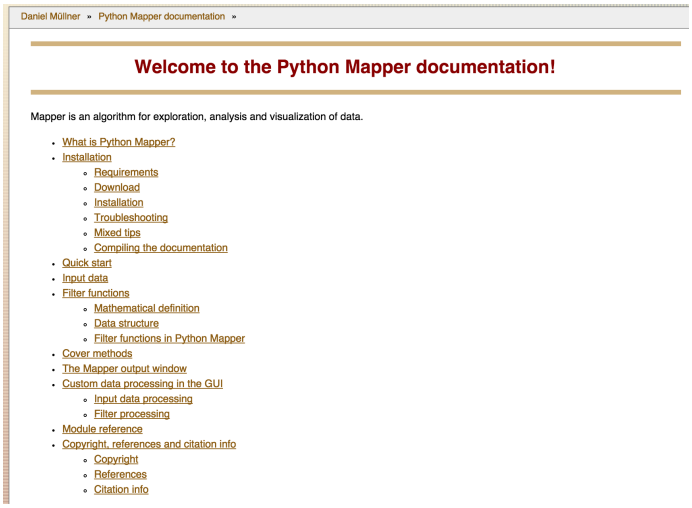
Introduction to MAPPER

Leyda Almodóvar

You will find Mapper and instructions to download it and install it here:
`http://danifold.net/mapper`



The following packages are already installed in the computers in B5. You only need to install them if you wish to work on your PC or on other computers.

Requirements

1. Python 2.6 or higher. The GUI needs Python 2 since it depends on wxPython and PyOpenGL; Python Mapper itself can be run under Python 2 and Python 3.

2. NumPy and SciPy

3. Matplotlib

4. Graphviz

5. Optionally cmappertools. Python Mapper will run without this module, but with limited functionality. Note that cmappertools need the Boost C++ libraries.

For the GUI:

1. wxPython
2. PyOpenGL

Download: The source distribution of Python Mapper can be downloaded here: `http://danifold.net/mapper/mapper.tar.gz`

Make sure to look at `http://danifold.net/mapper/installation/index.html` for installing instructions and troubleshooting tips.

In order to run Mapper you must make certain choices.

1. **Metric**
   (a) *Euclidean*: The *Euclidean* metric between two points $\mathbf{x} = (x_1, x_2, , x_n)$ and $\mathbf{y} = (y_1, y_2, , y_n)$ is given by $d(\mathbf{x}, \mathbf{y}) = |\mathbf{x} - \mathbf{y}| = \sqrt{\sum_{i=1}^{n} |x_i - y_i|^2}$.
   (b) *Minkowski*: The *Minskowski* metric of order p between two points $\mathbf{x} = (x_1, x_2, , x_n)$ and $\mathbf{y} = (y_1, y_2, , y_n)$ is defined as $d(\mathbf{x}, \mathbf{y}) = \sum_{i=1}^{n} (|x_i - y_1|^p)^{1/p}$.
   (c) *Chebychev*: The *Chebychev* metric between two points $\mathbf{x} = (x_1, x_2, \ldots, x_n)$ and $\mathbf{y} = (y_1, y_2, \ldots, y_n)$ is defined as $d(\mathbf{x}, \mathbf{y}) = \max_i |x_i - y_i|$.
2. **Filter function**

   A filter function is a function on the data set, $f : X \to \mathbb{R}^k$. The Mapper algorithm supports general, vector-valued functions, while the GUI is restricted to real-valued functions (the case $k = 1$) for simplicity.

   (a) Eccentricity
   (b) kNN distance
   (c) Distance to a measure
   (d) Density, Gaussian Kernel
   (e) Graph Laplacian
   (f) Distance matrix eigenvector
3. **Type of cover**
   (a) Uniform 1-d cover
   (b) Balanced 1-d cover
   (c) Subrange decomposition

4. **Clustering algorithm**

   (a) *Single*: The distance between two clusters is defined as that of the closest pair of individuals, where it only considers pairs consisting of elements from different clusters: $d(r,s) = min(dist(x_{ri}, x_{sj})), i \in (i, \ldots, n_r), j \in (1, \ldots, n_s)$.

   (b) *Complete*: The distance between two clusters is defined as that of the most distant pair of individuals, where it only considers pairs consisting of individuals from different clusters: $d(r,s) = max(dist(x_{ri}, x_{sj})), i \in (i, \ldots, n_r), j \in (1, \ldots, n_s)$.

   (c) *Average*: The distance between two clusters is defined as the average of the distance between all pairs of individuals that are made up of one individual from each cluster: $d(r,s) = \frac{1}{n_2 n_s} \sum_{i=1}^{n_r} \sum_{j=1}^{n_s} dist(x_{ri}, x_{sj})$.

   (d) *Weighted*: The distance between two clusters is defined as the weighted average of the distance between all pairs of individuals that are made up of one individual from each cluster. It uses a recursive definition for the distance between two clusters: $d(r,s) = \frac{(d(p,s)+d(q,s))}{2}$.

   (e) *Median*: The distance between two clusters is defined as $d(r,s) = \|\tilde{x}_j - \tilde{x}_s\|_2$ where $\tilde{x}_r$ and $\tilde{x}_s$ are weighted centroids for the clusters r and s and $\tilde{x}_r$ is defined recursively as $\tilde{x}_r = \frac{1}{2}(\tilde{x}_p + \tilde{x}_q)$.

   (f) *Centroid*: The distance used is the Squared Euclidean distance between centroids $d(r,s) = \|\tilde{x}_j - \tilde{x}_s\|_2$ where $\tilde{x}_r = \frac{1}{n_r} \sum_{i=1}^{n_r} x_{ri}$.

   (g) *Ward*: The distance used is $d(r,s) = \sqrt{\frac{2n_r n_s}{(n_r+n_s)}} \|\tilde{x}_r - \tilde{x}_s\|_2$. The distance is defined as the incremental sum of squares, that is, the increase in the total within-cluster sum of squares as a result of joining two clusters. The within-cluster sum of squares is defined as the sum of the squares of the distances between all objects in the cluster and the centroid of the cluster.

