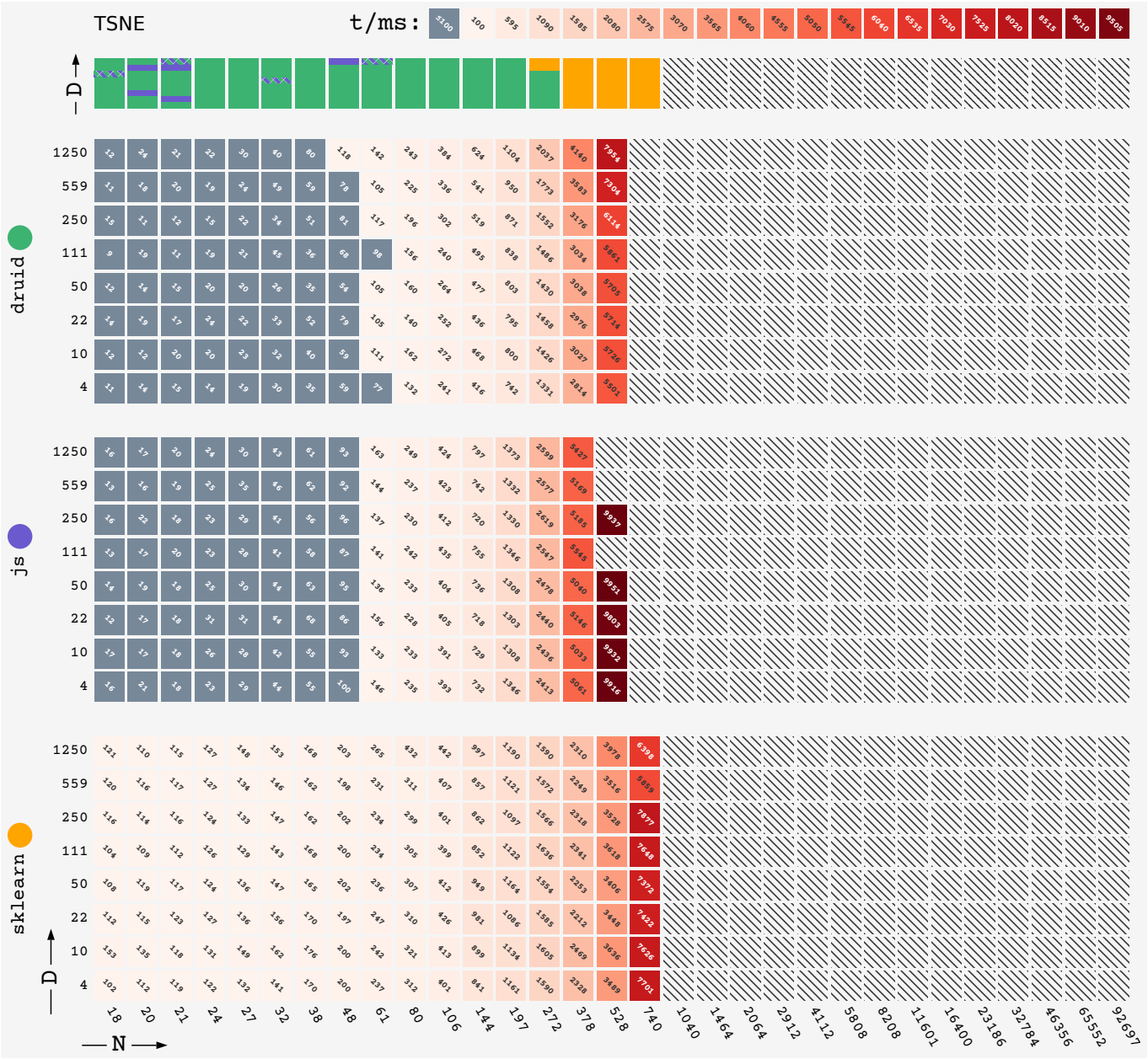


Figure 4: *t-SNE* [17] tries to preserve local distances with a stochastic approach. DRUIDJS's *t-SNE* implementation is based on the python variant of Scikit-learn and the existing JavaScript library, but tailored to work with the `druid.Matrix` class.

