Spora: A Journal of Biomathematics

Volume 8 Article 5

2022

Clustering for the Neophyte: An R Shiny App for Self-Organizing Maps

Olcay Akman
Illinois State University, oakman@ilstu.edu

Zury Betzab-Marroquin
Scripps College, zmarroqu1225@scrippscollege.edu

Christopher Hay-Jahans
University of Alaska Southeast, cnhayjahans@alaska.edu

Joshua Walsh
University of Alaska Southeast, joshak87@gmail.com

Trenton Wesley
Harvey Mudd College, trentonjwesley@gmail.com

Recommended Citation

Akman, Olcay; Betzab-Marroquin, Zury; Hay-Jahans, Christopher; Walsh, Joshua; and Wesley, Trenton (2022) "Clustering for the Neophyte: An R Shiny App for Self-Organizing Maps," *Spora: A Journal of Biomathematics*: Vol. 8, 31–37.
DOI: https://doi.org/10.30707/SPORA8.1.1647886301.847652
Available at: https://ir.library.illinoisstate.edu/spora/vol8/iss1/5

Intercollegiate Biomathematics Alliance

SPORA
A Journal of Biomathematics

ISSN 2473-3067 (Print)
ISSN 2473-5493 (Online)

Clustering for the Neophyte: An R Shiny App for Self-Organizing Maps

Olcay Akman[1], Zury Betzab-Marroquin[2], Christopher Hay-Jahans[3,*], Joshua Walsh[3], Trenton Wesley[4]

*Correspondence:
Christopher Hay-Jahans,
Dept. of Natural Sciences,
University of Alaska
Southeast, 11066 Auke Lake
Way, Juneau, AK 99801 USA
cnhayjahans@alaska.edu

Abstract

This article provides an outline of clustering, key stages in creating self-organizing maps for purposes of clustering, instructions on how to use a free online R Shiny app that constructs self-organizing maps for data provided by users, and interpretations of the graphics produced.

Keywords: Unsupervised classification algorithm, competitive learning algorithm

1 Introduction

These days it is not uncommon to have high-dimensional massive data sets in research projects, and finding similarities among attributes for such data sets is at best difficult. The use of advanced software in multivariate data analysis has become a requirement, and reducing the computational cost to analyze data has become vitally important. However, software is typically expensive and the coding involved can be time consuming.

This article presents an online open-source app prepared specifically to perform clustering, one possible avenue in the analysis of high-dimensional data. The app is implemented using *R Shiny* [9], an R package with which interactive web apps can be built using R, [8]. One of the nice features of R Shiny is that these apps can then be hosted as standalone apps on a webpage.

In what follows, an overview of clustering and the clustering algorithm used by the app is presented. This overview is along the lines of the presentation in [1]. Then, the key steps in using the app are outlined and, following this, a complete example of its implementation with relevant discussions and interpretations of the graphics produced is provided.

Clustering

Informally, a *cluster* refers to a group of observations within a larger data set that exhibit similar attributes, and *clustering* refers to the actual process of partitioning the whole data set into reasonably homogeneous groups. The clustering process itself uses an algorithm that takes advantage of a similarity measure of some form to classify observations in a data set in a manner that permits the grouping.

Since clustering algorithms classify data observations into classes based on attributes present in the data, and without direction from the analyst, these algorithms are referred to as *unsupervised classification algorithms*. While there are several such algorithms, see for example [1], the app under discussion uses what is called a *self-organizing map*.

Self-organizing maps

As mentioned in [1], a self-organizing map (SOM), also called a *Kohonen map*, makes use of *artificial neural networks* to produce a low-dimensional graphical representation of high-dimensional data. This is done in a way that not only preserves the structure in the original data, but also brings to light any similarities present in the data by providing information that can be used to construct visual aids to identifying clusters within the data. The process by which a SOM accomplishes a clustering of data in a high-dimensional data set can be summarized as follows.