To find more information about the covers provided by Mapper:
`http://danifold.net/mapper/cover.html`

To find more information about the filters provided by Mapper:
`http://danifold.net/mapper/filters.html`

It is important that your data contains only numbers. Remove dollar signs, commas, NA, NaN and headers.

In order to open Mapper you could double click the folder named 'mapper' and then double click the folder named 'bin' and then double click the file named 'MapperGUI.py'. Alternatively, you could open up a terminal (found on the bottom of the screen):

Figure 1: Open up a terminal

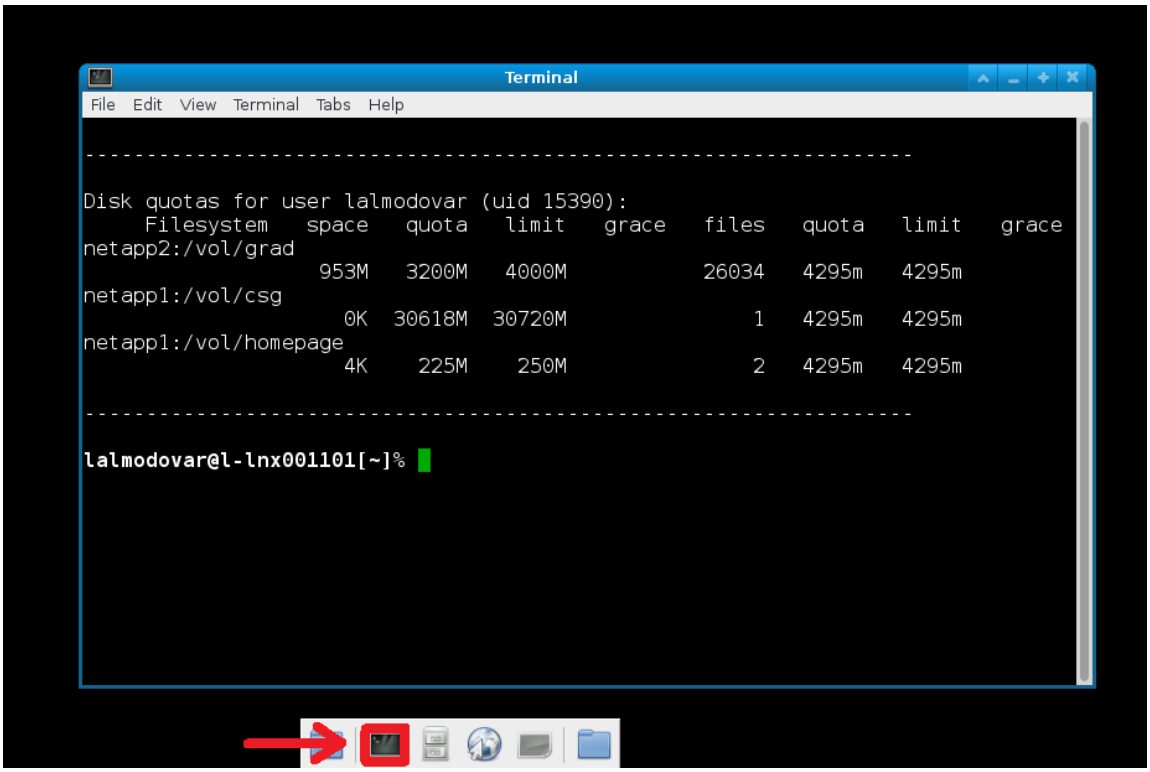


Figure 2: In order to open the GUI from the terminal:

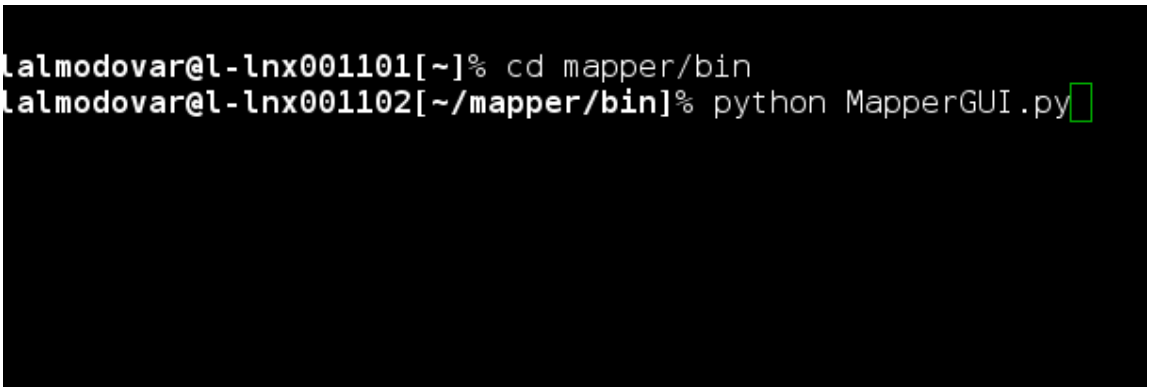


Figure 3: Once you press enter the GUI should appear and it looks like this. The next step is to load the data.