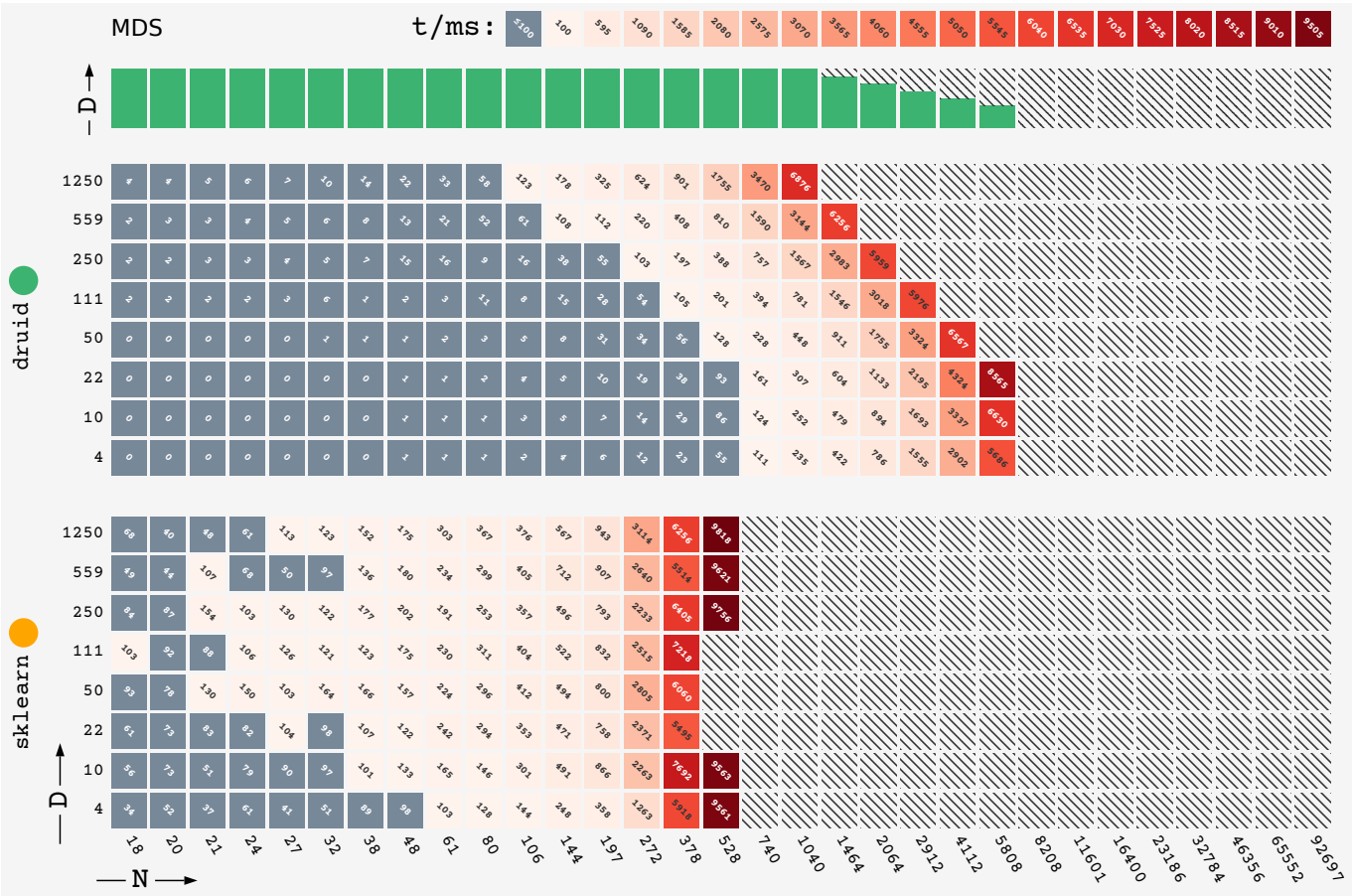


Figure 5: *MDS* [9] tries to preserve all distances of a dataset as good as possible. DRUID$_{JS}$ uses the classic *MDS* approach, computing the projection directly with a eigendecomposition. Scikit-learn uses a iterativ approach.

