Scientific Research Publishing

Journal of Data Analysis and Information Processing, 2020, 8, 229-240
https://www.scirp.org/journal/jdaip
ISSN Online: 2327-7203
ISSN Print: 2327-7211

Resizable, Rescalable and Free-Style Visualization of Hierarchical Clustering and Bioinformatics Analysis

Ruming Li

School of Information Engineering, Baise University, Baise, China
Email: rli@alumni.lsu.edu

How to cite this paper: Li, R. (2020) Resizable, Rescalable and Free-Style Visualization of Hierarchical Clustering and Bioinformatics Analysis. *Journal of Data Analysis and Information Processing*, **8**, 229-240.
https://doi.org/10.4236/jdaip.2020.84013

Received: June 24, 2020
Accepted: September 26, 2020
Published: September 29, 2020

Abstract

Graphical representation of hierarchical clustering results is of final importance in hierarchical cluster analysis of data. Unfortunately, almost all mathematical or statistical software may have a weak capability of showcasing such clustering results. Particularly, most of clustering results or trees drawn cannot be represented in a dendrogram with a resizable, rescalable and free-style fashion. With the "dynamic" drawing instead of "static" one, this research works around these weak functionalities that restrict visualization of clustering results in an arbitrary manner. It introduces an algorithmic solution to these functionalities, which adopts seamless pixel rearrangements to be able to resize and rescale dendrograms or tree diagrams. The results showed that the algorithm developed makes clustering outcome representation a really free visualization of hierarchical clustering and bioinformatics analysis. Especially, it possesses features of selectively visualizing and/or saving results in a specific size, scale and style (different views).

Keywords

Hierarchical Clustering, Clustering Visualization, Dendrogram Drawing, Tree Drawing, Resizable and Rescalable, Free-Style Visualization

1. Introduction

Hierarchical cluster analysis plays a great role in exploratory data analysis, information classification, phylogenetics and so forth. The clustering is capable of processing the data such that each of the items is treated as a cluster, pairwise clusters can be merged as one cluster on a similarity basis, and then the paired clusters are joined on a relaxed similarity basis until all the clusters are merged as a single cluster. It is the agglomerative hierarchical clustering as a default fashion. Alternatively, this fusion can be carried out reversely (divisive fashion). Whichever fashion the clustering may take, it performs computing and merging all the items and this process is completed [1]-[10].

The next process is how we represent its clustering results—the last but not least part of our work. Usually, clustering results are represented graphically with a tree diagram such as an evolutionary or phylogenetic tree, which is termed as dendrogram (including cladogram and phenogram). It is a tree-like branching diagram representing a hierarchy of classes or categories based on the degree of similarity or the number of shared characteristics or traits especially in biological taxonomy. A dendrogram is drawn with the line connecting two clusters that are closest to each other, the line connecting two primary mergers that are closest to each other, the line connecting two secondary mergers that are closest to each other, and the like. This stepwise process continues until all mergers and items are connected as a single big merger (*i.e.*, a fusion of clusters).

The dendrogram helps visualize the relationships among clusters, which is the way we use hierarchical data to understand phylogenetics, taxonomy, information architecture, etc. Being resizable (changeable size) and rescalable (adjustable scale in length and width) are desirable for these purposes. However, this graphical representation is static with a fixed size and scale between length and width. A static dendrogram is not resizable and rescalable so that a desired tree view of clusterings is not available. As far as the topic is concerned, almost all existing mathematical or statistical software may not be capable of drawing dendrograms in a free style manner. A nonresizable and nonscalable dendrogram cannot actually meet the needs of viewing clustering results in an arbitrary way. As a result, the static dendrogram is not pragmatically usable but a merely graphical output of data clusterings. Most commercial software can provide only such a simple output function such as MatLab, SPSS and SAS. The typical dendrogram drawing programs available are TreeView, Phylip, Paup, MEGA (Molecular Evolutionary Genetic Analysis) and so on, which provide a representation of graphical clusterings [3] [4] [6] [7]. Some of these programs may be dynamic and the others are static. Some of the dynamic programs (e.g., TreeView) are rescalable but not resizable or vice versa, as well as not in a free-style fashion. Even if these programs are both resizable and rescalable in time, their algorithms are not known to the world or known in different approaches and programming languages (e.g., R and Python). They only possess the basic function for a dendrogram. This impedes the communications between developers to better understand, improve or enhance these algorithms.