In the simplest of terms, a SOM takes *input nodes*, observations from the data, and maps these to a collection of *output nodes* which provide the information needed to produce a low-dimensional graphical representation of clusters present in the data. The process by which this is accomplished involves the use of what is referred to as a *competitive learning algorithm* which runs through the following stages.

Suppose the data comprise n rows of observations with columns typically associated with a collection of $p \geq 3$ variables. Corresponding to each row, $i = 1, 2, \ldots, n$, of the data, let the vector $\mathbf{x}_i = (x_{i1}, x_{i2}, \ldots, x_{ip})$ contain the preferably scaled non-dimensional entries of the i^{th} row in the data. The index i identifies an input node.

[1]Center for Collaborative Studies in Mathematical Biology, Illinois State University, Normal, IL, [2]Department of Mathematics, Scripps College, Claremont, CA, [3]Department of Natural Sciences, University of Alaska Southeast, Juneau, AK, [4]Department of Mathematics, Harvey Mudd College, Claremont, CA

Stage 1 – Initialization A set of units representing the output nodes are typically arranged in a square lattice of dimensions at most $\lfloor\sqrt{n}\rfloor \times \lfloor\sqrt{n}\rfloor$. Then the connection between each input node i and every output node j is represented by a *weight vector* $\mathbf{w}_j = (w_{j1}, w_{j2}, \ldots, w_{jp})$ having entries from the interval $[0, 1]$ assigned either randomly, or using prior knowledge of the data set.

Note: From a biological point of view, the output nodes in a SOM can be associated with *neurons* and the components of the vectors $\mathbf{w}_j$ are associated with *synapses*. These form the artificial neural network mentioned earlier. These terms will not, however, be used in this paper. Rather, an input node will be referred to as such, or as a data observation, and an output node will be referred to as an output node, or simply a node when this is obvious.

After the initialization of the weight vectors, the algorithm begins a sequence of iterative processes.

Stage 2 – Training the weight vectors To start this stage, a data observation (an input node), say $\mathbf{x}_i$, is selected at random.

Finding the Best Matching Unit: The similarity of $\mathbf{x}_i$ to each of the weight vectors $\mathbf{w}_j$ is measured using a distance function, commonly the Euclidean distance. That is, for the chosen input node i and for each output node j, the distance

$$d(\mathbf{x}_i, \mathbf{w}_j) = \sqrt{\sum_{k=1}^{p} (x_{ik} - w_{jk})^2}$$

is calculated. The weight vector that minimizes this distance is called the *winning weight vector*, or *best matching unit* for $\mathbf{x}_i$. Let J denote the best matching unit for the output node i, that is,

$$J = \arg\min_j \{d(\mathbf{x}_i, \mathbf{w}_j)\}.$$

Then, once the best matching unit J for the input node i is found, a process referred to as *activation* takes place for all weight vectors within some predetermined neighborhood of the best matching unit. This leads to the *competitive learning* part of the algorithm.

Training the Weight Vectors: Denote the predetermined neighborhood of the best matching unit by $N(J)$ and define

$$h_{Jj} = \begin{cases} \eta(t), & j \in N(J) \\ 0, & j \notin N(J), \end{cases}$$

where the monotonically decreasing function $\eta(t)$ defines the learning rate. See, for example, [1] for further details on defining neighborhoods and on options available for learning rate functions.

For the `som` function, which the app uses, the learning rate decreases linearly from 0.05 to 0.01, [7, p. 19], and the neighbourhood decreases linearly from 2/3 of the nearest nodes on the map to zero, see [2, p. 6] and [6]. When the neighborhood is 1 or less, only the winning weight vector learns. Hence, for this app, the SOM starts with a cooperative and competitive learning process, then switches to a strictly competitive learning process.

Now, let $t = 1, 2, \ldots, T$ denote time-steps, with $t = 1$ being the starting time-step. Then the current weight vectors, denoted $\mathbf{w}_j(1)$, are updated (they are trained) using a predetermined number of iterations of the activation function

$$\mathbf{w}_j(t+1) = \mathbf{w}_j(t) + h_{Jj}\left[\mathbf{x}_i - \mathbf{w}_j(t)\right].$$

Once all of the weight vectors in the neighborhood of the node J have been updated, or trained, another different input node is selected at random and Stage 2 is repeated. The process ends once all input nodes have been selected, and all of the weight vectors have been trained for every input node.