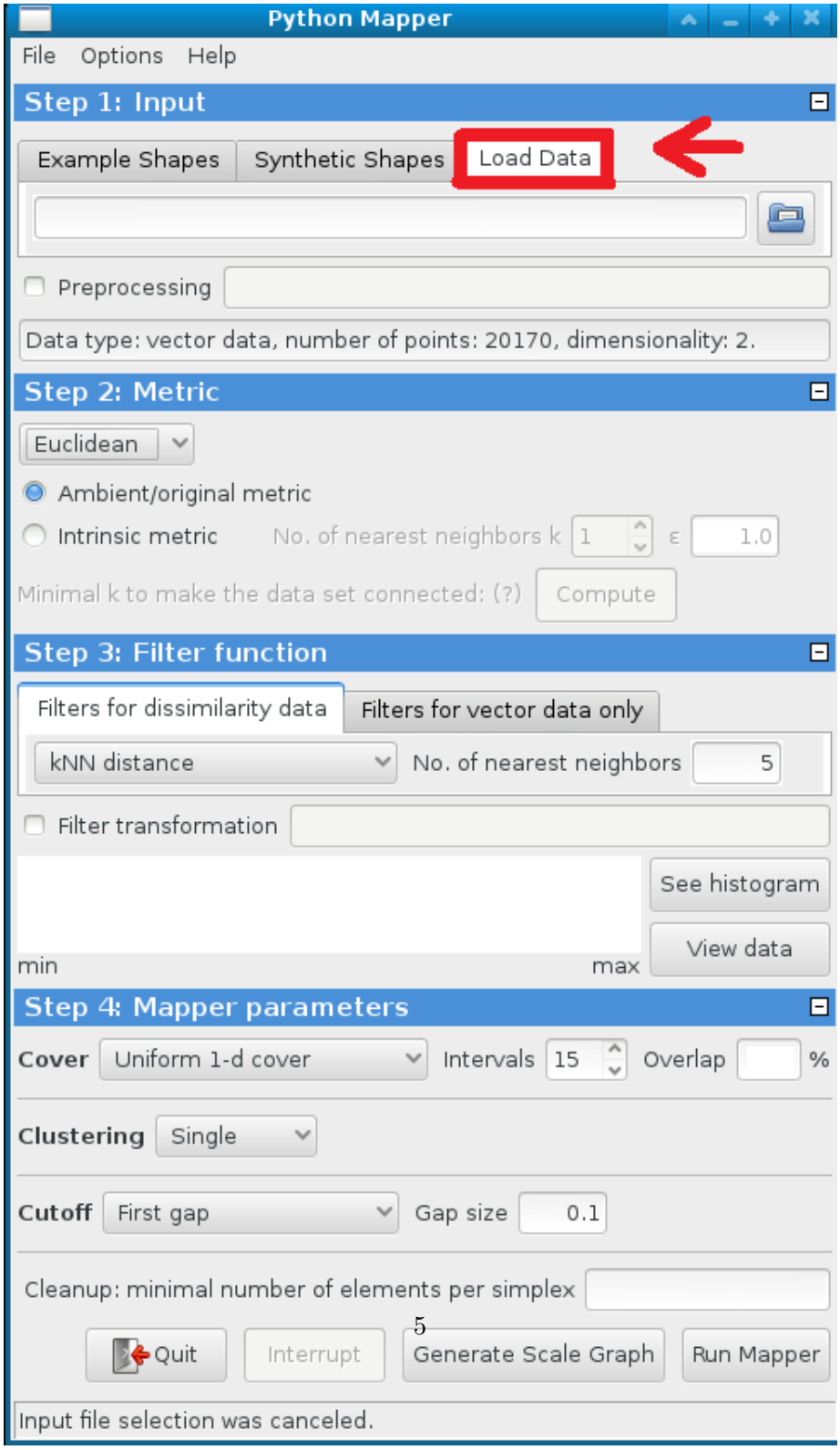


Figure 4: Browse for data file

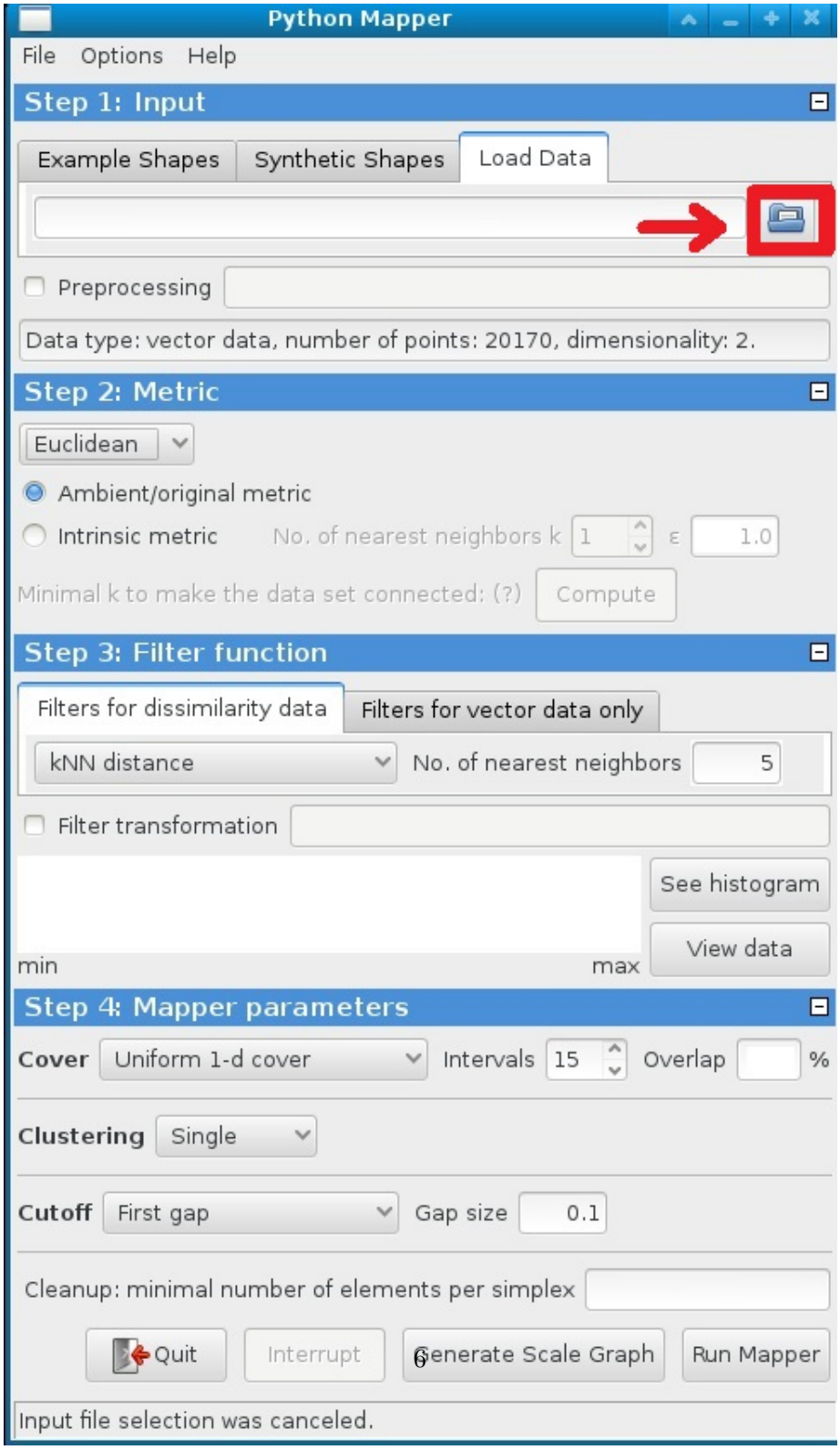


Figure 5: Mapper tells you the number of data points and dimensionality of your data set once you select the file

