NIST Technical Note 2160

VEMOS: A GUI for Evaluation of Similarity Metrics on Complex Data Sets

Eve Fleisig
Günay Doğan

This publication is available free of charge from:
https://doi.org/10.6028/NIST.TN.2160

National Institute of
Standards and Technology
U.S. Department of Commerce

NIST Technical Note 2160

VEMOS: A GUI for Evaluation of Similarity Metrics on Complex Data Sets

Eve Fleisig
Princeton University

Günay Doğan
Information Technology Laboratory
Applied and Computational Mathematics Division

This publication is available free of charge from:
https://doi.org/10.6028/NIST.TN.2160

June 2021

U.S. Department of Commerce
Gina M. Raimondo, Secretary

National Institute of Standards and Technology
James K. Olthoff, Performing the Non-Exclusive Functions and Duties of the Under Secretary of Commerce for Standards and Technology & Director, National Institute of Standards and Technology

National Institute of Standards and Technology Technical Note 2160
Natl. Inst. Stand. Technol. Tech. Note 2160, 34 pages (June 2021)
CODEN: NTNOEF

This publication is available free of charge from:
https://doi.org/10.6028/NIST.TN.2160

Abstract

Similarity and dissimilarity metrics are a fundamental component of many tasks requiring the analysis and comparison of complex, often visual data. Applications ranging from computer vision to forensics require ways to effectively identify images, find clusters or outliers in data sets, or retrieve data items similar to a query item. However, finding an effective metric for a specific task is challenging due to the complexity of modern data sets and the myriad of possible similarity metrics arising from that complexity. We present **VEMOS**, a Python package that provides an accessible graphical user interface (GUI) for the evaluation of such comparison metrics. **VEMOS** provides user-friendly ways to examine individual data items or groups in a data set alongside analyses of metrics' performance on the whole data set, such as clustering, multi-dimensional scaling, and retrieval performance analyses. **VEMOS** aims to help researchers and practitioners evaluate multiple comparison metrics (of similarity or dissimilarity) on rich, diverse data sets.

Key words

GUI; similarity metric; dissimilarity metric; distance metric; comparison metric; exploratory data visualization; data retrieval.

Table of Contents

List of Figures

1. Introduction

Similarity and dissimilarity metrics are a fundamental component of many tasks requiring the analysis and comparison of complex, often visual data. Applications from computer vision to forensics require ways to effectively identify images, find clusters or outliers in data sets, or retrieve data items similar to a query item. All of these tasks require comparing items in a data set using an effective comparison metric (of similarity or dissimilarity) for that application. However, finding an effective metric for a specific task is challenging due to the complexity of modern data sets and the myriad possible similarity metrics arising from that complexity.

The data sets for many contemporary applications often have multiple components, such as different visual representations, along with categorical, geometric, and unstructured information. Examples of such data sets and applications are

- Image classification, such as for object detection or cell imaging, using color images, segmentations enumerating objects or features in the images, or points of interest
- Analysis of material samples, based on orientation images, spectral curves, segmented grains, and topological summaries
- Forensic analysis of crime scene images, such as fingerprints or shoeprints, based on 2D grayscale images, 3D depth images, geometric patterns on the print, and distinguishing characteristics or markings

In any of these examples, there are multiple ways to define a metric to quantify the similarity or dissimilarity of two data records in a given data set. For example, we can

- Correlate the pixel values of color images for computer vision
- Match interest points in a given pair of images
- Segment the images to obtain an enumeration of the entities in the image, and formulate a comparison based on two sets of entities
- Compare feature vectors corresponding to summaries of the data records

Each of these approaches can lead to a similarity/dissimilarity metric for a pair of data records; the effectiveness of each metric will likely depend on the data analysis goal, such as clustering, image retrieval, or outlier detection. Any of the possible metrics can be combined to achieve even more effective hybrid metrics, resulting in many potential comparison metrics for such problems. Thus, a critical problem is determining which comparison metric works better for a given application.

In this paper, we present a general-use software tool, **VEMOS** (Visual Explorer for Metrics of Similarity), to help researchers and practitioners evaluate multiple comparison metrics on rich, diverse data sets for a variety of data analysis tasks. **VEMOS** allows users to load a set of data files and matrices of similarity/dissimilarity scores from one or more metrics. Then, users can:

- Browse the complex data records, whose components can include images, boundary curves, text, or other data types
- Examine relationships to other data records in the data set via predefined groupings or matches

- Examine the similarity/dissimilarity values in each matrix and the associated data records
- Cluster the data records based on the different metrics
- Visualize the data records as 2D or 3D point clouds
- Evaluate match/nonmatch classification performance with analyses such as histograms and ROC (receiver operating characteristic) curves
- Generate new metrics or combine existing metrics to create improved fused metrics
- Select data records or similarity/dissimilarity scores of interest in any of the analyses and return to the associated data records for detailed examination

Existing Python packages for data analysis, such as **SciPy**, **matplotlib**, **scikit-learn**, provide some functionality that can be used for analyses of comparison metrics, but without the ability to easily examine individual items or groups in a data set or to do so alongside analyses of metrics' performance on the whole data set. Even when some functionality is available, it requires significant scripting and knowledge of Python.

Instead, **VEMOS** is built around an easy, user-friendly graphical interface (GUI) that seamlessly integrates both data set exploration and metric evaluation and experimentation. **VEMOS** provides straightforward, intuitive ways to follow the connections and relationships between data records and metric scores, for both broad analyses and in-depth evaluations of subsets of interest. This is of high value to experienced users as well as novices. Our goal with **VEMOS** is to provide a GUI tool that is ready to use, so that experienced researchers and novices alike can load the data and directly begin their analyses. Naturally, no tool can meet all the needs of a user. For this reason, **VEMOS** is provided as an open-source Python package that experienced users can extend for functionality not included in our tool.

VEMOS consists of two interconnected interfaces, the Visual Metric Analyzer and the Data Record Browser. The Data Record Browser enables closer examination of the details of individual data records, including matches to other data records. The Visual Metric Analyzer focuses on a broader exploration and analysis of the data set and the similarity/dissimilarity matrices, relating data visualizations back to individual records. In the following sections, we discuss the visualizations and analyses available in our tool, as well as the data to which this functionality can be applied. In Section 2, we describe the specific kinds of data that are within our scope, the data formats accepted, and how to load files. In Section 3, we describe how the Data Record Browser can be used to view the details of individual data records and their components in a given data set. In Section 4, we discuss the Visual Metric Analyzer and the types of analyses that we can perform on the score matrices.

2. The Data Set

The purpose of VEMOS is to enable users to analyze and evaluate multiple similarity or dissimilarity metrics associated with complex data sets. Thus, the two elements essential to this analysis are the data record files and the matrices of similarity/dissimilarity scores

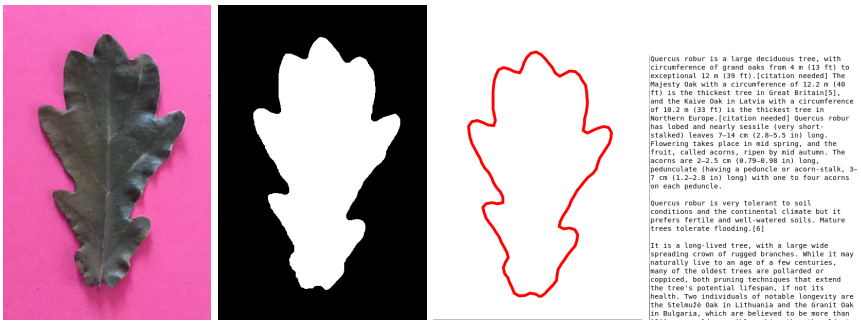


Fig. 1. Sample data record: a leaf image, its segmentation, its boundary curve, and a text description.

between pairs of records. The data records can be heterogeneous, i.e., they can consist of multiple data types, such as images, curves, point clouds, or text descriptions. An example of such a data set is the leaf data set shown in Figure 1 [1]. We computed two dissimilarity score matrices for this data set: one using mean square error between segmentation (binary mask array) pairs, another using the elastic shape distance between pairs of boundary curves [2]. In this section, we describe the data types that data records can have, possible relationships between records, and score matrices. We also describe how the data files should be organized on disk and the acceptable file formats so that **VEMOS** can load and process the data.

2.1 Data types and relationships

A data set for **VEMOS** consists of a collection of data records, each of which has the same heterogeneous structure with various data types. An example is the leaf of a tree species shown in Figure 1. The data record for the leaf includes the leaf image, a binary mask of its segmentation, its boundary curve, and a text description. The data set can include relationships between data records. A given data set of data records can contain groups, classes, clusters, or other divisions. For example, the leaves in the leaf data set can be grouped based on the tree species they come from. More groupings may be generated within the GUI, through user definition or the results of clustering. Data records can also be associated with each other as matches, defined by the user depending on the application. We explain these aspects of the data set and the data records in detail below.

2.1.1 Data records

A data record represents one item of a data set (e.g., one leaf). It consists of:

1. ID: A unique string identifying the record
2. Data types and files: A data record is a list of ”things”. It typically has a heterogeneous structure, consisting of multiple data types, each of which is stored in a

separate file. All records in a data set have the same structure, though all data types do not have to be available for all records. A data record can contain the following data types:

(a) Grayscale or color images

(b) Binary masks/segmentations, stored and viewed as binary images

(c) Point clouds: set of interest points (x,y) from the data record

(d) Curves: the boundary curve of the regions in the image (closed or open)

(e) Text descriptions: text files describing the data record

For example, the leaf data record with ID="Alnus sp_2" has image, segmentation, curve, and text description data types. The corresponding data are stored in the files "Alnus sp_2.jpg", "Alnus sp_2.tiff", "Alnus sp_2.txt", "Alnus sp_2_description.txt" respectively. Appendix B includes the accepted file formats for each data type.

3. Relationships between data records: matches and groups
   Each data record stores a list of records that match this record (e.g., fingerprints from the same person). Matches are always mutual: if `record A` matches `record B`, then `record B` must match `record A`. The matches may come from the description file or user definition in the GUI.

   Each data record also stores a list of groups to which it belongs. For example, leaves from the same species could be grouped together. Groups can contain subgroups, such as subspecies of leaves.