The author previously published and presented a textual tree drawing of biological data clusterings but it was not free-style and graphical [11]. This paper introduces an algorithm that is very useful for one to get insight into how a resizable, rescalable and free-style tree is drawn yet usable immediately for developers to build, improve or enhance such programs.

2. Methods

The context of dendrogram drawing programs is not confined to phylogenetic data as is by most of the tree drawing programs and is applied to any other data that are clusterable. Likewise, the metrics used to define the relationships between clusters are not limited to distance measures and can be any other measurements as long as they can generate quantitative differences between clusters [5] [12]. To put it simply, however, the metric data are clustering distances/dissimilarities that are used to define and measure the branch lengths of nodes in a dendrogram. The computer language used to create a dendrogram is Java that is most popularly used programmatic solution than others, although it can be any other language. Thus the Java-based portable program produced by this algorithm can be migrated to and employed in any other systems or platforms.

As we know, a dendrogram is structured as horizontally clustering levels (*i.e.*, nodes) and vertically clustering lines (*i.e.*, branches), and constructed from both ways. Then the two-way tree drawing constitutes the formation of a dendrogram scaled in width and height. There are three possible directions (styles) of tree drawing: the first is drawing a tree from left to right (default view), the second is drawing a tree from right to left (mirror view), and the third is drawing a tree from the bottom up (upright view). Creating a dendrogram is done by drawing different nodes and branches whose graphical representation is composed of pixels. Indeed, a dendrogram is drawn by arranging pixels on the background. The so-called static image of tree drawing is made in essence by a fixed number of pixels whose number is constant such that the image is not alterable. The so-called dynamic image of tree drawing is made essentially by a number of pixels whose number is changeable such that the image is alterable. Therefore, the alterability of an image drawn depends on how pixels are dynamically incremented or decremented over the background. This is just a matter of dynamically redrawing a picture on the canvas or, just as what the algorithm suggests, rearranging pixels at runtime of tree drawing. But how pixels are rearranged while resizing and rescaling a dendrogram is what the algorithm addresses. This algorithm is based on such a "dynamic" drawing instead of "static" one with pixels being freely rearranged. It is described in detail as below.

When a dendrogram is resized and rescaled, the pixels are required to increment or decrement in order to adapt to a new scale and size. What is more, this pixel increment or decrement is a continuous process or called "seamless" progression such that resizing and rescaling a dendrogram is a course of smoothly growing or shrinking. However, the pixels incremented or decremented may not be able to be equally assigned to each of level/line intervals (*i.e.*, the space between clustering levels or branches). If the number of pixels happens to be the same as the number of levels/lines, each of spaces between levels/lines can remove or be assigned one pixel such that each of space widths/heights can decrease or increase. But if the number of pixels is not the same as the number of levels/lines or the former is not the multiple of the latter, the former is divided by the latter and gets a remainder. How these remainder pixels, *i.e.*, those pixels greater or less than or not the multiple of the number of levels/lines, could be allocated to or removed from a dendrogram?

There are two solutions to this issue about the remainder pixels not matching the number of levels/lines: one is to assign the remainder pixels incremented to the intervals of right-left levels/top-down lines or removed from there the remainder pixels decremented. Another solution is to add the remainder pixels to or remove them from the intervals of left-right levels/bottom-up lines. The core code below is the algorithm that implements the resolution to how the remainder pixels are rearranged while resizing and rescaling a dendrogram. The intervals of right-left levels/top-down lines are assumed to be the default spaces between levels/lines where pixels are added or removed. So the algorithm addresses this way of dendrogram drawing, but one can choose the intervals of left-right levels/bottom-up lines and the rationale behind the way is the same.