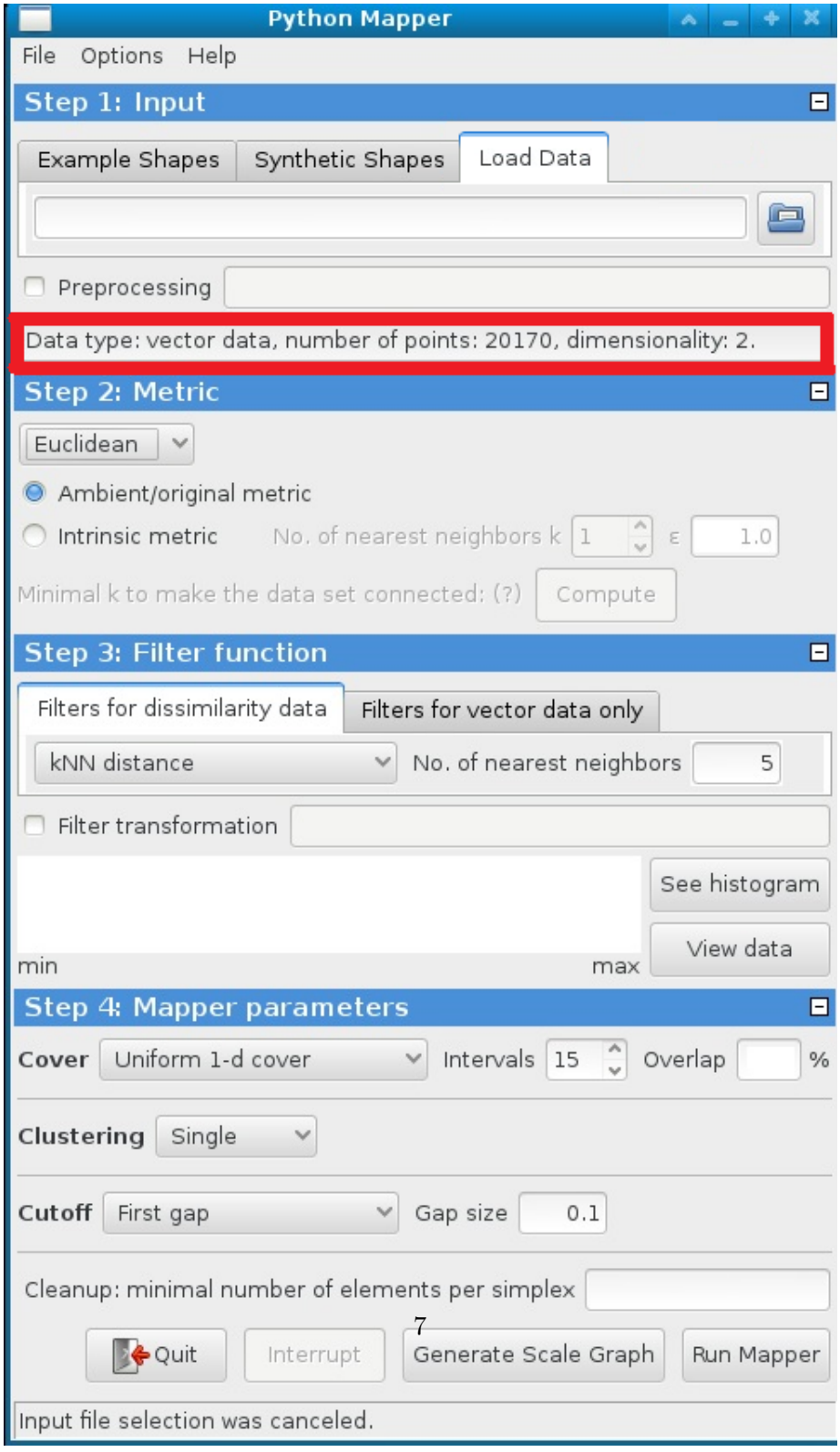


Figure 6: Choose a metric: Euclidean, Minkowski or Chebychev

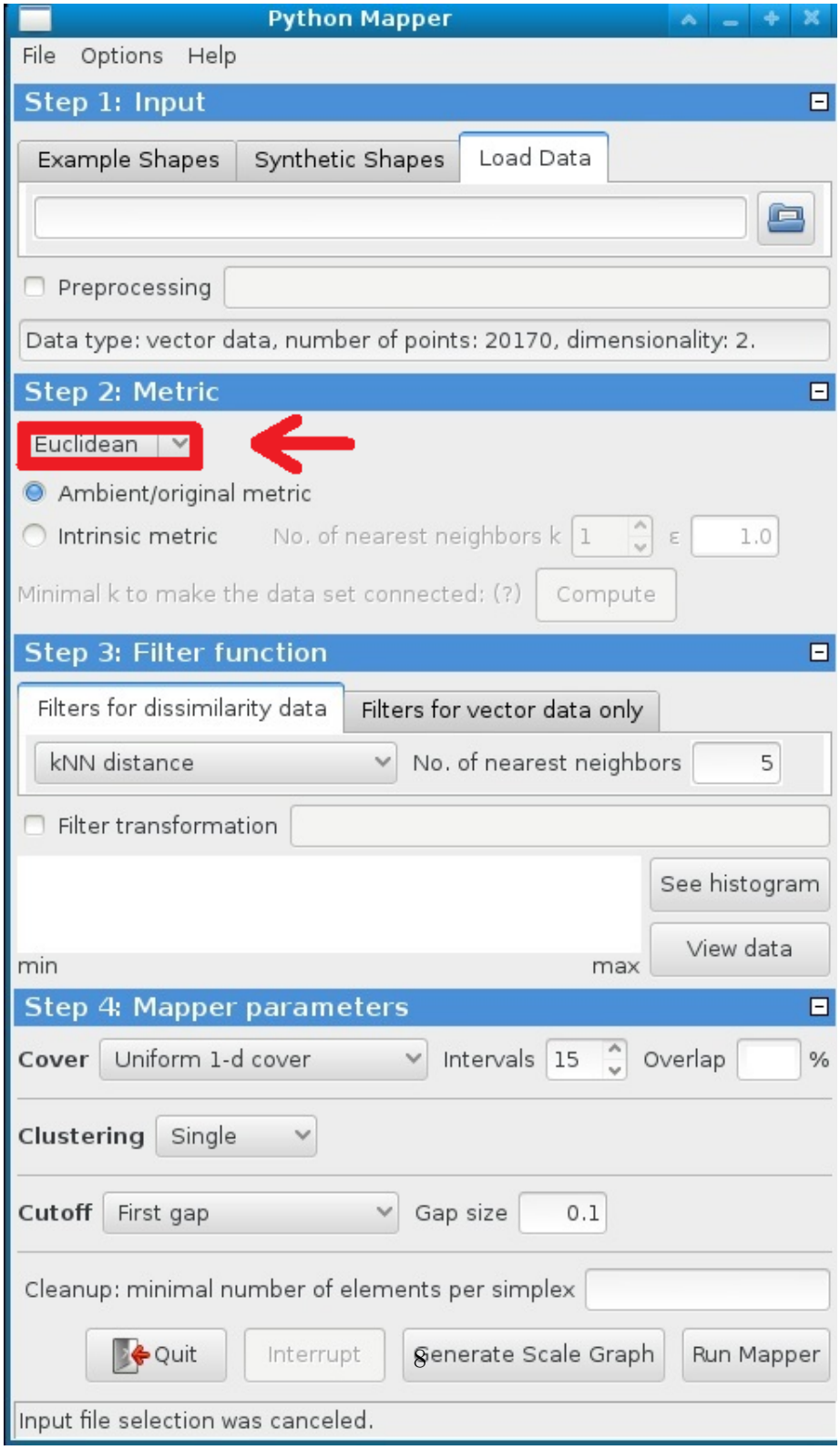


Figure 7: Choose a filter function: Eccentricity, kNN distance, Distance to a measure, Density (Gaussian Kernel), Graph Laplacian or Distance matrix eigenvector