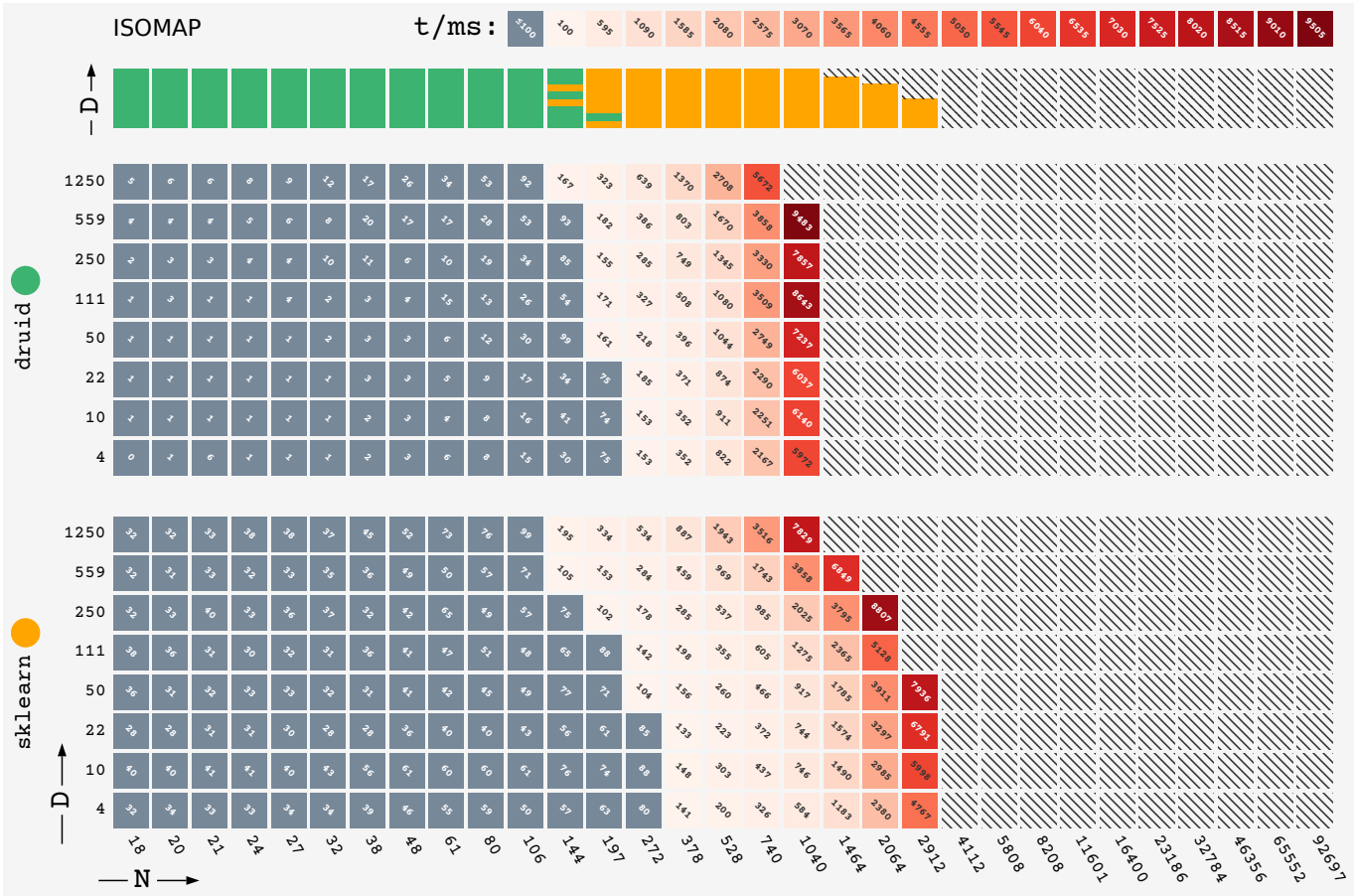


Figure 6: *Isomap* [16] computes the geodesic distances between the points on a k neighborhood graph, and applies then *MDS* [9] on this geodesic distance matrix.