2.1.2 Groupings

Within a data set, records may be grouped according to different characteristics. A grouping is a partition of the data set into groups of records. This can be based on factors, levels, or covariates in the data set. There are usually multiple ways to partition the data set, resulting in multiple possible groupings. Groups can be predefined in a data description file, created in the GUI, or result from clustering performed in the GUI. For example, performing two different kinds of clustering with different parameters would result in two different ways to separate the data records into clusters and therefore create two groupings. These groupings are stored for analysis in **VEMOS**. The grouping in the leaf data set is based on the species of the trees. The groups are "Alnus sp", "Betula pubescens", . . . , etc.

2.1.3 Dissimilarity and similarity matrices

Because different comparison metrics may reveal different degrees of similarity or dissimilarity between data record pairs, each data set may have one or more matrices of pairwise similarity or dissimilarity scores under different metrics. Each matrix entry is a similarity/dissimilarity score between two records in the data set. Different metrics can

then be compared using **VEMOS**. For the leaf data set example, we compute two matrices of dissimilarity scores that we compare. The first is the dissimilarity of the boundary curves of the leaves using the elastic shape distance algorithm described in [2]. The second matrix is the mean squared error (MSE) between two images $I, J \in \mathbb{R}^{m \times n}$, computed by $d(I,J) = \frac{1}{mn} \sum_{k,l=1}^{m,n} (I_{kl} - J_{kl})^2$.

2.2 Data organization, loading, and storage

For **VEMOS** to load and process the data, the data set should be organized on disk or its organization should be provided in a data description file according to the conventions below.

1. File organization
   The user may load data from an organized directory of files or from a data description file. If loading from a file directory, the data may be organized using named folders or naming conventions:

   (a) Named folders: Files for each data type should be in folders with the name of the datatype. For the leaf data set, this can correspond to four top level folders: Images, Segmentations, Curves, Descriptions. Under the top folders, there can be separate folders for individual tree species files (see Figure 13).

   (b) Naming conventions: Each data type must have a naming convention that is different from the naming conventions for other data types, as indicated in the startup screen. For the leaf data set, the top level folders can be the species names: "Alnus sp", "Betula pubescens", ... Under each top folder, the data files with names conforming to the conventions are stored. For example, under folder "Alnus sp", we have the files "Alnus sp_0.jpg", "Alnus sp_0.tiff", "Alnus sp_0.txt", "Alnus sp_0_description.txt", "Alnus sp_1.jpg", ... (see Figure 14). In this example, "*.jpg" are the image files, "*.tiff" are the segmentation files, "*.txt" are the curve files, "*_description.txt" are the text description files.

   VEMOS will import the files following the directory structure and the data types indicated in the startup screen. See Appendix B for details and examples of directory and file naming formats.

2. Data description file
   If no organized format for the data files storage is available, the user may load the data using a data description file. The data description file should be a text file in which each row describes one data record in the format
   `ID; (group1, group2); (match1, match2,...); data_type_1: file1; data_type_2: file2;...`

3. Load a previous session: After loading data once, all analyses, score matrices, and data record files from a previous session can be reloaded in one step from the startup screen (see below).

2.2.1 The data loading widget

At startup (Figure 2), the user may choose how to load data for the two interfaces of **VEMOS**.

1. `Load Previous Session`: Load all information (directory, score matrices, data record files, analyses) from a previous session.

2. `Choose File Directory`: Select the file directory containing the data files.

3. `Interface to Open`: Select whether to open the Visual Metric Analyzer (for score matrix analyses) or the Data Record Browser (for viewing individual records).

4. `Load Data Files`: Select whether to use a data description file or file directory to load record files. If using a description file, use `Choose Description File` to load the file. If opening the Visual Metric Analyzer, data record files are not required; if they are unavailable, select `No Data Records Available`.

5. `Set Data Types`: Select the file naming formats and file extensions for a given data type. `Read Data Types from Folders` sets **VEMOS** to scan the data folders instead in order to infer this information automatically.

6. `Load Similarity/Dissimilarity Matrices`: Load score matrices to use for the analyses in the interface. If opening the Data Record Browser, score matrices are not required; if they are unavailable, select `No Scores Available`.

3. Data Record Browser

The Data Record Browser enables users to examine individual data records more closely. While browsing the data files and viewing the records' matches and groups, they also gain an overview of the record's similarity to other records under different metrics. They can also assign matches and groups for later use in the Visual Metric Analyzer.

3.1 Browsing individual data records

The Data Record Browser displays the components (e.g. images, curves, text) of a data record in a series of **matplotlib** figures (see Figure 3). The user may browse through the records using the buttons directly underneath the figure. The `Jump Forward` and `Jump Backward` buttons allow the user to move forward or backward by a specified increment. The `Go To` button allows the user to view the data record with the desired ID.

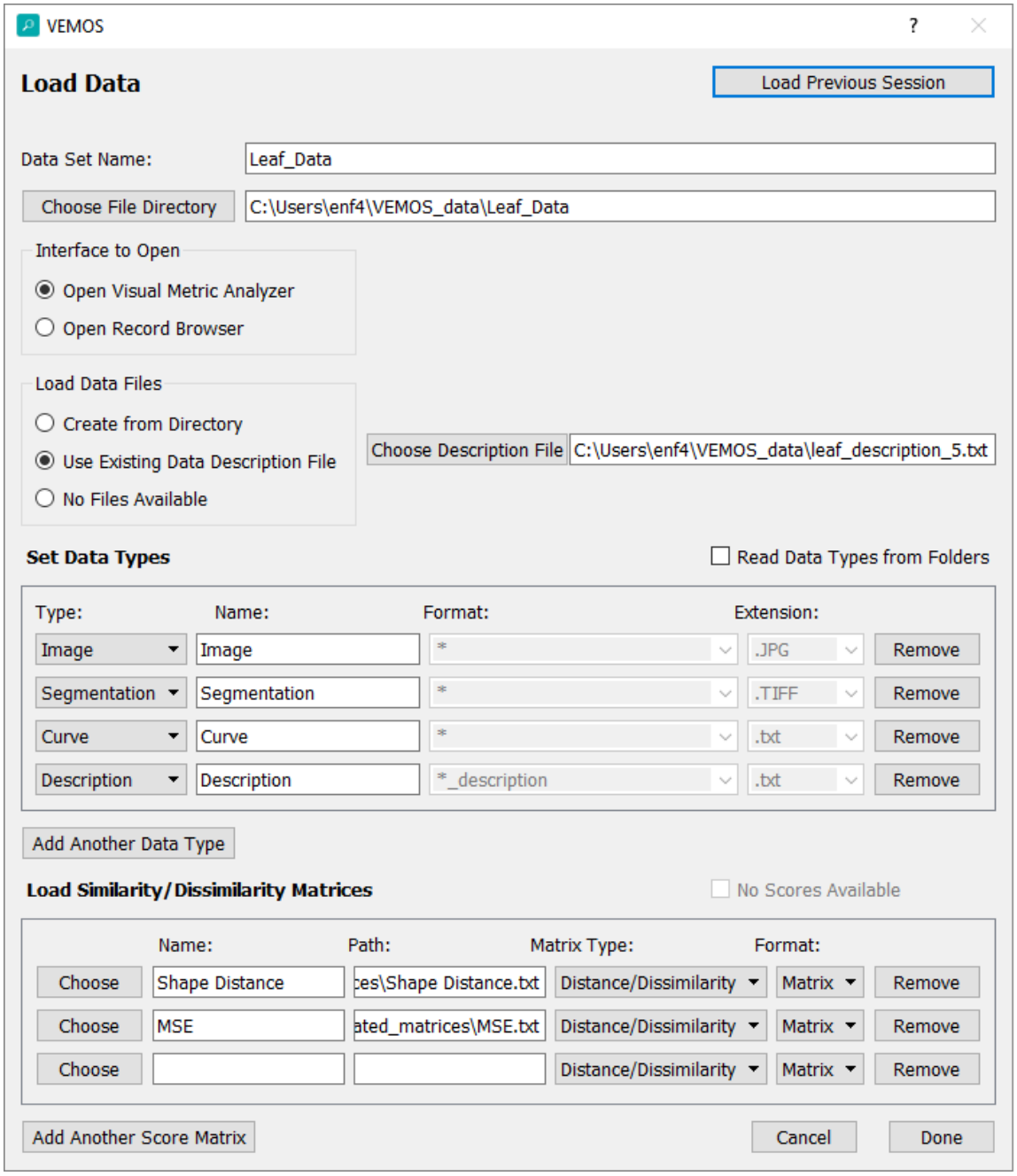


Fig. 2. Startup screen for loading data into VEMOS.