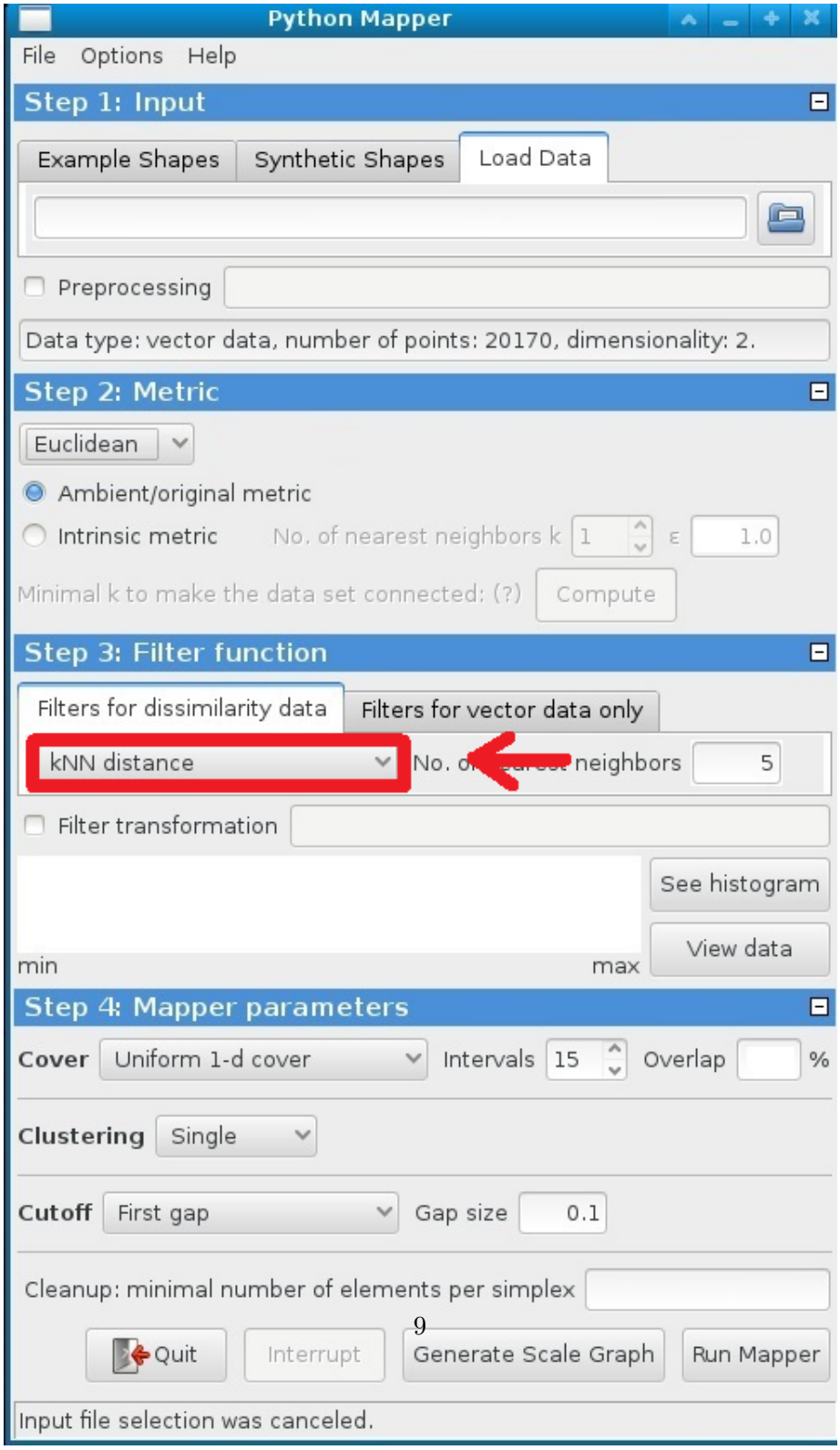


Figure 8: View data (as long as it is 1-dimensional, 2-dimensional or 3-dimensional)

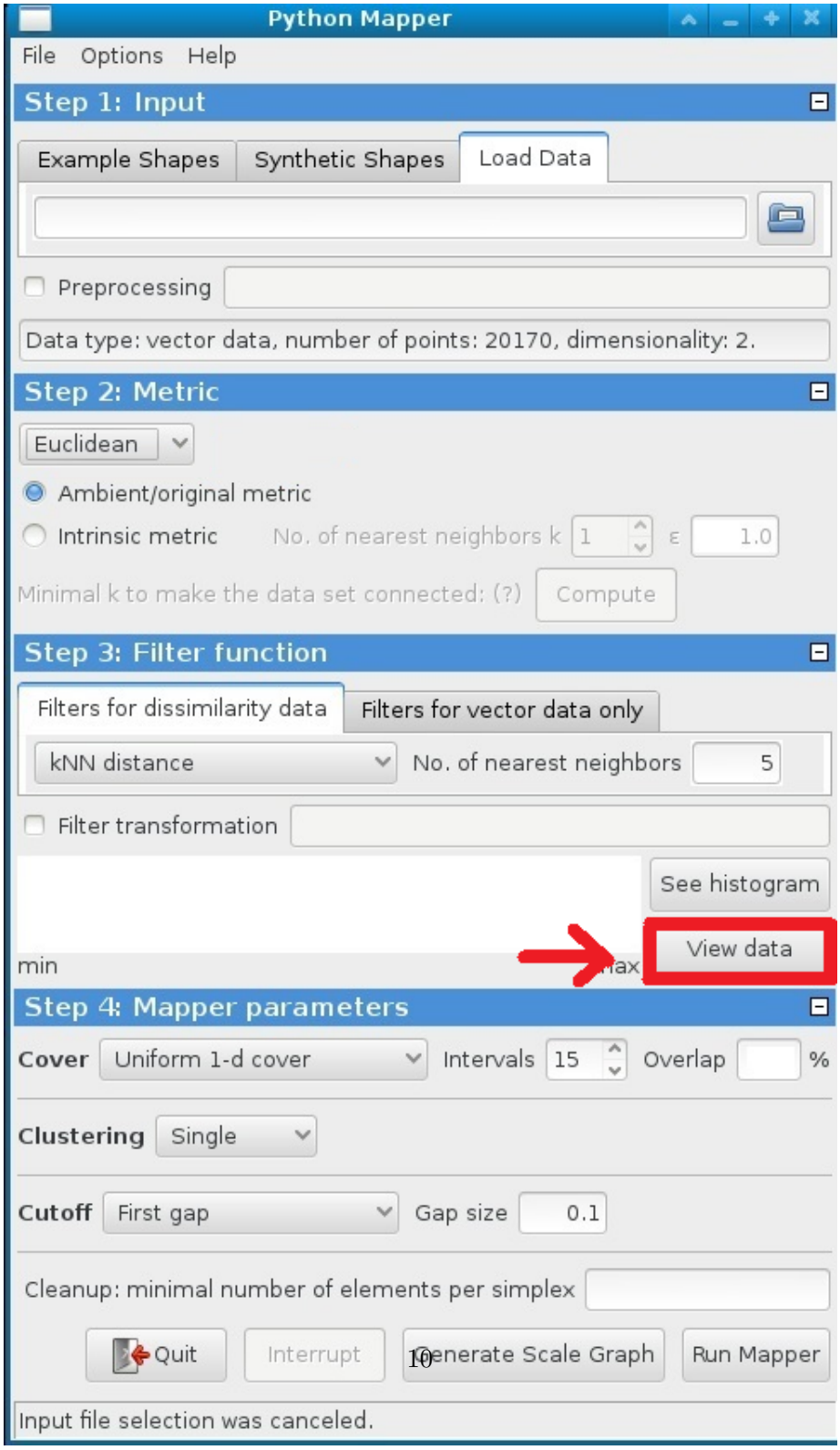


Figure 9: Choose the type of cover, number of intervals and percentage of overlap between successive intervals

