ScagnosticsJS: Extended Scatterplot Visual Features for the Web

V. Pham and T. Dang

Computer Science Department, Texas Tech University, Lubbock, Texas, USA

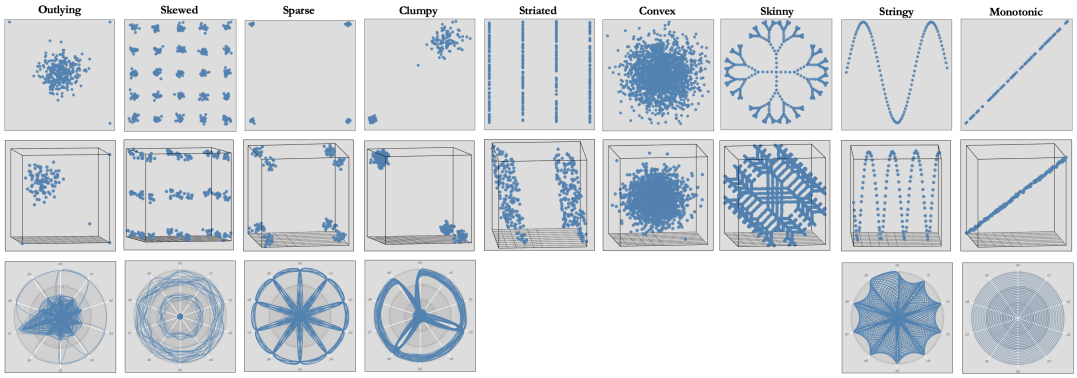

Figure 1: *The nine Scagnostics measures and their exemplar scatterplots in 2D (top row), 3D (middle row), and nD (bottom row).*

Abstract

Scagnostics is a set of features that characterizes the data distribution in a scatterplot. These visual features have been used in various applications to detect unusual correlations of bivariate data. However, there is no formally published implementation for 3D or higher. This project aims to provide the Scagnostics implementation in JavaScript, called ScagnosticsJS, and also extend these measures for higher dimensional scattered points. We also present a Scagnostics exploration webpage, which makes the underlying algorithms transparent to users.

1. Introduction

In 2005, Wilkinson [WAG05] proposed a set of the graph-theoretic summaries of scattered point data, called Scagnostics. Since then, Scagnostics has been used in different applications domains with various purposes, such as high-dimensional time series analysis [PNL*19], clustering of scatter plots [DW14a], and outlier detection [Wil18; PD19]. Two dimensional (2D) Scagnostics were mainly implemented using either R, Python, or Java programming languages, and there is no formally published library for Scagnostics implementation in JavaScript which is gaining popularity for *visualization for the web* [BOH11].

Our JavaScript library enables the flexibility of executing Scagnostics calculations either at the client sides (to reduce the server load) or server-side (e.g., using node.js) comparing to the dependence on server side calculation by wrapping existing implementations in a web service. This work also offers 3 dimensions (3D) and higher dimensions (nD) extensions to Scagnostics. Especially Scagnostics features of nD data might be more valuable as such data is not directly accessible by visualization, for example, using such features to guide the detection of outliers [Wil18; PNL*19], the selection of clustering [AKMS07], or dimension reduction techniques in deep space [Dan19]. We also provide some hints on their run-time evaluation within the web environment. Also we augment this library with a Scagnostics explorer application, which makes the Scagnostics black box transparent to users.

2. Related Work

This section briefly summarizes the nine Scagnostics measures [WAG05] and current implementations. Five (*outlying*, *skewed*, *sparse*, *clumpy*, and *striated*) out of nine Scagnostics measures can be computed based on the distribution of the edge lengths of minimum spanning tree (MST) built on the underlying scatter points. Besides the MST, the shape measures also depend on the alpha hull (A), and the convex hull (H) of the scattered points. *Convex* score is measured by the ratio of the area of the alpha hull and the area of the convex hull. *Skinny* score measures the ratio of perime-

DOI: 10.2312/egs.20201022

ter to the area of the alpha hull. Also, *Stringy* shape is similar to skinny shape, but it does not have branches. Finally, *Monotonic* score is calculated via the partial correlation of the two variables.

In terms of implementations, there are several Scagnostics implementations in programming languages such as R [Lee18], Python [Jos15] (it is Python binding to R Scagnostics), and Java [DW14a]. Most of these implementations are for 2D Scagnostics, and there is only one attempt to implement Scagnostics in application to 3D data [Fu09]. However, this implementation has some flaws and points to improve that we will explain in our implementation section. To the best of our knowledge, there are no freely available JavaScript implementation for Scagnostics measures.