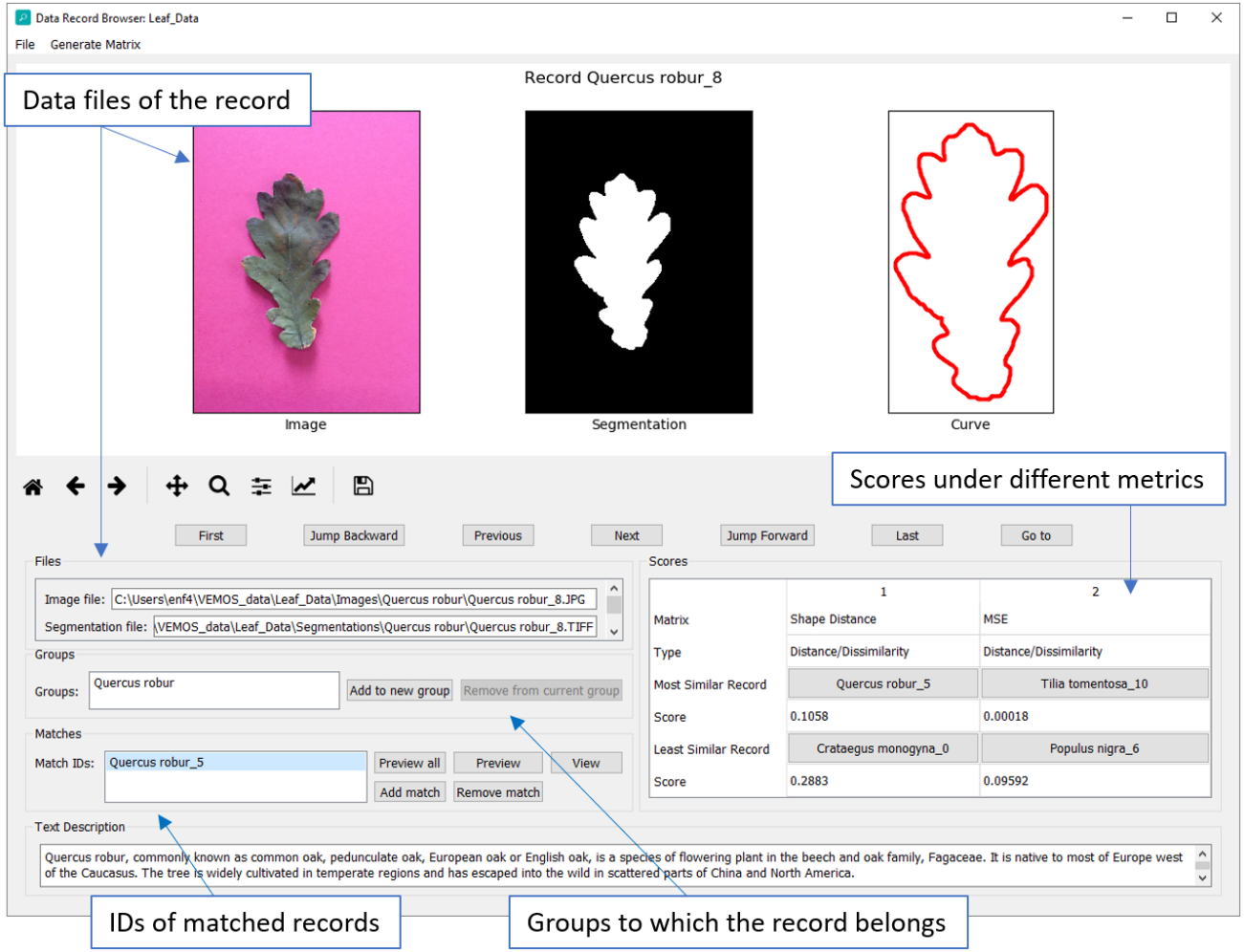


Fig. 3. A data record displayed on the Data Record Browser.

3.2 Files, groups, and matches

The files, groups, and matches for the displayed data record are shown underneath the figures. The user may select `Add to new group` or `Remove from current group` to change the groups to which a record belongs.

The IDs of records that match the current record are displayed underneath the groups. The user may do the following:
`Preview all`: Open a window displaying all matches of the current record for each data type of the record.
`Add match`: Add another record to the current record's list of matches.

After selecting a match from the list, the user may do the following:
`Preview`: Open the match in a new window.
`View`: Display all information about the match on the Record Browser.
`Remove match`: Removes the match. If `record B` is removed from `record A`'s list of matches, then `record A` is also removed from `record B`'s list of matches.

3.3 Information from score matrices

The data record on display is linked to its score entries in the score matrices. For each matrix loaded, **VEMOS** displays the matrix's name and type (similarity or dissimilarity). It also lists the most and least similar records to the current record under that metric and their similarity/dissimilarity scores. The user may click on the ID of the most or least similar record to view that record.

3.4 The File menu

The Record Browser facilitates setting data record relationships (matches and groups) by allowing the user to save a data description file (`Save data description`/`Save data description as`). Then, this information can be quickly loaded for use in the Visual Metric Analyzer. The user may also:
`Open data description file`: View the data description file.
`Update loaded data`: Reopen the loading screen to change what data records or matrices are included.
`Open Visual Metric Analyzer`: Open the GUI for analysis of similarity/dissimilarity score matrices.

4. Visual Metric Analyzer

The Visual Metric Analyzer allows the user to analyze a data set as a whole, relating the analyses of the metrics back to the individual data records and visualizations. It facilitates comprehensive analyses, such as the distribution of match and nonmatch scores, ROC curves of the data, clustering, and multidimensional scaling, as well as the examination of specific outliers or other data points of interest. This allows users to better understand how

their metrics perform on whole data sets or specific subsets of data records. These analyses can also be saved and reloaded later. In this section, we describe the metric analyses available in the GUI.

4.1 Visualizing the similarity/dissimilarity matrices

The central table and heat map allow for quick visualization of the similarity/dissimilarity score matrices and the distribution of high and low scores.

4.1.1 The central table

The central table (Figure 4 displays the current score matrix, which stores the similarity/dissimilarity score for records i and j in cell (i, j). The user may highlight table entries with the mouse and/or keyboard, then right-click to view the corresponding record pairs.

4.1.2 Heat map

By selecting `Analyses` → `Heat Map`, each score matrix is displayed as an image in which the similarity/dissimilarity score of each pair of data records is a colored (x, y) point, with red being least similar and blue most similar (Figure 5). It provides a visual summary of the areas of similarity or dissimilarity in the data.

4.1.3 Settings

The following settings let the user adjust how data is displayed:

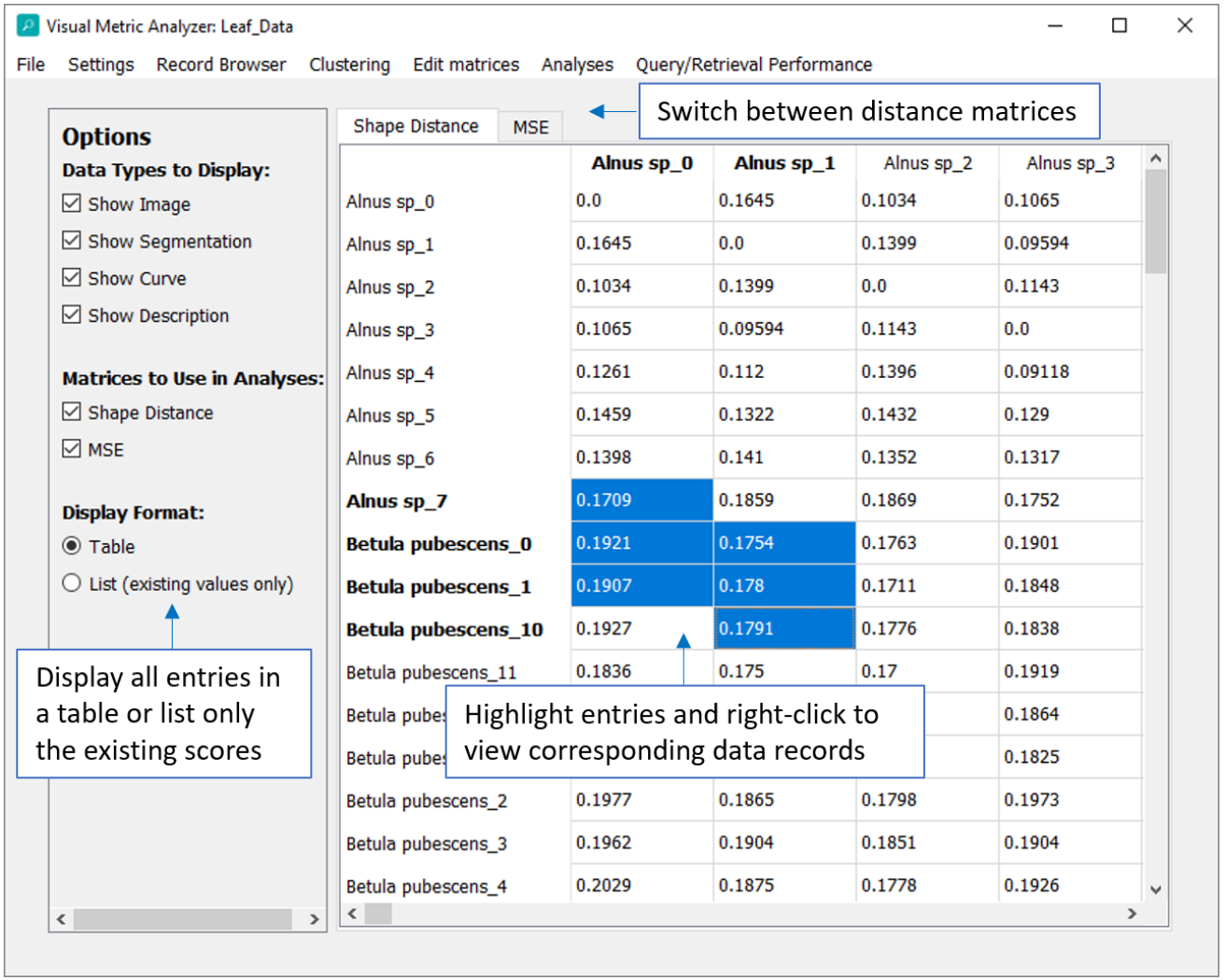
`Data Types to Display`: Sets which data types to see when examining a record, such as images, segmentation masks, or boundary curves.

`Matrices to Use in Analyses`: Sets which score matrices to include when performing analyses on the data.

`Display Format`: For incomplete matrices, it may be more convenient to view the scores as a list in the form $(i, j, score)$ rather than a table, where i and j denote the IDs of the data records being compared, and *score* is the matrix entry for the pair (i, j).

4.2 Spatial visualization of data records as 2D/3D points

Since comparison metrics provide a measure of the similarity of data records, it is often informative to view the records as points in space with the distance between them determined by their similarity. This allows the user to identify clusters and outliers visually and note patterns in the spatial layout of the data. However, complex data sets often do not reside in a low-dimensional vector space. Instead, a similarity or dissimilarity matrix can be used to represent the data records as a set of 2D or 3D points that have approximately the same distances between them as the original data records using multi-dimensional scaling (MDS) [3]. This allows the user to examine the spatial layout of the data records as if they were 2D or 3D points (Figure 6).