Stage 3 – Visualization At the termination of the iterations in Stage 2 there are available the input nodes, the output nodes arranged in a square lattice, and the trained weight vectors connecting each input node to output nodes.

It is worth noting that the trained weight vectors themselves will be partitioned into $K \leq n$ clusters. It is this information that is used to project graphical representations of similarities, or clusters, in the data onto the two dimensional lattice of output nodes.

Those interested in delving deeper into the finer details of self-organizing maps can find further information on the underlying theory and development of self-organizing maps in, for example, [2] or [5].

2 The App in a Nutshell

The R Shiny SOM app provides a user interface for the `som` function contained in package `kohonen`, [2] and [7]. This app can be accessed at `https://github.com/iba-community/R-Shiny-for-Clustering`. Once started up, there are three tabs in the app that navigate the user through the three main windows.

The **Introduction** tab opens the *Introduction* window in which the app starts up, see Figure 1. Here the user is introduced to the purpose of this application, given general directions on how to use it, and given descriptions of the plots that it produces.

The **Import Data** tab opens the *Import Data* window, see Figure 2. This is where users are given the option to upload their own data by selecting the option *Upload Data*, or explore the app with data provided in the app by selecting *Use Sample Data*.

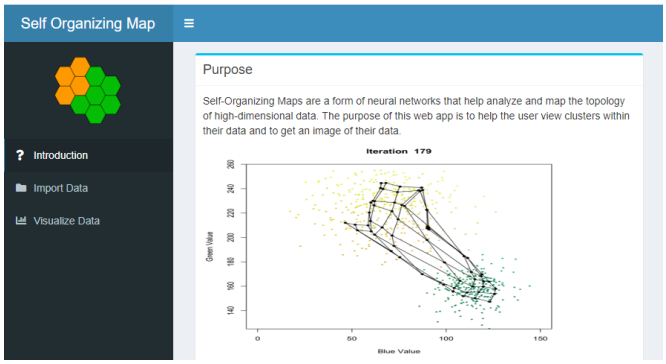

Figure 1: The Introduction window.

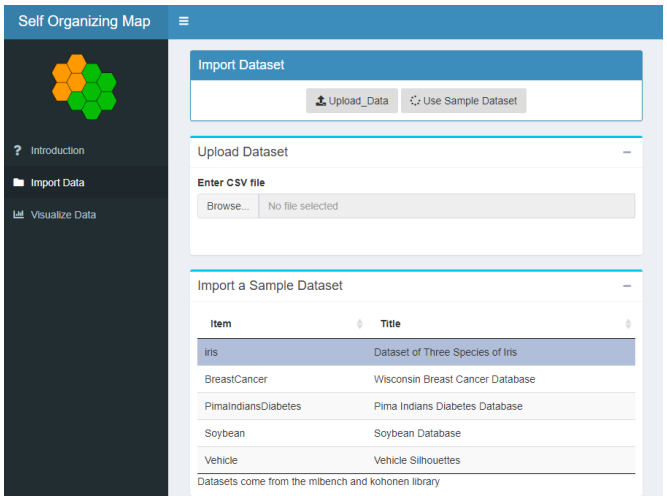

Figure 2: The Import Data window.

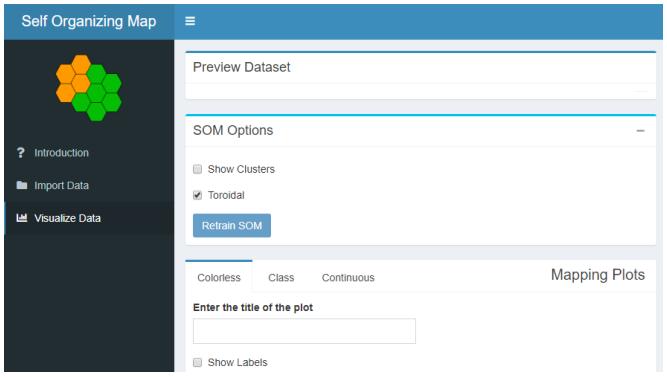

Figure 3: The Visualize Data window.