3. ScagnosticsJS Implementations

3.1. Binning

Due to the constant increase of data and the limited computing power, binning helps to reduce the computation expense. There are two standard ways to bin scatterplots: hexagon vs. leader algorithms. Each algorithm has its pros and cons depending on the data and the analysis task, so our JavaScript implementation provides the flexibility to choose either one of them. While the number of cells in hexagon binning is predefined (e.g., Wilkinson et al. [WAG05] use 40 by 40 hexagon binning) and increases exponentially to the number of dimensions n as 40^n, the number of leaders is always smaller than the number of observations (N). For this reason, we select leader binning for our nD Scagnostics implementation.

3.2. 2D implementation

The 2D ScagnosticsJS implementation includes several intermediate steps described in Figure 2.

1. **Data Normalization** standardizes value ranges of different dimensions into unit range [0,1].
2. **Binning** helps Scagnostics calculation relatively independent of the total number of data points and therefore more scalable.
3. **Triangulation**: MST, concave hull, and convex hull are calculated based on the Delaunay triangulation on the binned data.
4. **MST**: Six out of nine Scagnostics measures (except *convex*, *skinny*, and *monotonic*) are calculated based on the MST.
5. **Degree 1, 2 vertices** are used to calculate the *striated* and *stringy* measures.
6. **Convex hull vs. concave hull**: The *convex* and *skinny* are measured based on the perimeters and/or areas of these shapes.

3.3. 3D implementation

As in 2D, building 3D MST is based on Euclidean distances among data points. The calculations of the convex hull and the concave (alpha) hull in 3D are similar in 2D except the line is conceptually converted into the plane, and α radius for the circle used to calculate the alpha hull becomes the radius for the sphere. The *outlying*, *skewed*, and *clumpy* scores rely on the distribution of the Euclidean lengths of the MST, and therefore their calculations remained the same in 3D. Parallel lines, smooth lines (e.g., spiral) and parallel

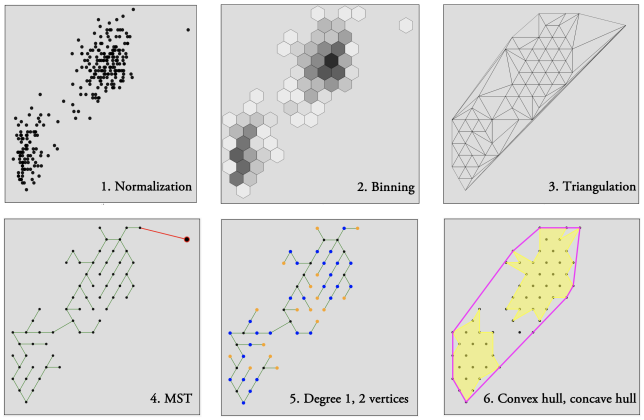

Figure 2: *Intermediate stages in computing 2D Scagnostics: Normalization, Binning, Triangulation, MST (with a red outlier), Degree 1 (orange) vs. Degree 2 vertices, and convex vs. concave hull.*

planes in 3D are considered as *striated* patterns. So, the *striated* score is revised to count on the angle between adjacent planes (p) formed by every three consecutive edges (e_1, e, and e_2) of the MST. Different from [Fu09], we need to consider $|\cos\theta| > 0.75$ instead of $\cos\theta < -0.75$ as the 2D version.

$$c_{striated} = \frac{1}{|V|} \sum_{v \in V^{(\geq 2)}} I(|\cos\theta_{p(e,e_1)p(e,e_2)}| > 0.75) \qquad (1)$$

where $V^{(\geq 2)} \subseteq V$ are vertices of degree$\geq$2. The $\cos\theta_{p(e,e_1)p(e,e_2)}$ is calculated as the dot product of the unit normal vectors of the two planes $p(e,e_1)$ and $p(e,e_2)$. These unit normal vectors are, in turn, computed using the cross products of the vectors made of source and target nodes of e and e_1 for the first plane and e and e_2 for the second plane.

To justify the use of $|\cos\theta| > 0.75$ instead of $\cos\theta < -0.75$, we generated samples for two parallel planes, four parallel lines, and a smooth line to test the striated scores for both cases as shown in Figure 3. While for the cases of two parallel planes and four parallel lines, the results are relatively similar, the case of the smooth line using $\cos\theta < -0.75$ threshold leads to *striated* score of 0 versus 1 when using $|\cos\theta| > 0.75$ threshold. As of the striated definition, its score of zero is not correct for the case of a smooth line.