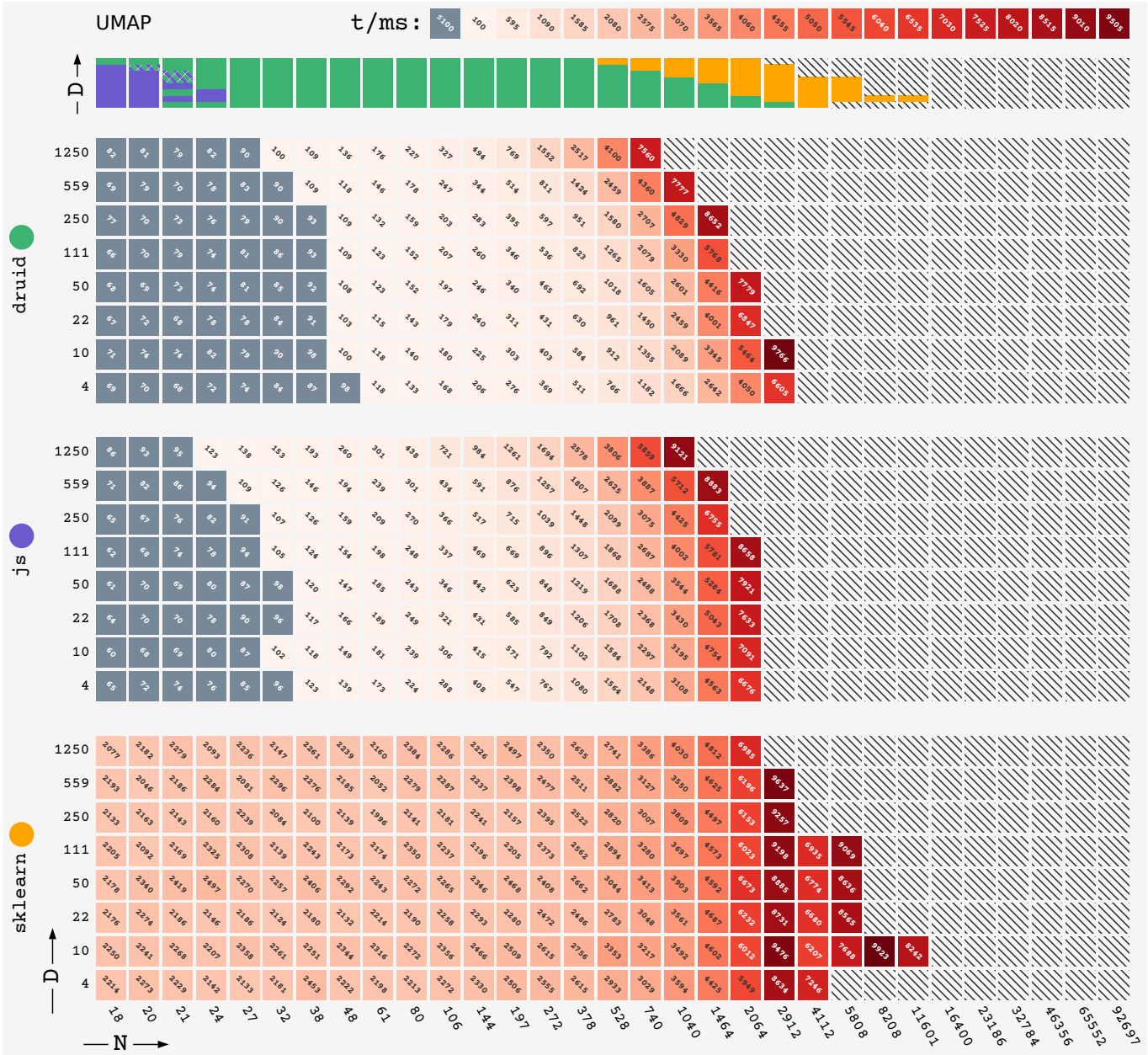


Figure 7: *UMAP* [10] tries to preserve local neighborhoods, but also respect farther distances. DRUIDJS's *UMAP* implementation is a JavaScript translation of the python variant UMAP-learn, with the difference that it starts with a random generation instead of a