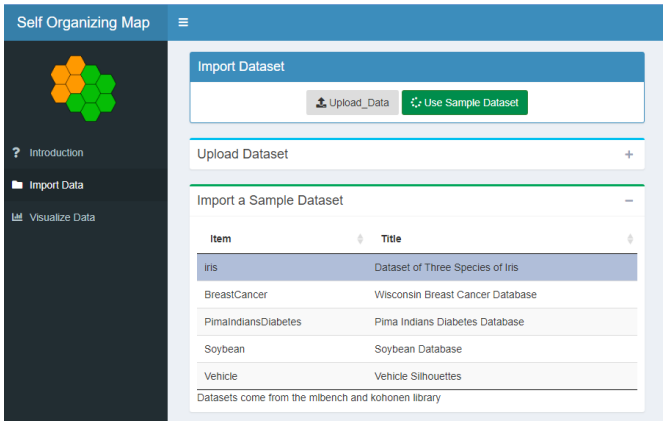

Figure 4: The Use Sample Dataset button is green because it has been pressed, and the iris dataset is highlighted because it has been selected.

Note: Should the user wish to upload their own data, then the data must be as a *.csv file, checked for missing entries, and organized such that rows represent observations and columns represent variables for which the observations are made. While not a requirement, one of the columns may be a labels column.

The **Visualize Data** tab opens the *Visualize Data* window, see Figure 3. Here the user selects the variables of interest and sets the various SOM parameter values as desired. The graphics produced by the SOM app are automatically displayed in this window. Illustrations of choices in this window appear in the following complete example.

3 A Complete Example

Fisher's iris data [3], one of the most common benchmark sample data sets, and a popular data set for validation purposes on clustering algorithms, is used in this example, see for example, [3] and [4, p. 3]. Here is a brief description of the data.

Observations on 50 each of three species of the iris flower family, *iris setosa*, *iris virginica*, and *iris versicolor* are obtained (150 observations total) under four variables. Data under these four variables are obtained through measurements of the sepal length, sepal width, petal length, and petal width for each flower. A fifth column in the data contains the labels variable which identifies the species to the which each flower belongs. It should be noted that the iris setosa species is clearly distinguishable from the other two species, while the iris versicolor and virginica are not as clearly distinguishable from each other.

To start things off, click on the **Import Data** tab, and then select *Iris* from the *Import a Sample Dataset* option, see Figure 4.

Next, click on the **Visualize Data** tab and select the variables of interest: sepal length, sepal width, petal length, and petal width in the *SOM Options* window,

see Figure 5. Then, for this example, set the *SOM Size* to 5. This sets the dimensions of the square lattice for the nodes to 5×5. Different dimensions can be chosen, as long as they do not exceed $\lfloor \sqrt{n} \rfloor$.

Note that the variables can also be selected in the *Preview Dataset* window, see Figure 6, by clicking on an observation of the variable (be aware that clicking on the variable does NOT select the variable, instead, clicking on the variable reorganizes the data), or in the *SOM Options* window by selecting the variables from the drop-down menu. Notice that the boxes for *Show Clusters* and *Toroidal* are NOT selected, however, turning each of these selections on and off can assist in identifying clusters in the data.

Side Note: It is recommended that a SOM should be retrained several times before coming to any conclusions about the data. The reason given for this is because of the randomness involved in the SOM initialization process, [2]. To retrain the SOM, click the *Retrain SOM* button in the *Visualize Data* window. Graphs are automatically updated every time a retraining is performed, and plots of interest can then be downloaded with the *Download* button located at the bottom-left of every graphic. Lastly, *Toroidal* is on by default and turns the map into a torus.

Examples of each plot produced, along with relevant options and interpretations are now given.