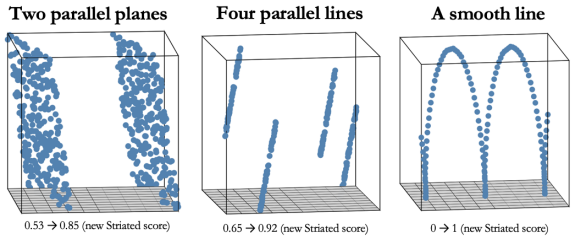

Figure 3: *Comparing striated scores (listed on top of each plot) using* $\cos\theta < -0.75$ *vs.* $|\cos\theta| > 0.75$ *(our revision) thresholds for two parallel planes, four parallel lines, and a smooth line.*

Convex score and *skinny* score in 3D version depend on the surface area and volume (instead of perimeter and area) of the convex

hull (H) and alpha hull (A). They are:

$$c_{convex} = volume(A)/volume(H) \quad (2)$$

$$c_{skinny} = 1 - \sqrt[6]{36\pi}\sqrt[3]{volume(A)}/\sqrt{surfacearea(A)} \quad (3)$$

where the $\sqrt[6]{36\pi}$ is to make sure that $c_{skinny} = 0$, in case of a sphere.

The area of a 3D hull is the sum of areas of all triangles of that hull. These triangles are the faces of the hull as the result of the hull computation. The volume of a 3D hull is computed efficiently using the algorithms from Robert Nürnberg [Nür13]. Algorithm 1 summarizes the steps that we used to calculate the volume of a 3D hull from its faces (this works for both convex and concave hulls).

Algorithm 1 Compute the volume of 3D hull from its set of faces

```
1: procedure COMPUTEVOLUME(faces)
2:     initialization: n = number of faces, volume = 0, i = 0
3:     while i < n do:
4:         triangle_i = faces[i]
5:         vector_i = one vertex of triangle_i
6:         n̂ = normal vector of triangle_i
7:         volume = volume + dot product of vector_i and n̂
8:         i = i + 1
9:     return volume
```

Finally, the 3D *monotonic* score is calculated via the partial correlations of the three variables: $c_{monotonic} = \max[\rho^2_{X,Y|Z}, \rho^2_{X,Z|Y}, \rho^2_{Y,Z|X}]$.

3.4. nD implementation

Many application domains require monitoring of multiple variables for better statistical analysis results, such as multivariate time series abnormality detection [PNL*19]. Therefore, we provide nD version for the Scagnostics scores. Specifically, the scores which are based on the the MST built from the scattered point data (they are *outlying*, *skewed*, *sparse*, *clumpy*, and *stringy* scores). Besides, we also provide the monotonic score, which is calculated based on the maximum partial monotonicity among all pairs of variables. Also, as a natural extension to the concept of distance in 2D, and 3D to nD version, we use the Euclidean distance metric for high dimensional data, and these calculations remained the same for nD implementation except for the correction of the sparse score.

Extending the idea discussed in 3D sparse score implementation, theoretically, the sparse score of nD implementation (using Euclidean distance metric) could get up to $\sqrt{n}$ (n is the number of dimensions), but this is an extreme case because at most there is only one edge length with this value. However, the higher the number of dimensions, the higher the possibility that the MST edges get longer than 1. For instance, Figure 4 shows two synthesized scatter plots (6D and 10D) with sparse scores as 1.76 and 2.16 correspondingly if we use the q_{90} as a sparse measure. Therefore, we propose the sparse score as:

$$c_{sparse} = q_{90}/\sqrt{\lfloor\frac{2\times n}{3}\rfloor} \quad (4)$$

where n is the number of dimensions, the floor operation ($\lfloor\rfloor$) is used because MST always selects the lower distances first, the numerator 2 is due to pairwise Euclidean distance between points, the denominator 3 is the requirement that we would like to have at least 3 MST edges with lengths greater than or equal to the q_{90}. Also, this is compatible with the cases $n = 2$ or $n = 3$ in 2D or 3D implementations. This correction factor reduces the sparse scores for the scatter plots in Figure 4 into 0.88 in both cases.