Suppose we have the following variables whose definitions are given in the graphical context of any dendrogram drawing program:

HC = the class of Hierarchical Clustering
CL = the number of clustering levels or nodes
CN = the number of clustering lines or branches
CK = the check point for forward or reverse drawing
UP = true if upright drawing, false if the upright drawing
is reset to get otherwise drawing
DH = the signed difference in height/width by resizing
PH = the picture height in drawing
PW = the picture width in drawing
UW = the increased/decreased unit width of a tree
HR = the signed remainder pixels of horizontal resizing
VR = the signed remainder pixels of vertical resizing
LSP = level/line spacing adjustment control (spacer)
Tot = the total of signed and accumulated HR
Sum = the sum of signed and accumulated VR
I = the initial PW
J = the initial PH
K = intermediately or temporarily stored value

```
// The core code that implements the algorithm of resizing, rescaling and
// restyling a dendrogram or tree diagram:
DH = UP? J - PH : I - PW; //DH is zero initially in conditional statement
// if UP is true by ?-checking, starting vertical coordinate of tree drawing
// if UP is false, starting horizontal coordinate of tree drawing
if (HC.Mode == 1) {
   // Horizontal/vertical auto-resizing is realized by positive/negative Tot
   Tot += DH;
} else {
   // Handle horizontal drawing width
```

```
    if (DH * HR < 0) {
        CK = 1; // To the reverse direction of resizing
    } else {
        CK = 0; // To the same direction of resizing
    }
    K = HR;    // Store HR value before being changed
    // Get remainder pixels after assigning pixels to clustering levels
    HR = DH % CL;
    if (HR == 0) CK = 0; // No resizing as DH = 0
    DH /= CL; // Get DH that is signed multiple of CL
    UW += DH; // Increase/decrease the unit width of tree
    if (Tot == 0) {
        Tot = Math.abs(HR); // Initialize Tot with HR
    } else {
        if (CK == 0) {
            Tot += Math.abs(HR); //Increment Tot by HR
        } else {
            Tot -= Math.abs(HR); //Offset Tot by reverse HR
            if (Tot > 0) {
                HR = -HR; //Reverse direction of resizing
            } else if (Tot < 0) {
                Tot = -Tot; //Make Tot positive
            }
        }
    }
    if (Tot >= CL) {
        Tot -= CL; //Subtract CL from Tot if Tot grows over it
        //Then each of clustering levels should decrement or increment one pixel,
        // that is, decrease or increase the unit width of tree drawing
        if (HR < 0) {
            UW--; // Decrement the unit width, shrinking tree.
        } else if (HR > 0) {
            UW++; // Increment the unit width, enlarging tree.
        }
    }
    if (HR == 0) HR = K; // If no resizing, restore HR value
}
// Handle vertical drawing height, which is analogous to the horizontal way.
DH = UP? I - PW : J - PH; // DH is evaluated by ?-checking UP value
if (DH * VR < 0) { // Vertical remainder pixels instead of HR
    CK = 1;
} else {
    CK = 0;
}
K = VR;    // Store VR value before being changed
// Get remainder pixels after assigning pixels to clustering lines
VR = DH % CN;    // Number of data points instead of CL
if (VR == 0) CK = 0;
DH /= CN;      // Get DH that is the signed multiple of CN
LSP -= DH;
if (Sum == 0) { //Sum instead of Tot
    Sum = Math.abs(VR);
} else {
    if (CK == 0) {
        Sum += Math.abs(VR);
    } else {
        Sum -= Math.abs(VR);
        if (Sum > 0) {
            VR = -VR;
        } else if (Sum < 0) {
            Sum = -Sum;
        }
    }
}
if (Sum >= CN) {
    Sum -= CN;
    if (VR < 0) {
        LSP++;
    } else if (VR > 0) {
        LSP--;
    }
}
if (VR == 0) VR = K;    // If no resizing, restore VR value.
```

The above code draws a tree from both level-wise and line-wise directions and conspires to form a dendrogram; that is, it is integrated by drawing horizontally by nodes and vertically by branches. So is the tree drawing while resizing and rescaling a dendrogram when pointing the lower right corner (grip handle) of the window using a mouse and holding and dragging it. Implementing the code is outlined as follows:

1) Initialize the signed difference in height/width (DH) with zero by evaluating J-PH for upright drawing or I-PW for both forward and mirrored drawing.