Colorless mapping plot

The *Colorless Mapping Plot* shows observations from the data as small circles contained in hexagons, the hexagons being the output nodes, see Figure 7. Observe that some of the nodes are associated with observations and some are not. Observations associated with a node (that is, within a hexagon) and with close-by nodes have similar weight vectors, indicating the corresponding data observations have similar attributes.

The SOM algorithm is unsupervised, as are all clustering processes, so the `som` function does not require labels. However, labels can be added afterwards if the a labels variable is available in the data. For example, the iris data has "Species"as the labels variable in the last column of the data set, which can be seen in Figure 6. To add labels and colors to the mapping plot, click on *Show Labels*, and select the variable Species from the drop-down list. The mapping plot in Figure 8 colors the observations by the associated species, and provides a legend.

Retrain the map several times until the cluster of iris setosa is clearly separated from the partially mixed clusters of iris virginica and iris versicolor. Follow the position of the setosa cluster on the map after every retraining, and observe that the setosa cluster will be clearly separated from the other mixed clusters, and can appear in a different corner of the map after each re-training. Notice that

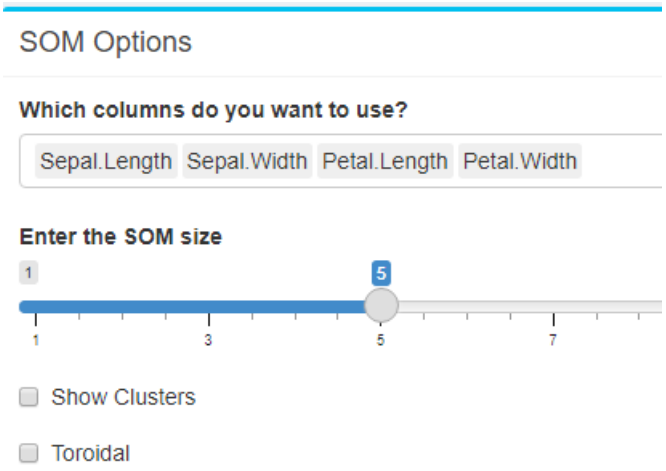

Figure 5: The four variables are selected from the drop-down list. The SOM size is set to 5, and Show Clusters and Toroidal are NOT selected.

Preview Dataset

	Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species
1	5.1	3.5	1.4	0.2	setosa
2	4.9	3	1.4	0.2	setosa
3	4.7	3.2	1.3	0.2	setosa
4	4.6	3.1	1.5	0.2	setosa
5	5	3.6	1.4	0.2	setosa
6	5.4	3.9	1.7	0.4	setosa

Figure 6: The four variables are selected from the iris data. The fifth column is named Species, and is NOT selected.

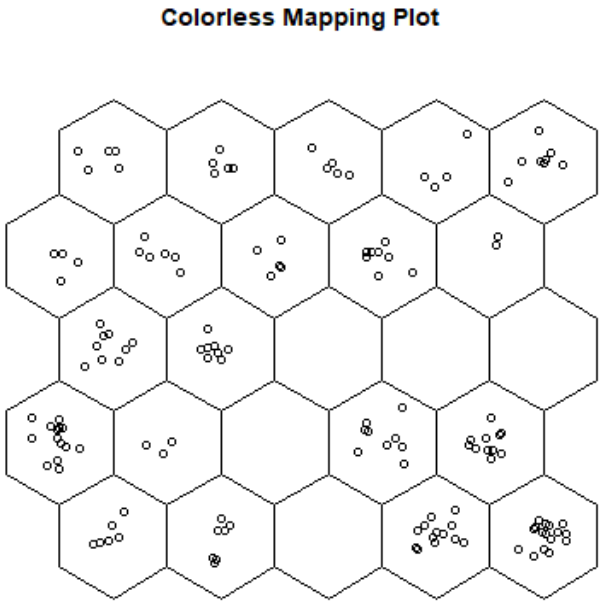

Figure 7: The bottom-right cluster of data is separated from the top-left mixed-cluster of data by the five empty nodes.

if *Toroidal* is on, then the setosa cluster will not always show up in the corners of the map.