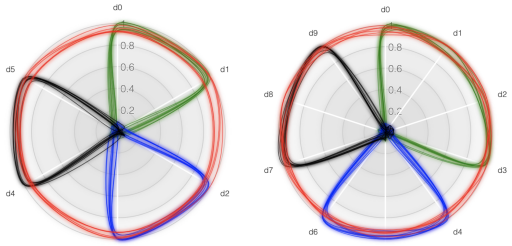

Figure 4: *Synthesized 6D and 10D examples with high sparse scores: 1.76 and 2.17, if we use the q_{90} measure. Radar chart, in this case, is one of many ways to represent multivariate data and line colors represent different classes.*

In this nD version, we do not provide the implementation for the scores based on the convex/alpha hull (*convex* and *skinny*) due to the limited utility [DW14b] and performance constraints. Even with approximation approach such as one from [SV16] the complexity of hull calculation is still $\mathcal{O}(N^2 m^{3/2} log\frac{m}{\varepsilon_0})$, where m is close to the number of vertices of approximation and ε_0 is the maximum error. The time complexity makes it impractical to incorporate these scores in the current nD version.

Regarding the striated score, there is no cross product of two vectors with dimensions higher than three because there are infinitely many unit vectors orthogonal to any given two. Also, lines and planes can be found in higher dimensions, but there is not often much reason to use them [Wor97]. One alternative to this is to explore the subspace of every 2 (or 3) dimensions and use the maximum as the score of nD dataset. We also advise using this approach with cautions [Wil18] since this might lead to issues in high-dimensional data analysis. In several cases, the Manhattan distance metric (L_1 norm) might be preferable than the Euclidean distance metric (L_2 norm), or even the L_k norm where k is a fraction should be explored [AWT09].

4. Evaluations

4.1. Memory and computation time

We computed Scagnostics on six test datasets retrieved from Bureau of Labor Statistics [Bur18] (the first two), World Bank [Wor18] (the next three), and Kaggle [Kag18] (the last one). All executions were performed on a 2.9 GHz Intel Core i5, macOS Sierra Version 10.12.1, 8 GB of RAM computer. Figure 5 summarizes the dataset information (variables, number of plots, and observations per plot), gives exemplar scatterplots, and shows average 2D version Scagnostics computing time for scatterplots in each dataset. Computing MST is the most expensive while calculating scagnostics scores is fast. Binning time depends on the distribution of the underlying data (i.e., a low or high number of bins require re-binning). Also, theoretical analysis and experimented results show

that scagnostics algorithms are efficient and scale well in terms of memory usage (especially with the use of binning).

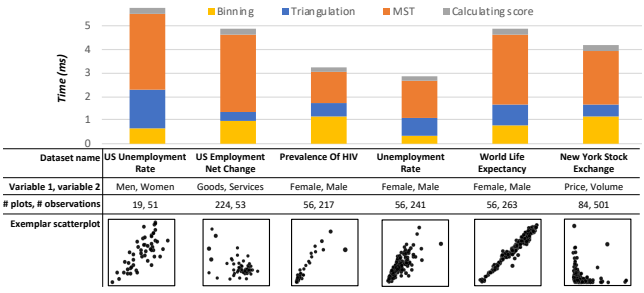

Figure 5: *Time for major computations of 2D Scagnostics.*

4.2. Scagnostics measures and their target patterns

Figure 6 graphically depicts 3D Scagnostics measures and how sensitive they are to the underlying data distribution. The nine measures are projected into 3D space using the first three principal components resulted from Principal Component Analysis (PCA) [WEG87]. It is observable that the corresponding patterns are close to the unit vectors (red lines) of the measures that they target. Interested readers can refer to the supplementary document for the Scagnostics measures on typical patterns that they target.

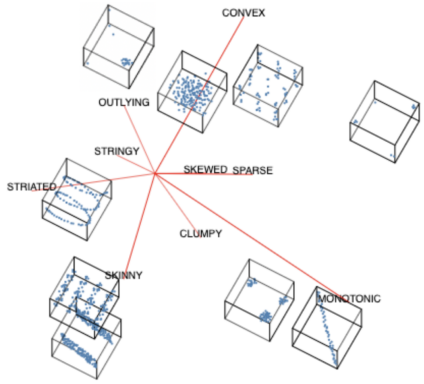

Figure 6: *3D PCA of Scagnostics measures and scatters.*

5. Conclusion and Future Work

In this paper, we provide the JavaScript packages of 2D Scagnostics and further extend it to handle higher dimensional scattered points. We also publish an exploration page, a visual lens into the intermediate stages of Scagnostics calculation. We evaluate ScagnosticsJS on various datasets to provide users a sense of how long it takes to deploy Scagnostics in the web settings. In the future, we will continue to explore more options, such as incorporating different distance metrics for the nD implementation and other approximation methods to convex/alpha hull calculation to complete the remained undefined scores in nD implementation. The source codes and exploration page are available at `https://idatavisualizationlab.github.io/ScagnosticsJS/`.