2) Enter into the Hierarchical Clustering mode, accumulate the amount of DH difference and store it in the variable total (Tot).

3) When the difference becomes great enough to handle it (*i.e.*, resizing occurs), set the horizontal drawing width first. If the signed product of the difference and horizontal remainder pixels is negative, resizing to the reverse direction (*i.e.*, shrinking), Otherwise resizing to the same direction (*i.e.*, enlarging).

4) Calculate the amount of horizontal remainder pixels using the difference divided by clustering levels.

5) Calculate the difference that is the signed multiple of clustering levels.

6) Calculate the signed amount of total that determines the direction of resizing but should minus the number of clustering levels if the total grows over it.

7) The negative sign of horizontal remainder pixels shrinks a tree by decrementing the unit width and the positive sign enlarges a tree by incrementing the unit width. If horizontal remainder pixels are zero, restore its previous value before resizing.

8) Second, set vertical drawing height, which is analogous to the horizontal way with vertical remainder pixels instead of HR, number of data points instead of CL and Sum instead of Tot. This redundancy is omitted here to save space.

3. Results

The implementation of the algorithm is made by using the real-world data sets ClustView.txt (for statistical analysis) and TreeView.tre (for bioinformatics analysis) in **Table 1** and **Table 2**.

Table 1. A four-dimensional (4 variables) real data set of 17 real-number objects (data points) that is used as clusterable data input for the dendrogram drawing program built in the software ParCluster v.3.0.

Objects	Variable 1	Variable 2	Variable 3	Variable 4
lau	0.38	626.5	601.3	605.3
ccu	0.18	654.0	647.1	641.8
bhu	0.07	677.2	676.5	670.5
ing	0.09	639.9	640.3	636.0
com	0.19	614.7	617.3	606.2
smm	0.12	670.2	666.0	659.3
bur	0.20	651.1	645.2	643.4
gln	0.41	645.4	645.8	644.8
pvu	0.07	683.5	682.9	674.3
sgu	0.39	648.6	647.8	643.1
abc	0.21	650.4	650.8	643.9
pas	0.24	637.0	636.9	626.5
lan	0.09	641.1	628.8	629.4
plm	0.12	638.0	627.7	628.6
tor	0.11	661.4	659.0	651.8
dow	0.22	646.4	646.2	647.0
lbu	0.33	634.1	632.0	627.8

Table 2. A real phylogenetic data set of 15 whole-number taxa (taxonomic data entries) that is used as clusterable data input for the tree drawing program built in the software ParCluster v.3.0.

```
BEGIN TREES;
TRANSLATE
1     'Pi3b-ST-Sh',
2     'Pi3c-ST-Sh',
3     'Pi3a-SP.ST-DC',
4     'Pi3d-ST-Ke',
5     'Pi2B-ST-KE',
6     'PI-IIb-ARPI-SL',
7     'PI-II-NA',
8     'PI-II-NS',
9     'PI-II-NT',
10    'PI-IIb-NC',
11    'PI-II-CA',
12    'PI-IIb-CA',
13    'PI-II-ST',
14    'PI-II-SA',
15    'PI-II-SL';
UTREE * PHYLIP_1=(14,15,(13,(((6,(5,(4,(3,(1,2))))),(10,(7,(8,9)))),(11,12))));
ENDBLOCK;
```