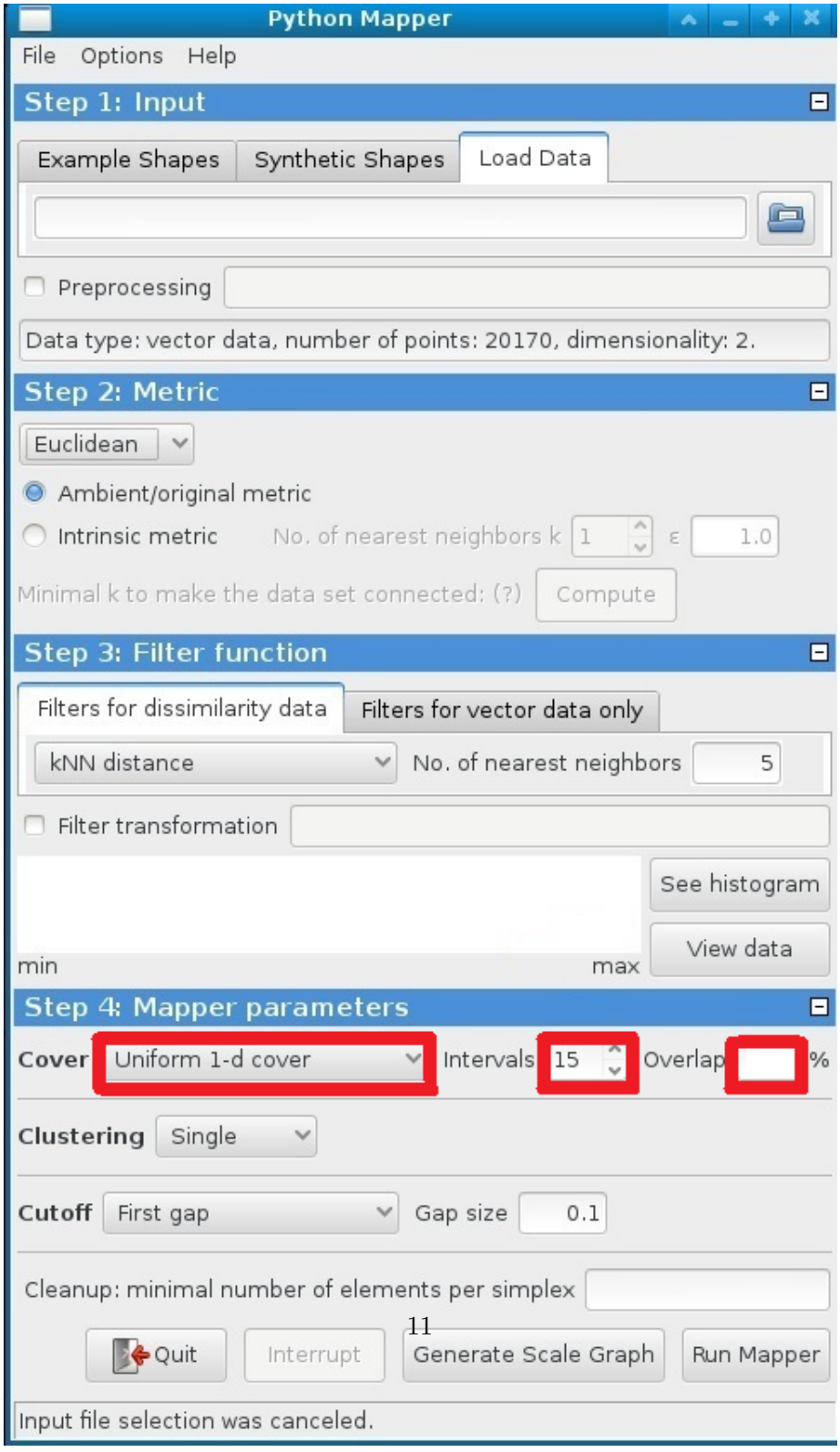


Figure 10: Choose clustering algorithm: Single, Complete, Average, Weighted, Median, Centroid, Ward