References

[AKMS07] ASSENT, IRA, KRIEGER, RALPH, MÜLLER, EMMANUEL, and SEIDL, THOMAS. "VISA: visual subspace clustering analysis". *SIGKDD Explor. Newsl.* 9.2 (Dec. 2007), 5–12. ISSN: 1931-0145. DOI: 10.1145/1345448.1345451 1.

[AWT09] ANAND, A., WILKINSON, L., and TUAN, D. N. "An L-infinity Norm Visual Classifier". *2009 Ninth IEEE International Conference on Data Mining*. Dec. 2009, 687–692. DOI: 10.1109/ICDM.2009.119 3.

[BOH11] BOSTOCK, MICHAEL, OGIEVETSKY, VADIM, and HEER, JEFFREY. "D^3 data-driven documents". *IEEE transactions on visualization and computer graphics* 17.12 (2011), 2301–2309 1.

[Bur18] BUREAU OF LABOR STATISTICS. http://www.bls.gov/data/. Sept. 2018 3.

[Dan19] DANG, TOMMY. "FSelector: Variable Selection Using Visual Features". *Proceedings of Graphics Interface 2019*. GI 2019. Kingston, Ontario: Canadian Information Processing Society, 2019. DOI: 10.20380/GI2019.07 1.

[DW14a] DANG, TUAN NHON and WILKINSON, LELAND. "ScagExplorer: Exploring scatterplots by their scagnostics". *IEEE Pacific Visualization Symposium* (2014), 73–80. ISSN: 21658773. DOI: 10.1109/PacificVis.2014.42 1, 2.

[DW14b] DANG, TUAN NHON and WILKINSON, LELAND. "Transforming scagnostics to reveal hidden features". *IEEE Transactions on Visualization and Computer Graphics* 20.12 (2014), 1624–1632. ISSN: 10772626. DOI: 10.1109/TVCG.2014.2346572 3.

[Fu09] FU, LIJIE. "Implementation of Three-dimensional Scagnostics". MA thesis. University of Waterloo, 2009 2.

[Jos15] JOSUA, KRAUSE. *Python binding to R scagnostics*. 2015. URL: https://github.com/nyuvis/scagnostics 2.

[Kag18] KAGGLE. https://www.kaggle.com/datasets. Sept. 2018 3.

[Lee18] LEE, WILKINSON; ANUSHKA, ANAND. *Compute scagnostics - scatterplot diagnostics*. 2018 2.

[Nür13] NÜRNBERG, ROBERT. "Calculating the area and centroid of a polygon in 2d". *URL: http://wwwf. imperial. ac. uk/˜ rn/centroid. pdf* (2013) 3.

[PD19] PHAM, V. and DANG, T. "Outliagnostics: Visualizing Temporal Discrepancy in Outlying Signatures of Data Entries". *2019 IEEE Visualization in Data Science (VDS)*. Vancouver, BC, Canada, Canada: IEEE, Oct. 2019, 29–37. DOI: 10.1109/VDS48975.2019.8973379 1.

[PNL*19] PHAM, VUNG, NGUYEN, NGAN, LI, JIE, et al. "MTSAD: Multivariate Time Series Abnormality Detection and Visualization". *2019 IEEE International Conference on Big Data (Big Data)*. IEEE. 2019, 3267–3276 1, 3.

[SV16] SARTIPIZADEH, HOSSEIN and VINCENT, TYRONE L. "Computing the approximate convex hull in high dimensions". *arXiv preprint arXiv:1603.04422* (2016) 3.

[WAG05] WILKINSON, LELAND, ANAND, ANUSHKA, and GROSSMAN, ROBERT. "Graph-theoretic scagnostics". *Proceedings - IEEE Symposium on Information Visualization, INFO VIS* (2005), 157–164. ISSN: 1522404X. DOI: 10.1109/INFVIS.2005.1532142 1, 2.

[WEG87] WOLD, SVANTE, ESBENSEN, KIM, and GELADI, PAUL. "Principal component analysis". *Chemometrics and intelligent laboratory systems* 2.1-3 (1987), 37–52 4.

[Wil18] WILKINSON, LELAND. "Visualizing Big Data Outliers Through Distributed Aggregation". *IEEE Transactions on Visualization and Computer Graphics* 24.1 (2018), 256–266. ISSN: 10772626. DOI: 10.1109/TVCG.2017.2744685. arXiv: NIHMS150003 1, 3.

[Wor18] WORLD BANK OPEN DATA. https://data.worldbank.org/. Sept. 2018 3.

[Wor97] WORLDWEBMATH. *N Dimensional Geometry*. Last accessed 11 June 2019. 1997. URL: http://web.mit.edu/wwmath/vectorc/ndim.html 3.