spectral embedding. Which is the reason for the differences in runtimes of UMAP-learn to DRUIDJS or the other JavaScript library.

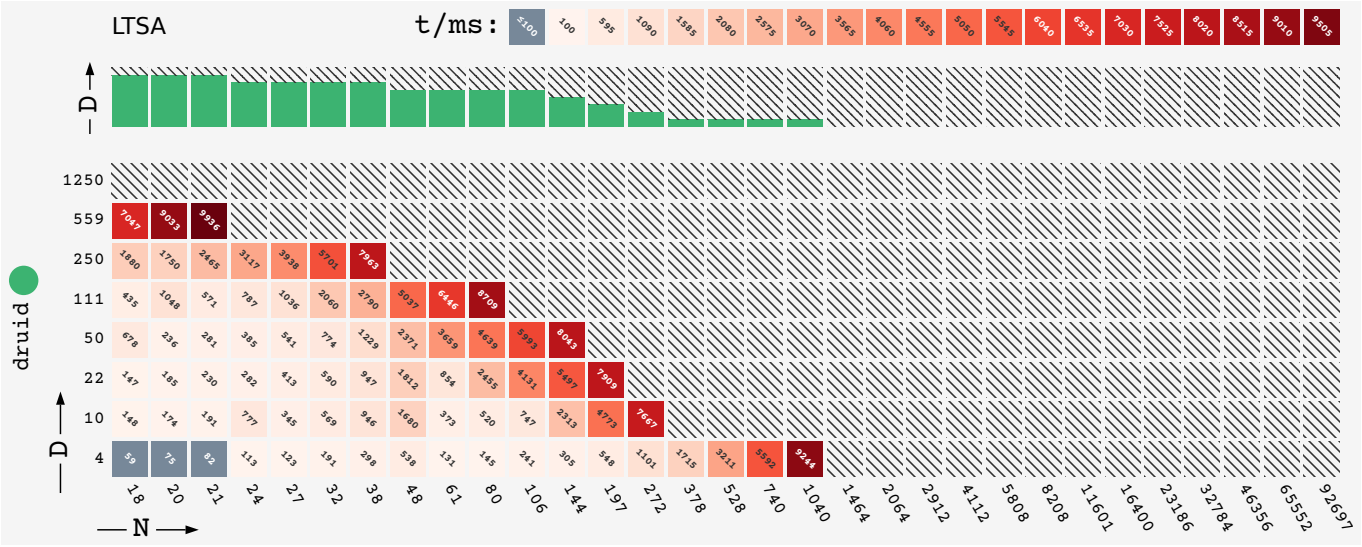


Figure 8: *LTSA* [18] approximates the tangent space at each data point. The projection is then, the alignment of those tangent spaces.