Class plot

The *Class Plot* requires a labels column in the data which must be selected from the drop-down list. This plot follows directly from the Colorless Mapping Plot with a legend. To get the Class Plot for the iris data, select the variable Species. Then the classes are updated with the species label, and each of the nodes (again represented by hexagons) is colored according to the color associated with the most frequently occurring species tied to the node, see for example [4, p. 15]. As the included legend indicates, grey hexagons represent nodes that do not contain any observations.

For comparison, in Figure 8, the bottom left hexagon of the map has five iris versicolor and one iris virginica observation, while, in Figure 9, the bottom-left hexagon of the map is purple, which is the color assigned to the most frequently occurring species associated with this node, iris versicolor. Additionally, the grey hexagons represent the clear separation of the setosa (green) cluster from the virginica (orange) and versicolor (purple) mixed clusters.

Continuous response map

The *Continuous Response Map*, see Figure 10, uses classical multidimensional scaling–`cmdscale` to lower the dimension of the weight vectors for the nodes in the map to 3-dimensions. The new 3D vectors are treated as RGB values for each node, then the node is colored by the RGB value. The map displays clusters with similar hues. For example, in Figure 10 the bottom right cluster of setosa is colored with similar hues of green because the weight's of nearby nodes are similar, while the virginica and versicolor clusters mix hues from orange to purple.

Counts plot

In the *Counts Plot* nodes are represented by discs, see Figure 11, each disc is colored according to the number of observations associated with the corresponding node, see for example [7, p. 13] or [4, pp. 15-16]. The legend for this plot, if included, displays a spectrum of colors with an associated count and the map displays colored discs. As before, grey discs represent nodes that are not associated with any data observations. Observe that in Figure 11 the grey discs clearly separate the bottom right setosa species from the two other species. The setosa cluster is clumped together with nodes having high values. The virginica cluster is more spread out across the top of the map, the data in this cluster being spread out across multiple discs. The versicolor cluster has about the same number of observations as the virginica cluster. However,

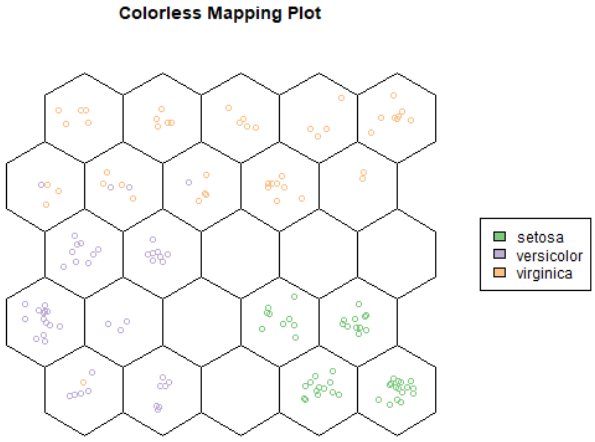

Figure 8: The bottom-right cluster is of the species setosa. The top cluster of data is iris virginica and is partially mixed with the bottom-left cluster of iris versicolor.

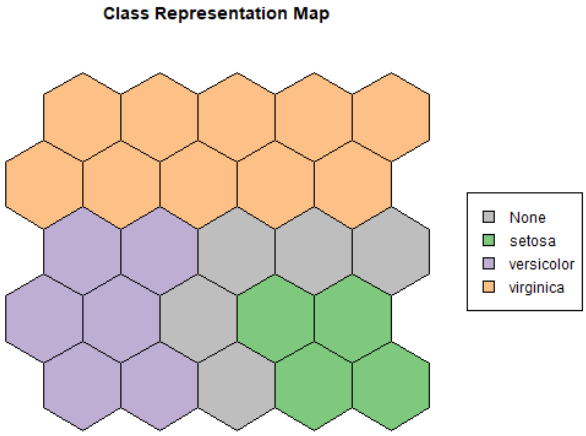

Figure 9: Unlike in Figure 8, the colors are assigned to the whole cells.