	Alnus sp_0	Alnus sp_1	Alnus sp_2	Alnus sp_3
Alnus sp_0	0.0	0.1645	0.1034	0.1065
Alnus sp_1	0.1645	0.0	0.1399	0.09594
Alnus sp_2	0.1034	0.1399	0.0	0.1143
Alnus sp_3	0.1065	0.09594	0.1143	0.0
Alnus sp_4	0.1261	0.112	0.1396	0.09118
Alnus sp_5	0.1459	0.1322	0.1432	0.129
Alnus sp_6	0.1398	0.141	0.1352	0.1317
Alnus sp_7	0.1709	0.1859	0.1869	0.1752
Betula pubescens_0	0.1921	0.1754	0.1763	0.1901
Betula pubescens_1	0.1907	0.178	0.1711	0.1848
Betula pubescens_10	0.1927	0.1791	0.1776	0.1838
Betula pubescens_11	0.1836	0.175	0.17	0.1919
Betula pubes				0.1864
Betula pubes				0.1825
Betula pubescens_2	0.1977	0.1865	0.1798	0.1973
Betula pubescens_3	0.1962	0.1904	0.1851	0.1904
Betula pubescens_4	0.2029	0.1875	0.1778	0.1926

Fig. 4. Two dissimilarity matrices displayed on the central table of the Visual Metric Analyzer.

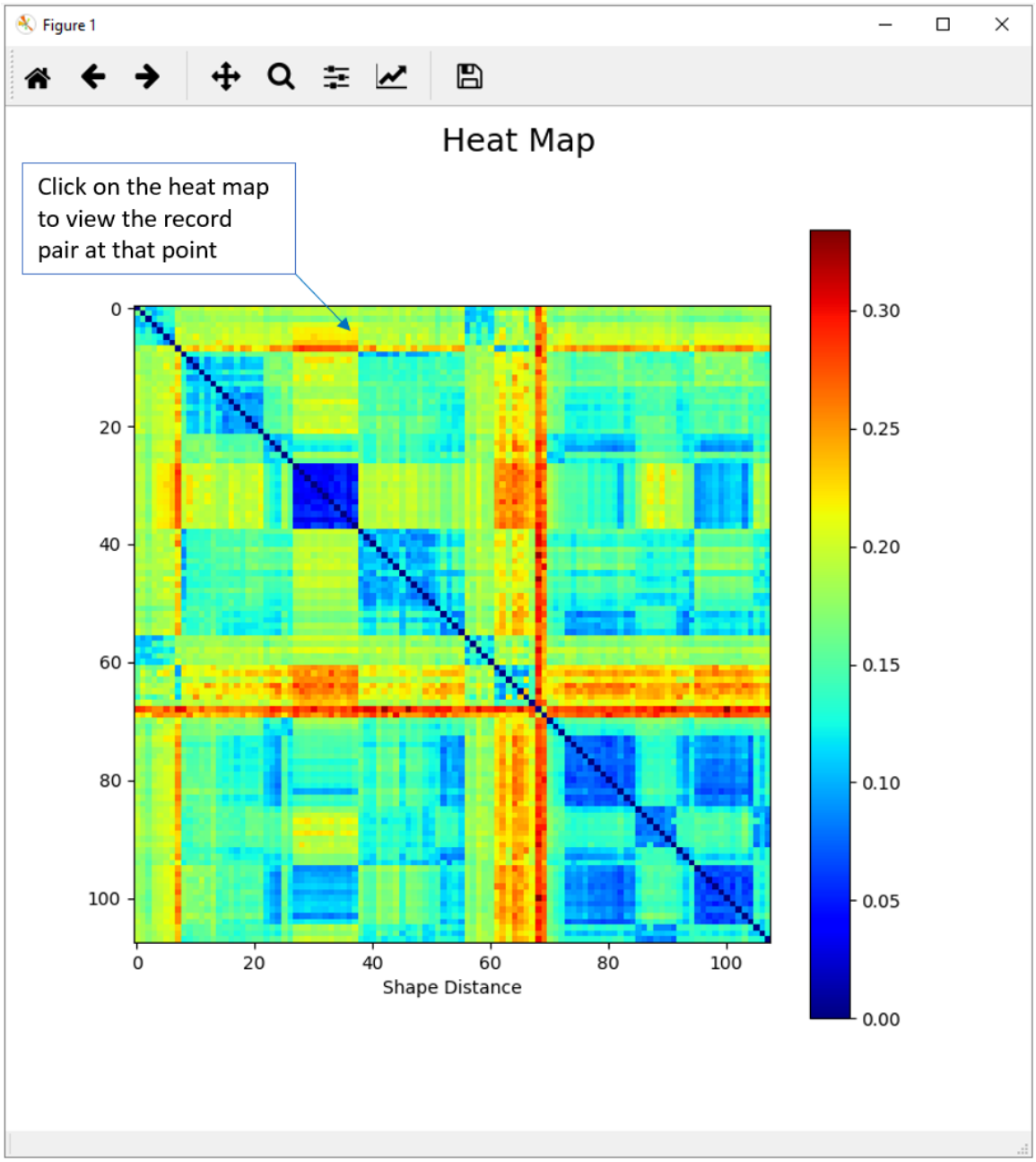


Fig. 5. The heat map of a dissimilarity matrix.

We use the **scikit-learn** implementation of MDS to place the data records as points in 2D or 3D space. The points may then be colored according to different groupings, allowing the user to see how the spatial layout correlates with preexisting groups or the results of clustering. Interactive options include:

- Hovering over a point on the plot displays its ID.
- Double-clicking on a point or selecting an area on the plot displays the full information of the corresponding data records.
- `Lasso/Rectangle`: On 2D MDS, toggles between selecting a rectangular or freeform area (Figure 9).
- `Color by Grouping`: Colors the points in the MDS depending on their groupings under clustering analyses or preexisting groupings in the data.

These options facilitate the identification of outliers or other points of interest.

4.3 Clustering

Just as the data set can come with one or more user-defined groupings (e.g., leaves grouped by tree species), a grouping can also be generated by clustering. These data-driven algorithms group the data records based on their distances from each other. For many applications, it is critical to understand whether a given comparison metric can lead to a clear grouping or clustering of the data records. For this reason, we provide two clustering algorithms to compute groups or clusters using the score matrices, hierarchical clustering and spectral clustering. After selecting hierarchical or spectral clustering, the user may adjust the number of clusters, clustering parameters, and display options. Clusters are automatically stored as groupings in the data set and can later be used to color other visualizations by grouping. **VEMOS** uses the **scikit-learn** implementation of spectral and hierarchical clustering.

4.3.1 Hierarchical clustering dendrogram

The dendrogram (Figure 7) displays merged clusters of data records as branches of the diagram [4]. Hovering over a blue leaf node displays that record's ID; hovering over a red internal node displays all records in that cluster. Clicking on a point displays all associated records in a separate window.

4.3.2 Spectral clustering plot

After performing spectral clustering [5], the plot displays each cluster of points in a different color on a two-dimensional MDS plot (Figure 8). As in the plain MDS plot, hovering over a point on the plot displays its ID and double-clicking displays its data records in a separate window. The user may also click and hold the mouse to select an area and view all records in that range (Figure 9). The records in each cluster are also listed underneath the plot.

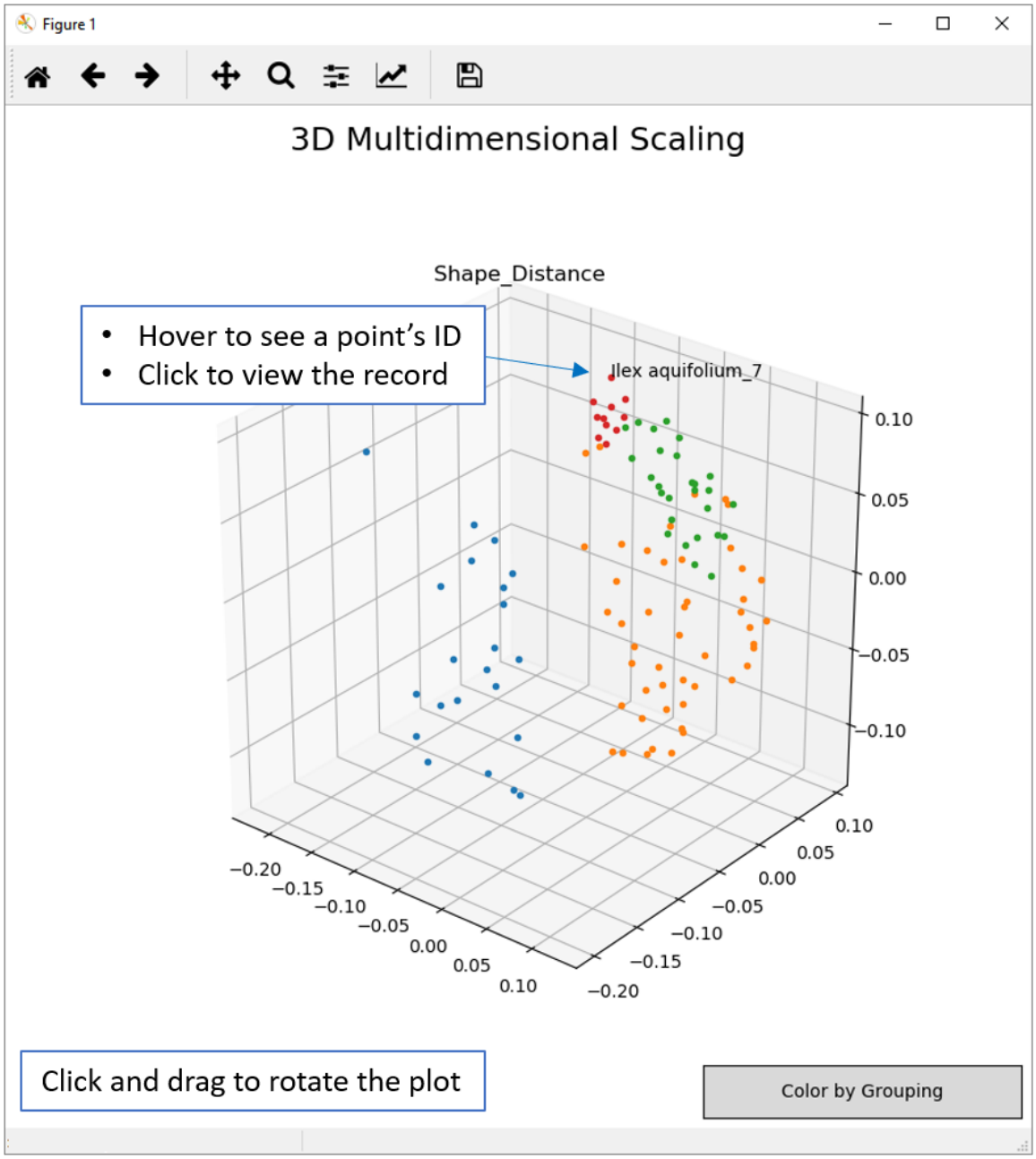


Fig. 6. Visualization of a data set as a 3D point cloud, using MDS on a dissimilarity matrix. Each point corresponds to a data record; here, points are colored by the results of spectral clustering.