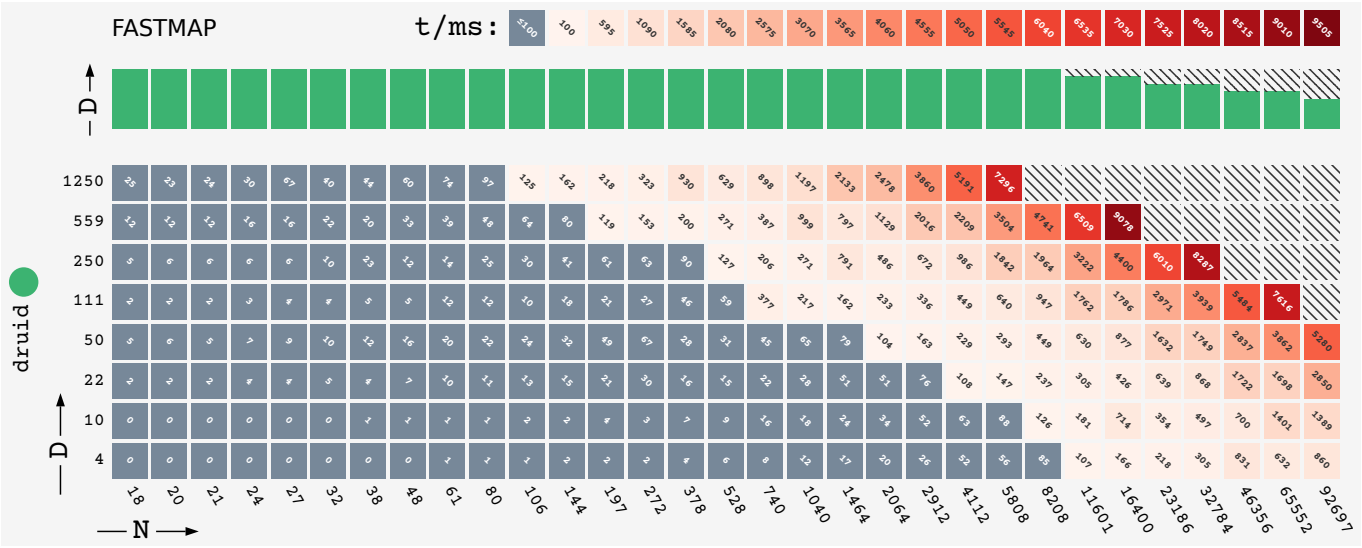


Figure 9: *FASTMAP* [4] is a linear DR technique. With each step *FASTMAP* projects the data to a line defined by the two most distant datapoints, reducing its dimensionality step by step.

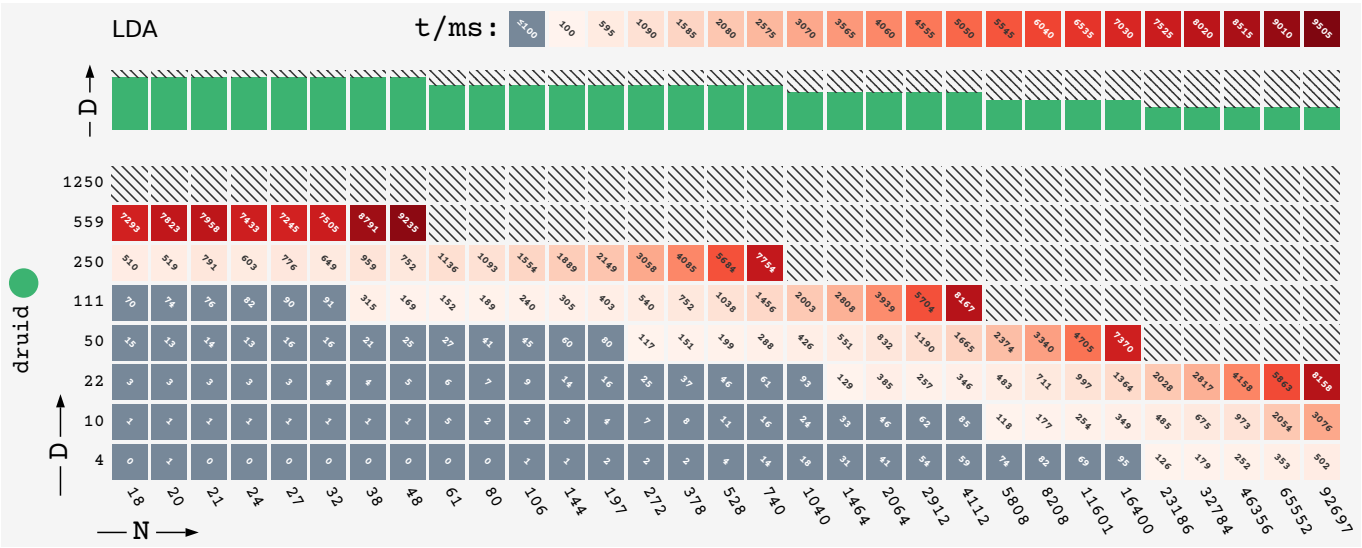


Figure 10: *LDA* [2] is a linear technique, which computes a projection of a dataset with cluster labels, in a way that clusters get preserved as good as possible while maximize the distance between the clusters.