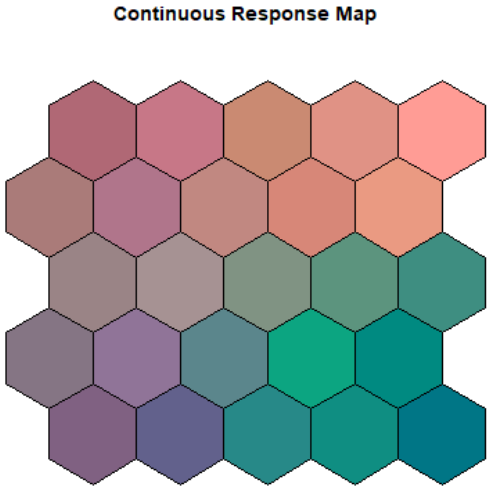

Figure 10: Each node is colored based on each node's weight.

one node belonging to the versicolor cluster has many observations, while the rest of the observations in this cluster are spread over multiple nodes.

Neighborhood distance plot

The *Neighborhood Distance Plot* is the SOM version of a U-matrix plot, see for example [4, pp. 17-18], where nodes are represented by discs. The color assigned to each node represents the sum of the distances to that node's immediate neighbors, [7, p. 13]. The legend for this plot, if included, also displays a spectrum of colors with associated distances, and the map shows colored discs. This means that a node having a color associated with a small distance will have close neighbors, and a node having a color associated with a large distance will have more distant neighbors.

In Figure 12, colors assigned to the discs indicate that the clusters have low values, which implies that neighboring nodes are close together. Alternatively, observe that the majority of the nodes not associated with any data observations have higher values, implying that neighboring nodes are further away. So, one may conclude that observations from the setosa cluster are very likely close to each other in terms of attributes, but further away from observations belonging to the other two clusters. At the same time, neighboring nodes associated with the versicolor and virginica clusters, being partially mixed, are closer together.

Codes plot

In the *Codes Plot*, see Figure 13, nodes are represented by discs containing what look like Pie Charts made up of sectors of circles (wedge-shaped pieces) having different radii. Each wedge represents one of the variables of interest, and the radius of the wedge is proportional to the data value corresponding to the variable for the node in question, [7, p. 13]. The legend for this plot, if included, shows the variables and their associated colors.

In Figure 13, the bottom-right discs representing the setosa cluster suggest that values for the variable sepal width vary considerably, while, values for the other variables barely vary. This contrasts with the versicolor and virginica clusters, where values for all four variables vary considerably from node to node.

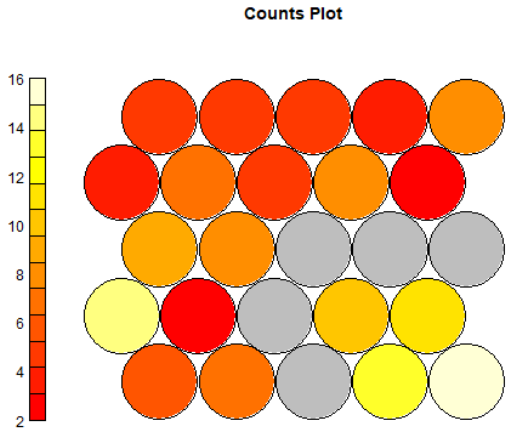

Figure 11: Nodes are colored to show how many observations are within the node. Grey nodes are empty nodes.

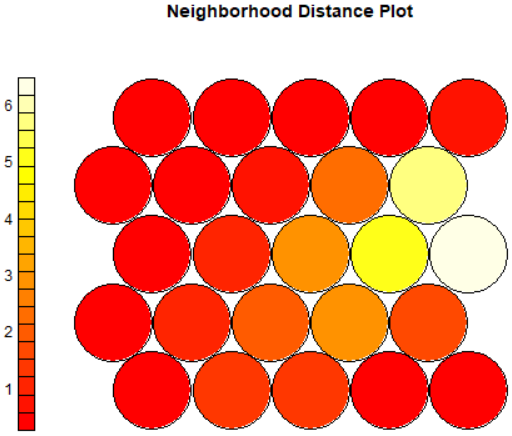

Figure 12: Node colors show distance to immediate neighbors.