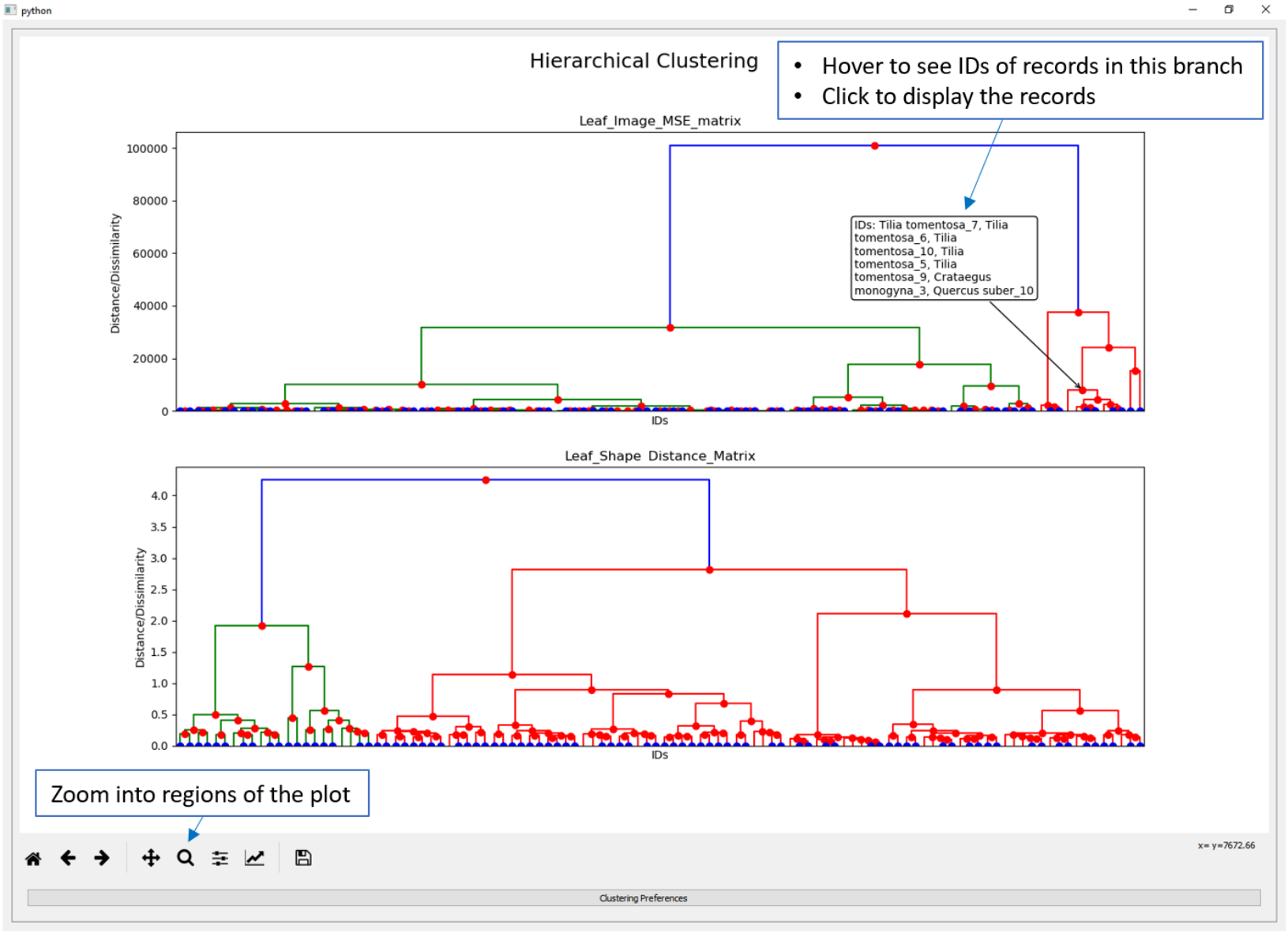


Fig. 7. Hierarchical clustering of the data records based on two distance matrices.

`Lasso Select/Rectangle Select`: Toggles between selecting a rectangular or freeform area on the MDS plot.
`View`: Displays the data records in one cluster in a separate window.
`View Records in All Clusters`: Displays the data records in each cluster in a separate window.

4.4 Binary classification performance

A common question when testing similarity metrics is whether the metric is effective in determining if two given items match–for example, whether the metric can determine that two leaves belong to the same plant, two fingerprints were made by the same finger, or two cell images belong to the same cell. Similarly, in database applications, queries can be executed to retrieve matches of a given record. Thus, the metric must be able to accurately identify the record that most closely matches the query record.

These questions amount to binary classification problems, as the metric must be able to classify data records into matches or nonmatches. When matches are indicated in the data set, the scores in a matrix are grouped according to whether they correspond to pairs of matching records. The user can then test a metric's binary classification performance using **VEMOS**. This helps the user to determine whether the similarity/dissimilarity scores of data record pairs under different metrics can effectively distinguish matches from non-matches.

4.4.1 Receiver operating characteristic curve

The receiver operating characteristic (ROC) curve (Figure 10) plots the true positive rate against the false positive rate when different score thresholds are chosen for distinguishing matches from non-matches [6]. The curve's proximity to the upper left of the plot measures the accuracy of the algorithm; the area under the curve (AUC) is displayed in the bottom right of the plot (see Figure 10). Hovering over the plot displays the approximate true positive rate (TPR) and false positive rate (FPR) at that point. Hovering over a particular score displays information about the quantity and percent of match and nonmatch scores to either side of that score in the table below the figure. Clicking on a score pins the true and false positive rates of the score to the figure. Highlighting a range on the plot displays all scores in that range.

`Show on Single Plot/Show on Separate Plots`: Displays the ROC curves for multiple matrices (if present) on the same plot or multiple plots.

4.4.2 Histogram

The histogram displays the distribution of similarity/dissimilarity scores for matched and unmatched data records. Greater separation between the distributions of matched and unmatched scores indicates that the metric is better able to discriminate between matches and nonmatches. Hovering over the figure displays the score at that point and the quantity and

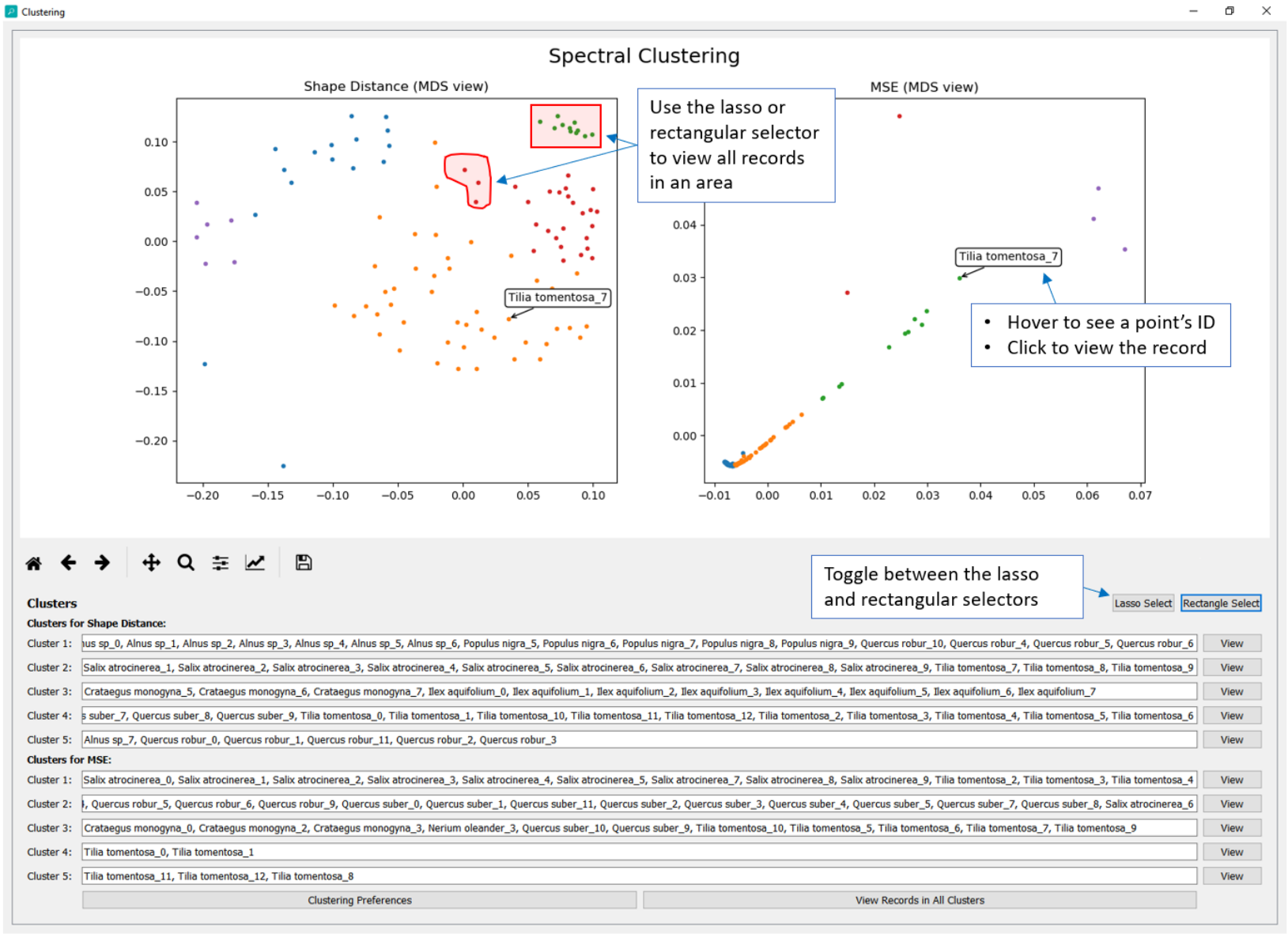


Fig. 8. Spectral clustering of the data set. The point cloud visualizations (colored by cluster) are based on 2D multidimensional scaling of each score matrix.