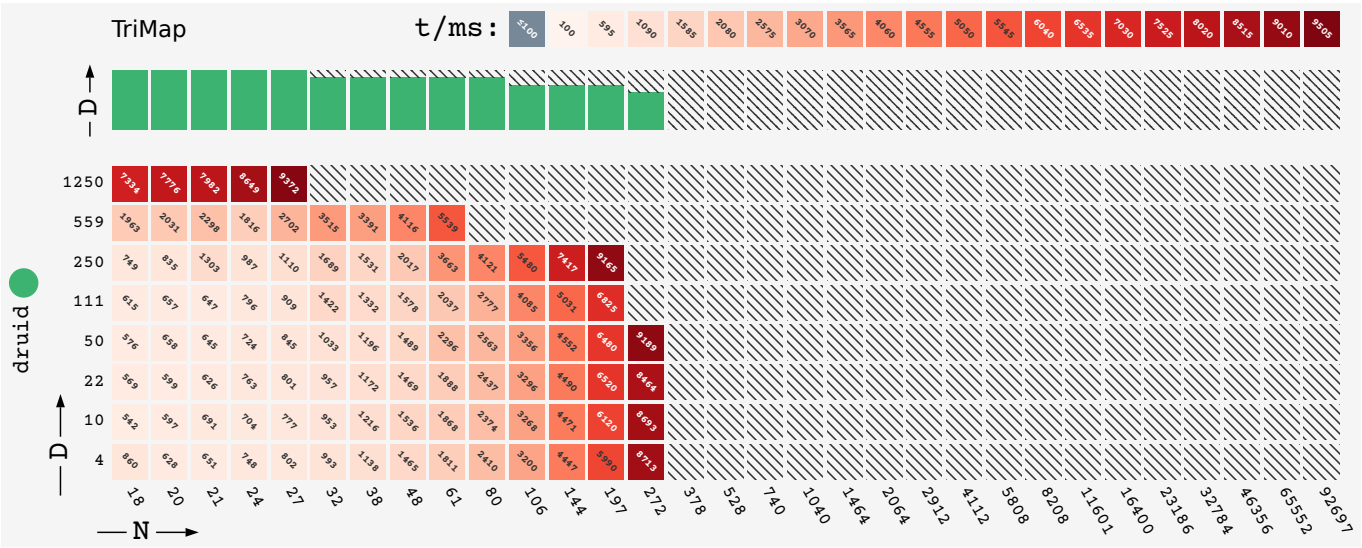
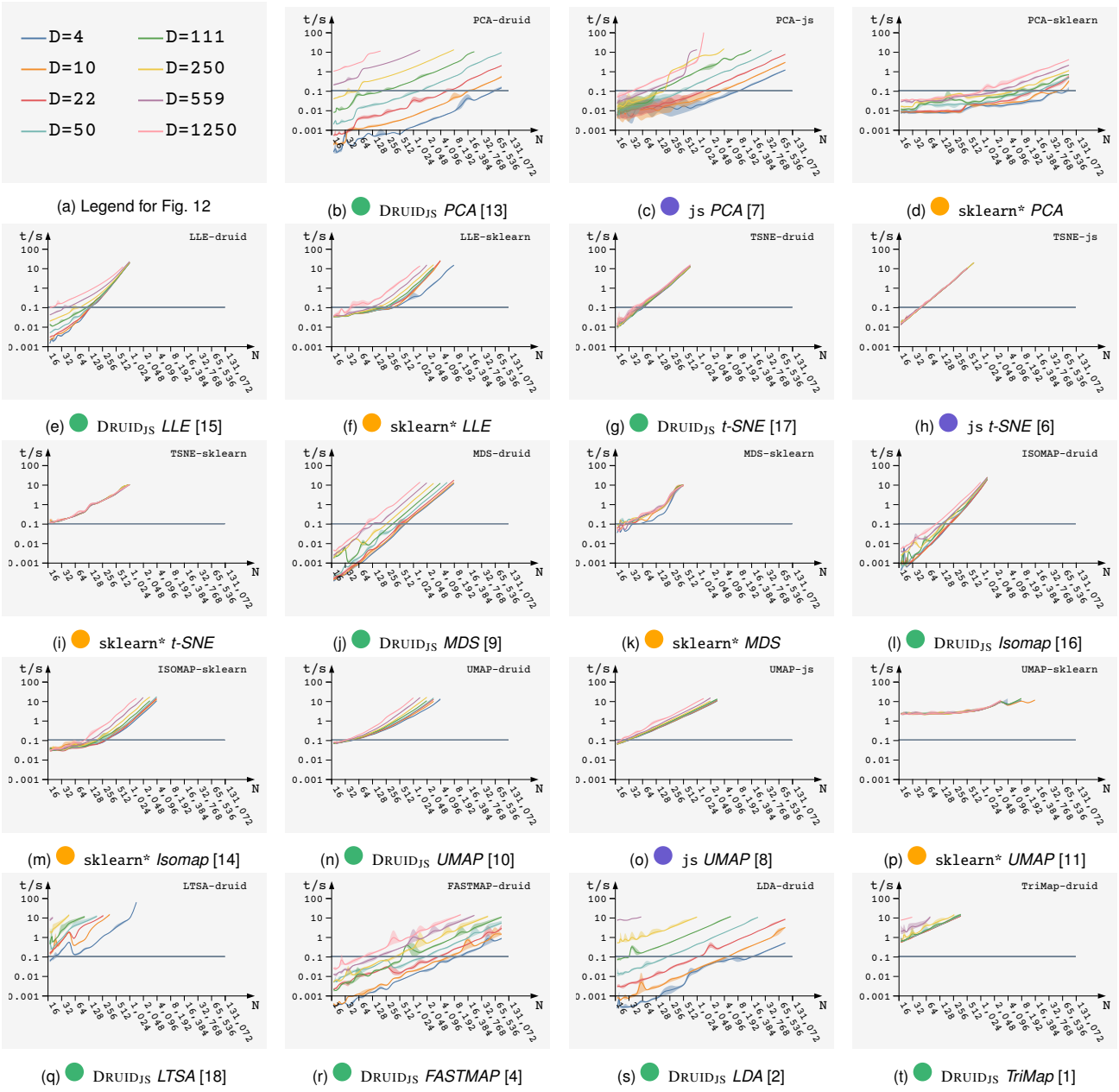


Figure 11: *TriMap* [1] tries to preserve the distances of random triplets of a dataset.



(a) Legend for Fig. 12 (b) DRUIDJS *PCA* [13] (c) js *PCA* [7] (d) sklearn* *PCA*

(e) DRUIDJS *LLE* [15] (f) sklearn* *LLE* (g) DRUIDJS *t-SNE* [17] (h) js *t-SNE* [6]

(i) sklearn* *t-SNE* (j) DRUIDJS *MDS* [9] (k) sklearn* *MDS* (l) DRUIDJS *Isomap* [16]

(m) sklearn* *Isomap* [14] (n) DRUIDJS *UMAP* [10] (o) js *UMAP* [8] (p) sklearn* *UMAP* [11]

(q) DRUIDJS *LTSA* [18] (r) DRUIDJS *FASTMAP* [4] (s) DRUIDJS *LDA* [2] (t) DRUIDJS *TriMap* [1]

Figure 12: Another visualization of the evaluation. Both axes have a log-scale, the y-axis shows the runtime, the x-axis shows the dataset length N. The colors of the lines correspond to the dimensionality D. The gray line respresents 100ms.