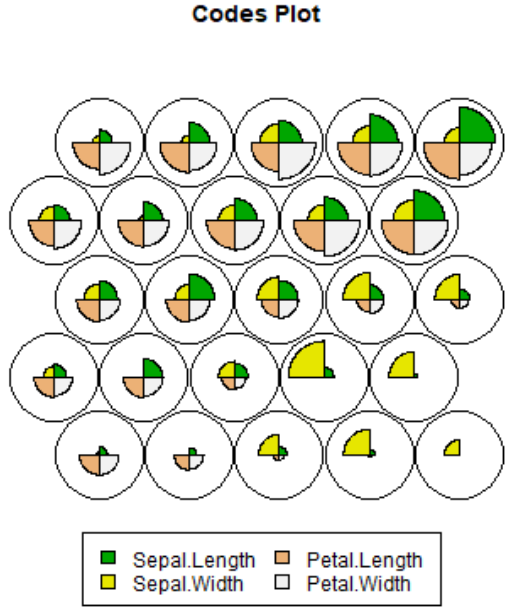

Figure 13: Wedges represent the variables of interest.

Acknowledgements

Work leading to the R Shiny SOM app, and this paper, was motivated and begun at the 2021 *Cross-Institutional Undergraduate Research Experience* (CURE) summer workshop. This annual workshop is sponsored by the *Intercollegiate Biomathematics Alliance* (IBA) and the

Center for Collaborative Studies in Mathematical Biology of Illinois State University.

Author Contributions

The idea of preparing an R Shiny app was conceived by Olcay Akman and Christopher Hay-Jahans at the 2021 IBA CURE workshop. Zury Betzab-Marroquin, Joshua Walsh and Trenton Wesley conducted preliminary research into the topics of clustering and self-organizing maps. Trenton Wesley then branched into R Shiny and the actual coding of the app, and Zury Betzab-Marroquin and Joshua Walsh focused on the theoretical and quantitative background material. Olcay Akman and Christopher Hay-Jahans served as mentors for this project, providing periodic guidance in conducting research, coding, and writing.

References

[1] Akman, O., Comar, T., Gonzales, J., & Hrozencik, D. (2019). Data clustering and self-organizing maps in biology. In *Algebraic and Combinatorial Computational Biology* (pp. 351-374). Academic Press.

[2] Buydens, L. M., & Wehrens, R. (2007). Self-and super-organizing maps in R: the Kohonen package. *Journal of Statistical Software, 21*, 1-19.

[3] Fisher, R.A. (1936). Iris Data Set. *UC Irvine Machine Learning Repository*. `https://archive.ics.uci.edu/ml/datasets/iris`.

[4] Kamath, C., & Ponmalai, R. (2019). *Self-organizing maps and their applications to data analysis* (No. LLNL-TR-791165). Lawrence Livermore National Lab.(LLNL), Livermore, CA (United States).

[5] Kohonen, T. (1998). The self-organizing map. *Neurocomputing: An International Journal, 21*(1–3), 1–6. `https://doi.org/10.1016/S0925-2312(98)00030-7`.

[6] Kruisselbrink, J., & Wehrens, R. (2018). Flexible self-organizing maps in kohonen 3.0. *Journal of Statistical Software, 87*, 1-18.

[7] Kruisselbrink, J., & Wehrens, R. (2019). Package 'kohonen'. *The Comprehensive R Archive Network*. `https://cran.r-project.org/web/packages/kohonen/kohonen.pdf`.

[8] R Core Team (2021). *R: A language and environment for statistical computing*. R Foundation for Statistical Computing Vienna, Austria. `https://www.R-project.org/`

[9] R Shiny. (2014). *shiny: Easy web applications in R*. RStudio, Inc. `https://shiny.rstudio.com`.

[10] Xu, R., and D. Wunsch. (2009). *Clustering*. Wiley-IEEE Press.