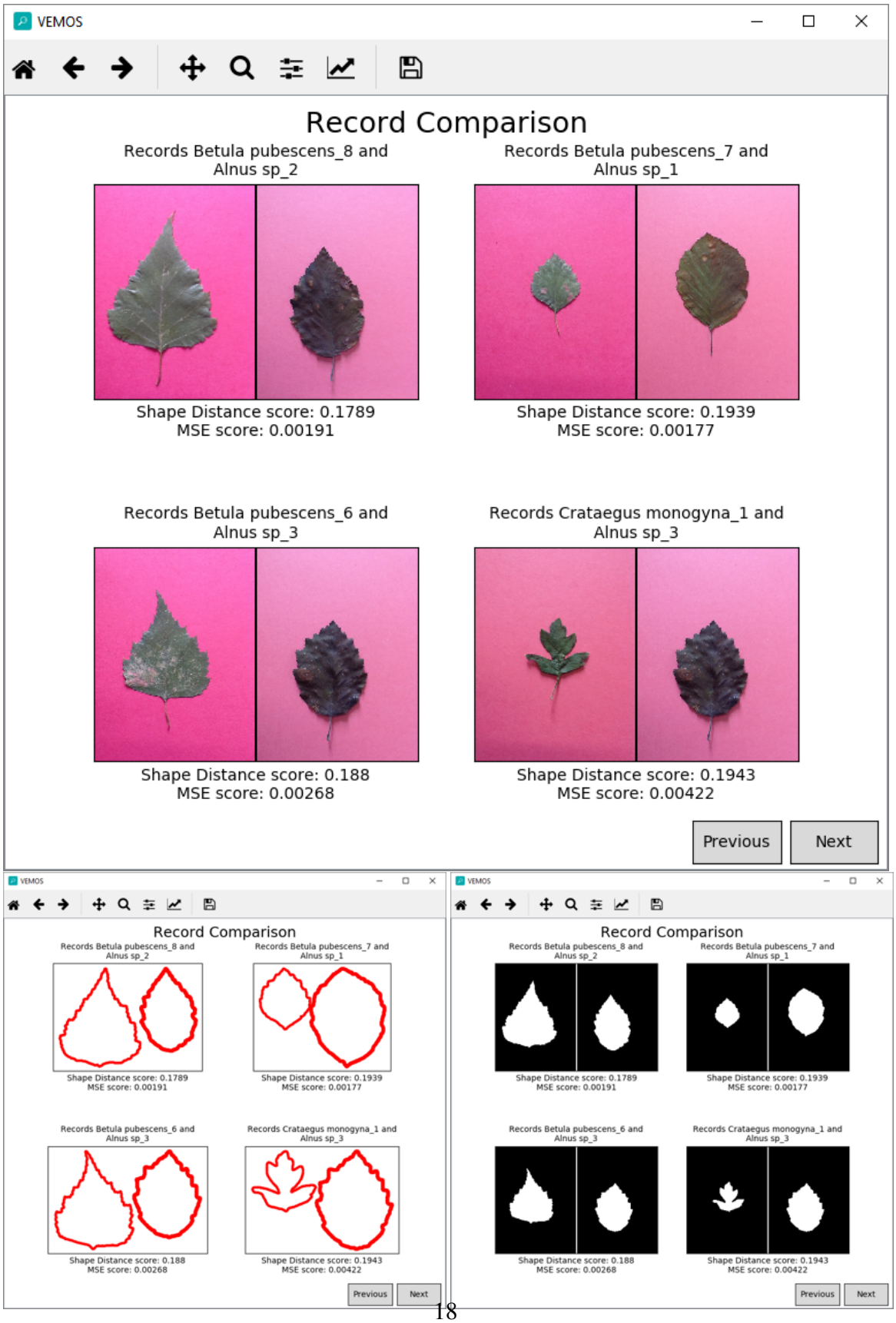
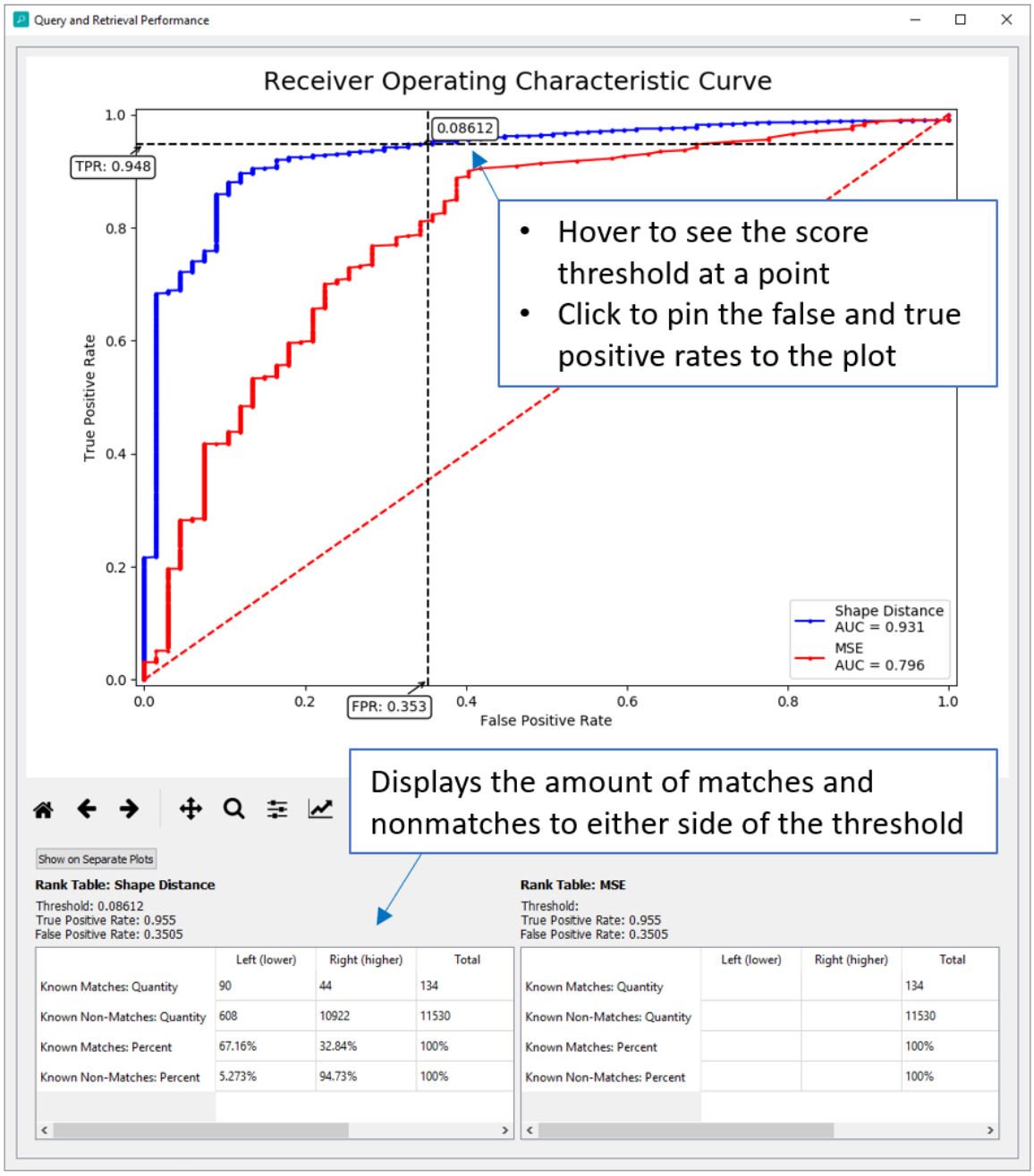


Fig. 9. Record pairs chosen from a lasso selection on the plot.



	Left (lower)	Right (higher)	Total
Known Matches: Quantity	90	44	134
Known Non-Matches: Quantity	608	10922	11530
Known Matches: Percent	67.16%	32.84%	100%
Known Non-Matches: Percent	5.273%	94.73%	100%

	Left (lower)	Right (higher)	Total
Known Matches: Quantity			134
Known Non-Matches: Quantity			11530
Known Matches: Percent			100%
Known Non-Matches: Percent			100%

Fig. 10. Receiver operating characteristic (ROC) curve of two dissimilarity matrices.

percent of match and nonmatch scores to either side of that score in the table below the figure. Clicking at that point pins the score to the figure.

4.4.3 Smoothed histogram

The smoothed histogram (Figure 11) displays the distribution of similarity/dissimilarity scores for matched and unmatched records, smoothed using Gaussian kernel density estimation [7]. Hovering over the plot displays the score at that point and the heights of the match and nonmatch distributions at that point, and lists the quantity and percent of match and nonmatch scores to either side of that score in the table below the figure. Clicking on that point pins the score to the figure.

4.4.4 Linear ordering

The linear ordering (Figure 12) plots the similarity/dissimilarity scores of all matched and unmatched scores on two lines, to see how scores are distributed at a glance. Compared to a histogram or smoothed histogram, the linear ordering makes it easier to see the distribution of smaller sets of scores and to spot outliers in the data. Hovering over the figure displays the score at that point and the quantity and percent of match and nonmatch scores to either side of that score; clicking at that point pins the score to the figure.

4.4.5 Statistics

Displays the standard deviation and mean and median score of each matrix.

4.5 Matrix generation and matrix fusion

In typical usage scenarios, users import score matrices that they created into **VEMOS** for analysis. Users may also be interested in examining the performance of other existing comparison metrics on the same data set or combining metrics to create a new, improved metric. Hence, **VEMOS** includes additional functionality for matrix generation and matrix fusion.

4.5.1 Matrix generation

Although some data sets already come with matrices of similarity or dissimilarity scores provided for different metrics, some data sets may include data files, such as images, with no preexisting scores to compare them. In other cases, the data set may already contain comparison scores under some metrics, but the user may benefit from comparing a custom similarity metric to a standard one.

For these use cases, **VEMOS** provides the ability to create score matrices using standard image comparison techniques. Images or segmentations can be compared using any of the metrics provided below. These metrics represent each pair of images I_1, I_2 as matrices

Fig. 11. The smoothed histogram of the dissimilarity scores under a metric. The blue and red histograms correspond to match and nonmatch scores, respectively.