With the hierarchical data ClustView.txt, the algorithm is implemented as the most-used hierarchical clustering through the options of being normalized and average linkage (UPGMA). It outputs the dynamically clustering results by drawing each of successive dendrograms as pixels are rearranged. For such a hierarchical cluster analysis, the default forward views (free-style with the smaller, medium and larger sizes and scales as well as colored labels, etc.) are produced respectively. They are exhibited in Figures 1-3 for demonstration of a resizable, rescalable and free-style data clustering visualization. With the phylogenetic data TreeView.tre, the algorithm is implemented as the typically-used phylogenetic or phylogenomic tree drawing. For such a phylogenetic or evolutionary analysis, the default view (forward), mirror view (backward), and upright view (upward with upright and italic labels) are produced respectively. They are exhibited in Figures 4-7 for demonstration of a resizable, rescalable and free-style tree drawing visualization (Figure 8, Figure 9).

All of the outputs are generated from the dendrogram drawing program built in the software ParCluster v.3.0 and are illustrated as below.

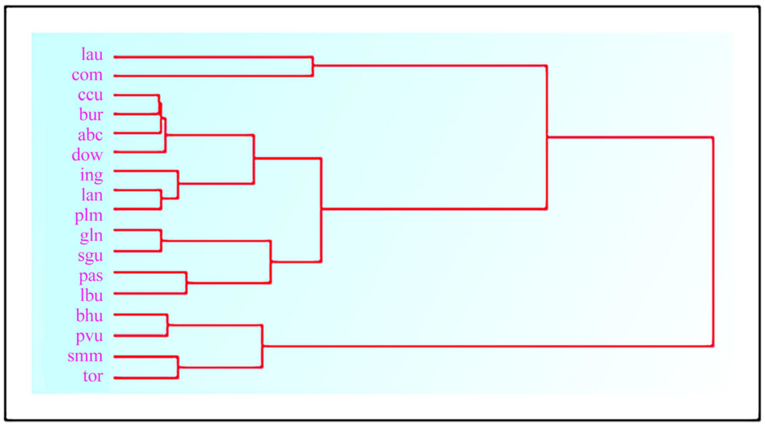

Figure 1. A clustering plot drawn to the right with the small-sized, scaled and styled view using the real data set ClustView.txt (17 data points of 4 dimensions).

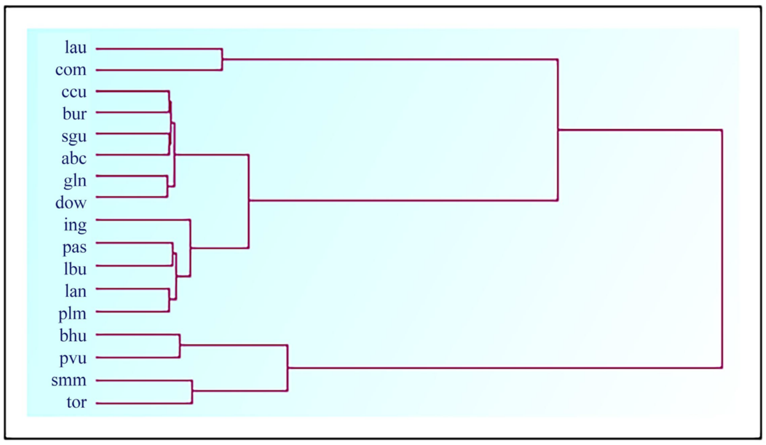

Figure 2. The consecutive clustering plot drawn to the right with the medium-sized, rescaled and restyled view using the real data set ClustView.txt.

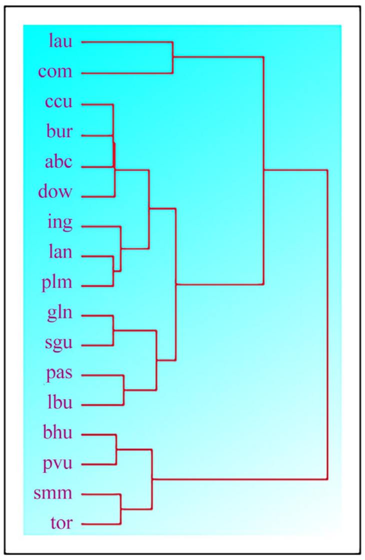

Figure 3. The successive clustering plot drawn to the right with the large-sized, rescaled and restyled view using the real data set ClustView.txt.