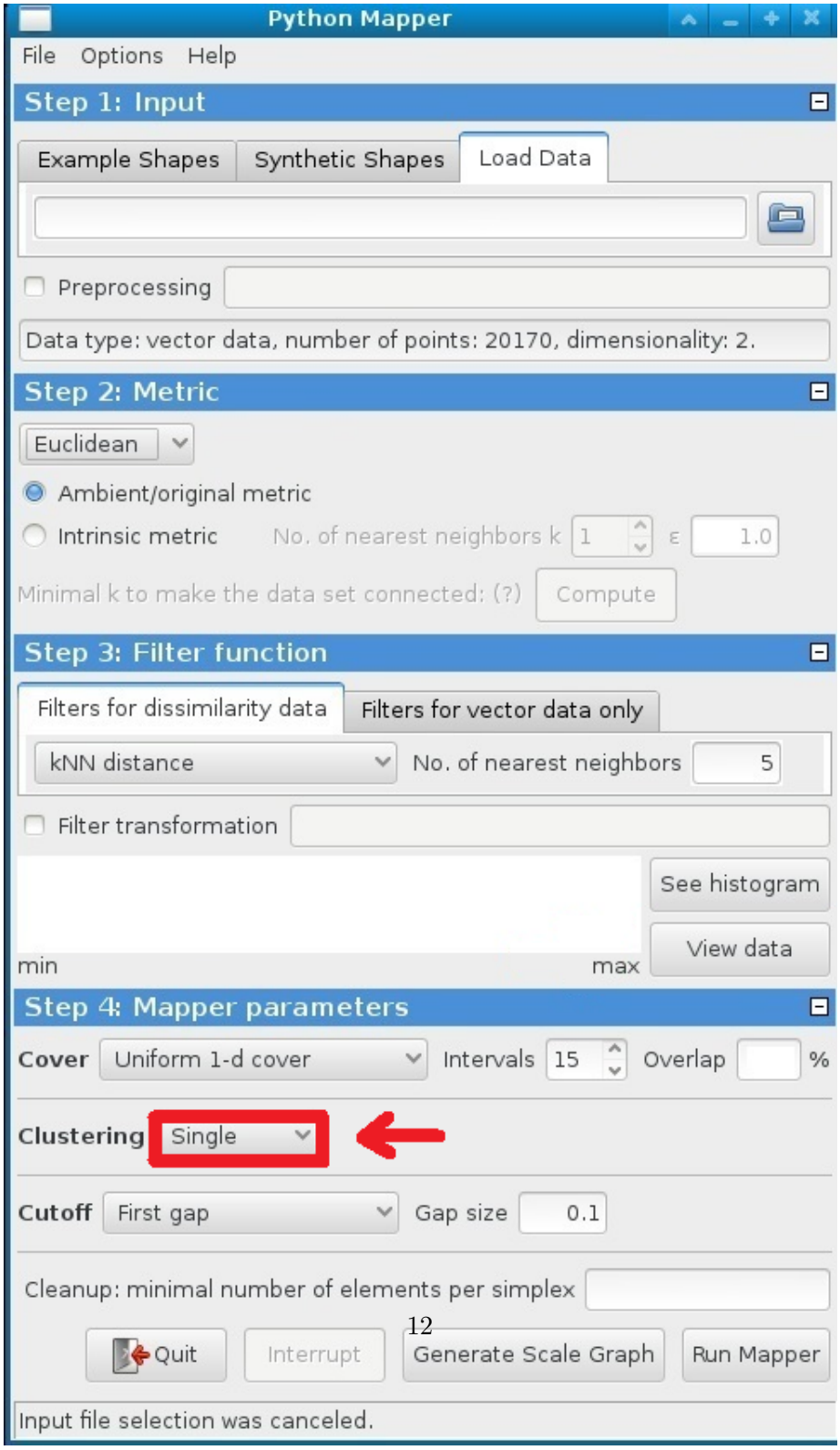


Figure 11: Run MAPPER

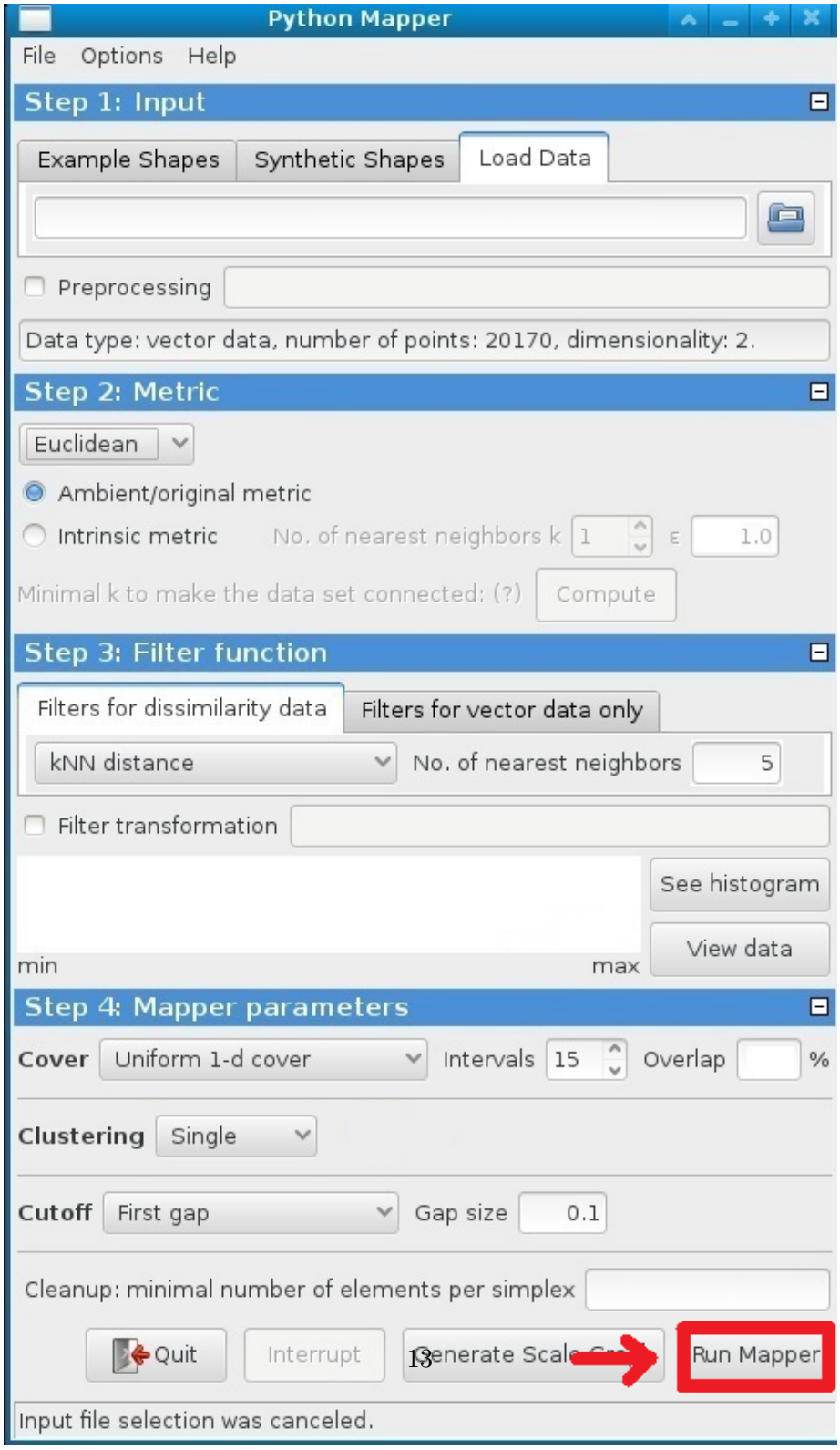


Figure 12: Sample Output