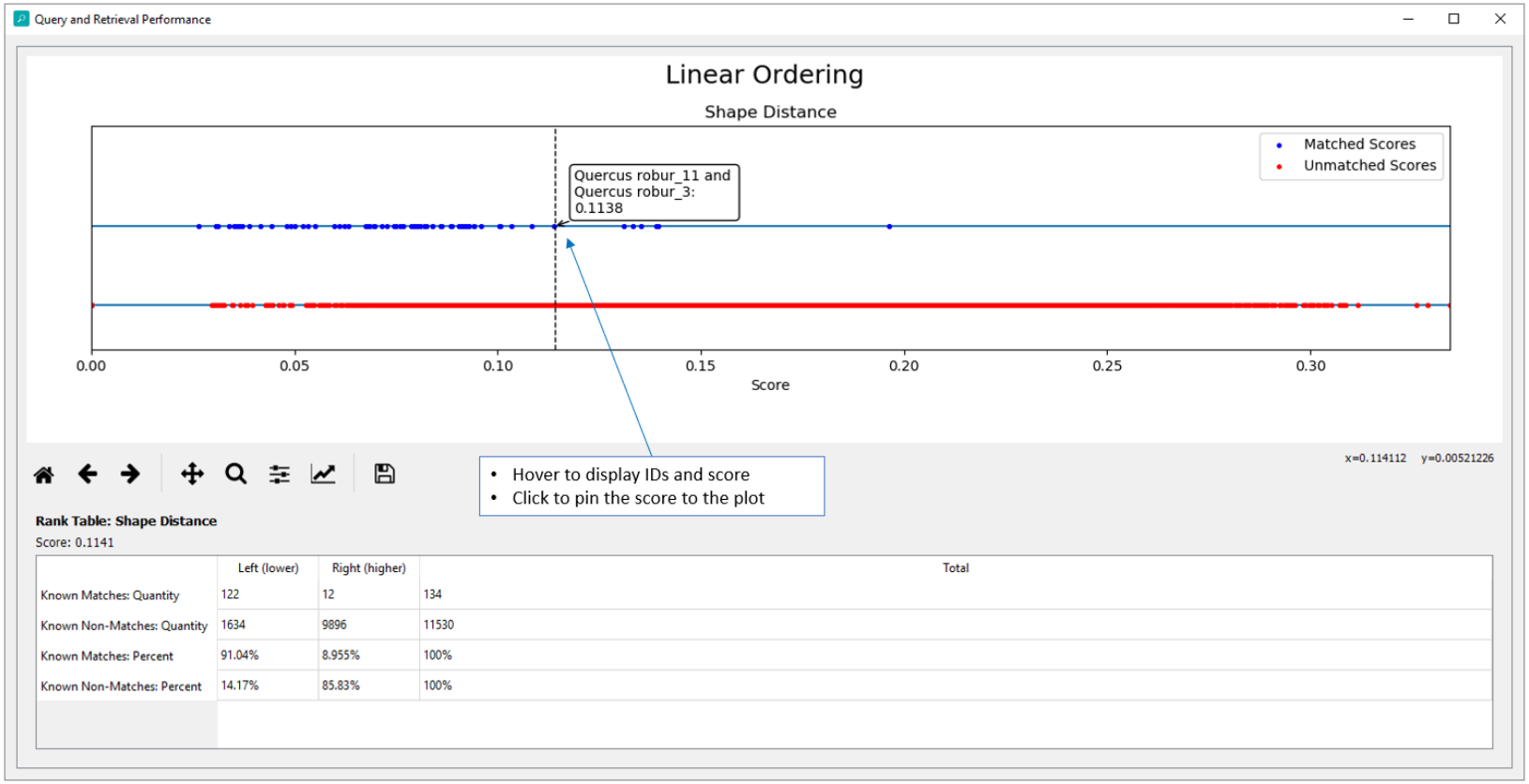


Fig. 12. The linear ordering of the data set.

in which each (i, j) value is the RGB color of that pixel. For each metric, the two images are compared as follows:

- Structural similarity (SSIM): Using the scikit-image implementation, calculates the SSIM score as developed in [8].
- Mean squared error (MSE)
- Normalized root mean squared error (NRMSE)
- Euclidean distance: Calculates the Euclidean or Frobenius norm of the difference between corresponding pixels in the two images.
- Hamming distance: Calculates the Hamming norm $\sum_{i,j} \mathbb{1}_{I_{ij} \neq J_{ij}}$ from the difference of corresponding pixels in the two images I, J.

The SSIM, MSE, and NRMSE metrics use the scikit-image implementation [9]. The Euclidean distance is computed with the `norm` function for arrays in NumPy [10].

Matrix generation is available both under the `Edit Matrices` menu of the Visual Metric Analyzer and in the menu of the Data Record Browser.

4.5.2 Combining metrics: matrix fusion

Individual metrics may take into account different aspects of the data records. For example, one metric may calculate the similarity of leaves based on the shapes of their boundary curves, while another may examine the colors of the images. Thus, combining these metrics to create a fused metric that accounts for both of these features may improve classification performance.

VEMOS provides the functionality to combine metrics to improve separation between matched and unmatched scores. Each pair of records is a data point classified as matching or non-matching, and its scores under each selected metric form its feature vector. The data points can then be represented in high-dimensional space. Support vector classification (SVC) then searches for a hyperplane in that space that provides the best boundary between matching and non-matching data points, acting as a classifier. The user may select whether to use a linear, polynomial, or radial basis function for the SVC kernel, which determines the shape of the classifier [11, 12].

The decision function then gives a data point's distance from the hyperplane (positive if classified as a match, and negative if not). Adding this distance to this matrix of distances results in the non-negative fused matrix. By taking information from multiple metrics into account, this new metric can improve binary classification performance.

Matrix fusion is available under the `Edit Matrices` menu of the Visual Metric Analyzer.

5. Conclusion

Developing effective comparison metrics to quantify the similarity or dissimilarity of data records is central to many data analysis tasks. Real-world data sets can be complex, consisting of data records with multiple components, such as images, text, or geometric elements

like curves. Such data sets can be analyzed with multiple possible comparison metrics depending on the application, so it is crucial to evaluate and compare the performance of each of these comparison metrics in conjunction with the associated data sets.

This paper introduced **VEMOS**, a Python GUI to facilitate evaluation of comparison metrics. **VEMOS** has two interfaces: the Data Record Browser, which allows examiners to organize and examine individual data records, and the Visual Metric Analyzer, which allows users to evaluate metrics using analyses such as clustering, multidimensional scaling, ROC curves, and histogram comparisons. Simultaneously, it allows users to view individual data records–outliers, clusters, or other points of interest–alongside the analyses.

VEMOS currently works primarily to evaluate user-defined metrics, computed externally, then loaded into **VEMOS**. Future versions of **VEMOS** will include more similarity/dissimilarity metric function in the GUI itself, so that more types of score matrices can be directly computed from the GUI. Future capabilities will also be extended to more complex analyses and data types, such as network graphs, geometric meshes, various feature vectors and their visualizations.

Acknowledgments

We are thankful for valuable suggestions and support from the members of the NIST Footwear Forensics Group: Martin Herman, Hari Iyer, Yooyoung Lee, Steve Lund, Adam Pintar, Gautham Venkatasubramanian. We also gratefully acknowledge the support from NIST Award 70NANB16H306, NIJ Award DJO-NIJ-17-RO-0202, Theiss Research, NIST Information Technology Laboratory, NIST Special Programs Office, and NIST Summer Undergraduate Research Program (SURF).

References

[1] Silva PF, Marcal AR, da Silva RMA (2013) Evaluation of features for leaf discrimination. *Springer Lecture Notes in Computer Science* 7950:197–204.

[2] Doğan G, Bernal J, Hagwood CR (2015) A fast algorithm for elastic shape distances between closed planar curves. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, , pp 4222–4230.

[3] Kruskal JB (1964) Multidimensional scaling by optimizing goodness of fit to a nonmetric hypothesis. *Psychometrika* 29(1):1–27. https://doi.org/10.1007/bf02289565

[4] Ward Jr JH (1963) Hierarchical grouping to optimize an objective function. *Journal of the American Statistical Association* 58(301):236–244. https://doi.org/10.2307/2282967

[5] Luxburg UV (2007) A tutorial on spectral clustering. https://doi.org/10.1007/s11222-007-9033-z.

[6] Fawcett T (2006) An introduction to roc analysis. *Pattern Recognition Letters* 27(8):861–874. https://doi.org/10.1016/j.patrec.2005.10.010

[7] Scott DW (2015) *Multivariate Density Estimation: Theory, Practice, and Visualization* (John Wiley & Sons), . https://doi.org/10.1002/9781118575574
[8] Wang Z, Bovik A, Dheikh H, Simoncelli E (2004) Image quality assessment: From error visibility to structural similarity. *IEEE Transactions in Image Processing* 13(4):600–612.
[9] van der Walt S, et al. (2014) **Scikit-image**: Image processing in Python. *PeerJ* 2(453).
[10] Oliphant TE (2006) *A guide to NumPy*. Vol. 1 (Trelgol Publishing USA), .
[11] Chang CC, Lin CJ (2011) Libsvm: A library for support vector machines. *ACM transactions on intelligent systems and technology (TIST)* 2(3):27.
[12] Pedregosa F, et al. (2011) **Scikit-learn**: Machine learning in Python. *Journal of Machine Learning Research* 12:2825–2830.
[13] Jones E, Oliphant T, Peterson P, et al. (2001–) **SciPy**: Open source scientific tools for Python. Available at http://www.scipy.org/.
[14] Hunter JD (2007) **Matplotlib**: A 2d graphics environment. *Computing In Science & Engineering* 9(3):90–95. https://doi.org/10.1109/mcse.2007.55
[15] **PyQt** (2012) **PyQt** reference guide Available at http://www.riverbankcomputing.com/static/Docs/PyQt4/html/index.html.
[16] **Anaconda** (2020) Anaconda Software Distribution. Available at https://www.anaconda.com/distribution.

Appendix A: System Requirements

VEMOS requires the following software:

- Python 3
- **NumPy** 1.16 or later [10]
- **SciPy** 1.2.1 or later [13]
- **matplotlib** 1.4.3 or later [14]
- **scikit-learn** 0.20 or later [12]
- **scikit-image** 0.15 or later [9]
- **PyQt5** [15]

Anaconda distribution of Python 3 packages can be downloaded and installed to satisfy these requirements [16]

Appendix B: Accepted File Formats

Accepted data file formats:

- Images and binary masks/segmentations: All formats that can be processed with matplotlib.image.imread, which includes .png, .jpg, .jpeg, .tif, and .tiff
- Curves and point clouds: Text file with a list of points in the format
  `x1, y1`

```
x2, y2
...
xn, yn
```

- Text description: Text file.

Accepted similarity/dissimilarity matrix formats:

- Recommended for complete matrices: Text or comma-separated value file of a 2D list of scores, where each row and column corresponds to a data record.
- Recommended for incomplete matrices: comma-separated value file in tabular form, as below:

		Metric 1	Metric 2	Match
ID_1	ID_2	.123	.234	Y
ID_1	ID_4	.456	.567	N
...				

  such that in the first row (of length n), columns 3 to $n-1$ are the names of the metrics; in each subsequent row, columns 1 and 2 are the IDs of the records being compared and columns 3 to $n-1$ are scores under the different metrics. Column n stores whether or not the two records are matches, `Y` if so and `N` if not).

Asymmetric matrices may be symmetrized using the average, minimum, or maximum of A_{ij} and A_{ji}.

Appendix C: Directory Organization

VEMOS is designed to load data record files from a wide range of directory organizations, with these requirements:

- List the names of the data types in the `Set Data Types` box.
- Either (a) files for each data type should be in folders with the name of the data type, or (b) each data type must have a naming convention that is different from the naming conventions for other data types. If (a), check the box `Read Data Types from Folders`. If (b), provide the data type naming conventions in the `Set Data Types` box.
- Data record files should not be in the exact same folder as matrices, and subdirectories of data record files should not be in the same folder as data record files themselves.

Figure 13 provides examples of directories loaded using folders with the names of the data types. Figure 14 provides an example of a directory of record files loaded using file naming conventions.

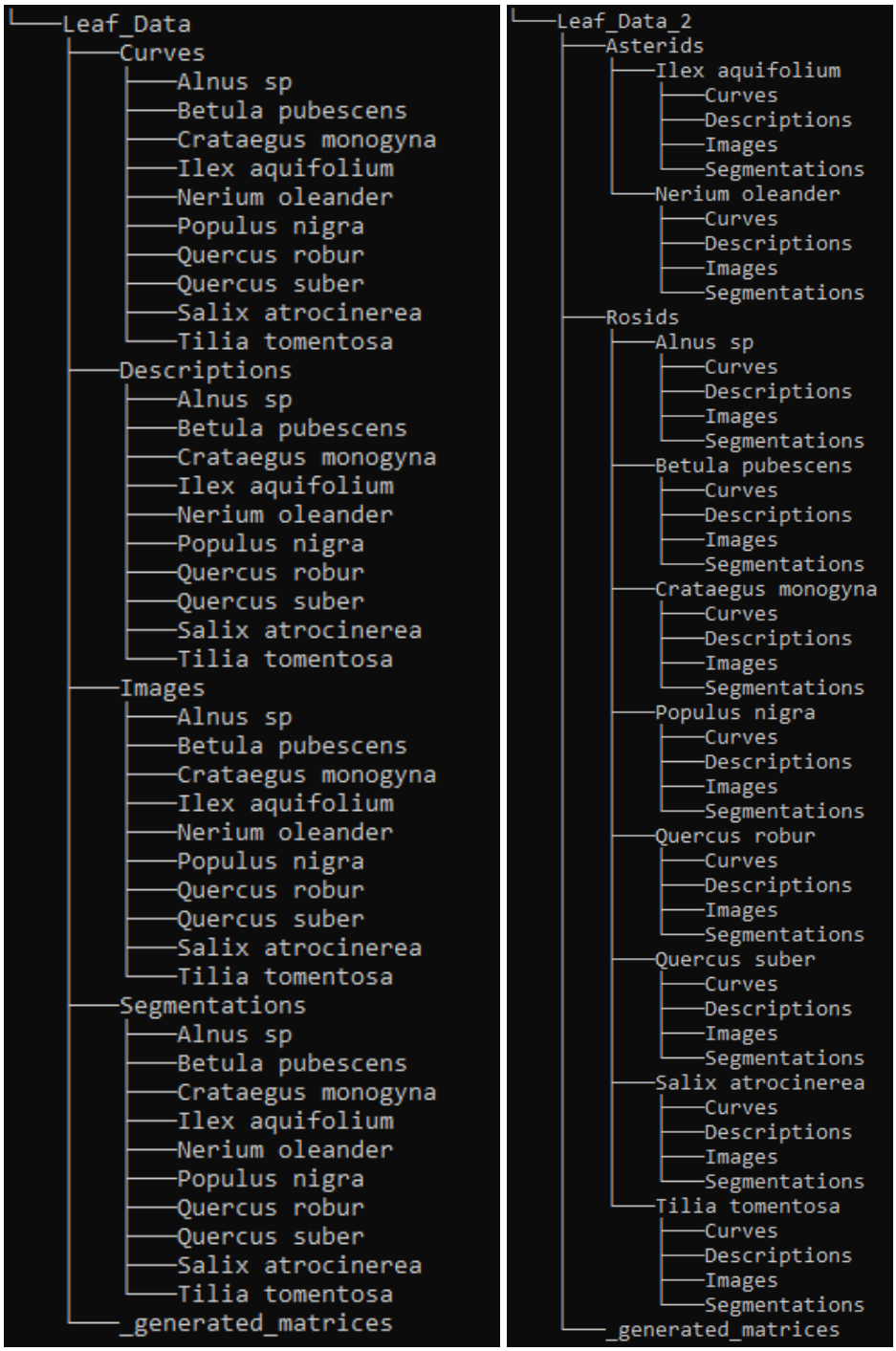


Fig. 13. Two possible directory structures for loading record files. At left: The folders for each data type contain folders with the names of the groups. At right: There are two levels of groups; at the bottom level, folders with the names of each data type contain the files themselves.

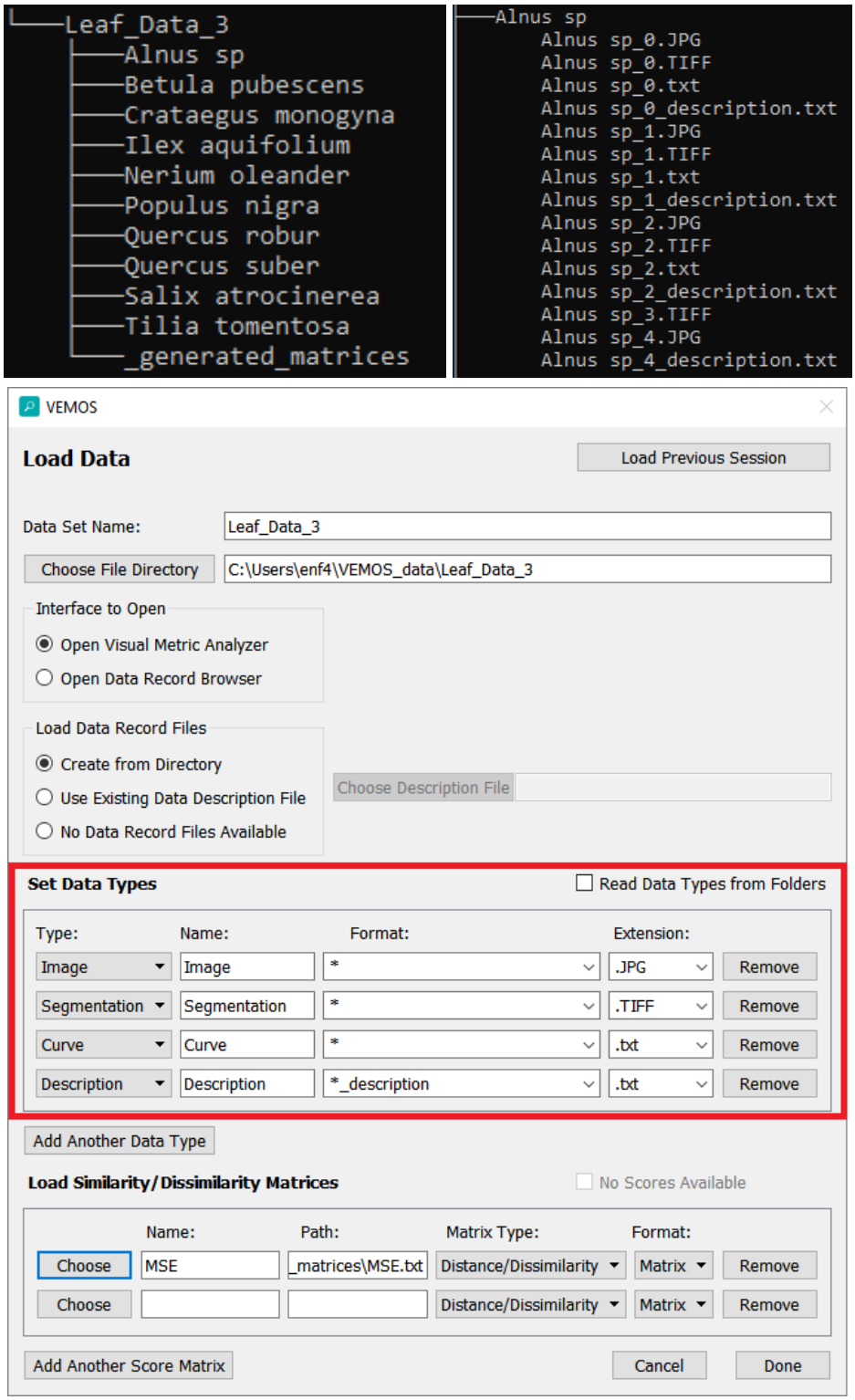


Fig. 14. A possible structure for loading record files using naming conventions. The directories contain only the names of the groups. Inside each group, such as Alnus sp, files for all data types (image, segmentation, curve, and description) are in the same folder. To determine which file corresponds to each data type, **VEMOS** uses the naming conventions under `Set Data Types`. Note that files for every data type do not have to be available for every record, as is the case for Alnus sp_3 and Alnus sp_4.