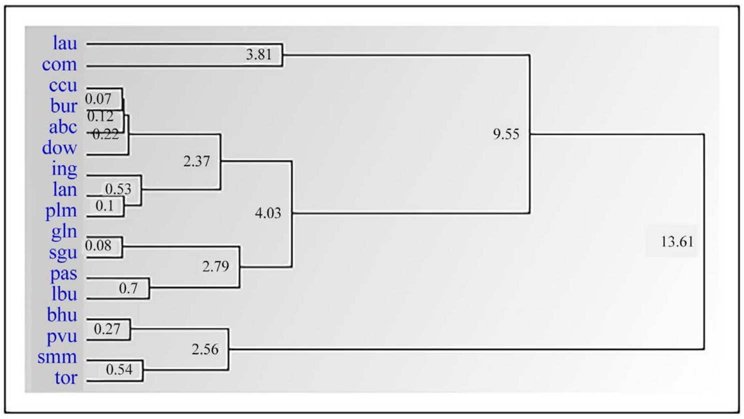

Figure 4. A clustering plot drawn to the right with the node-marked, resized, rescaled and restyled view using the real data set ClustView.txt.

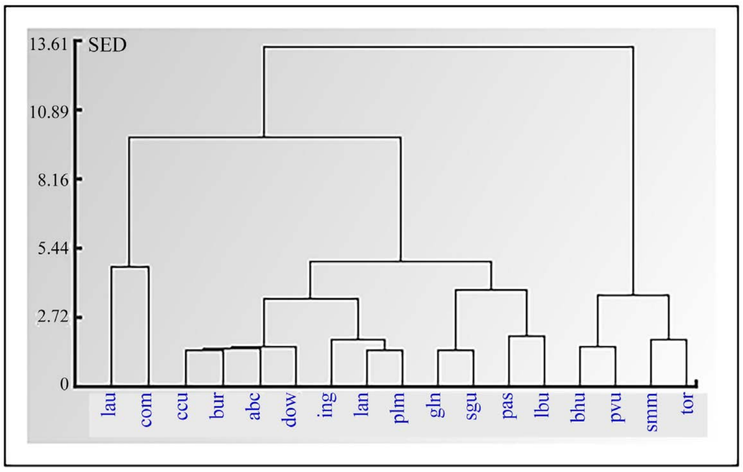

Figure 5. A clustering plot drawn to the upright with the labeled ticks, oblique font, resized, rescaled and restyled view using the real data set ClustView.txt.

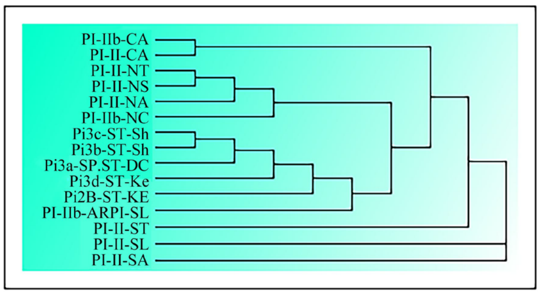

Figure 6. A clustering dendrogram drawn to the right with the default foreground, background and label (item entry name) color using the real phylogenetic data TreeView.tre (15 taxa).

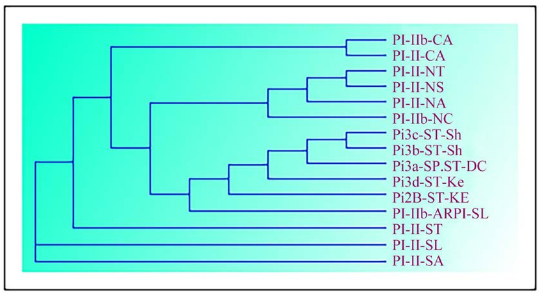

Figure 7. The consecutive clustering dendrogram drawn to the left with the free-style mirrored view and colored labels using the real data TreeView.tre.

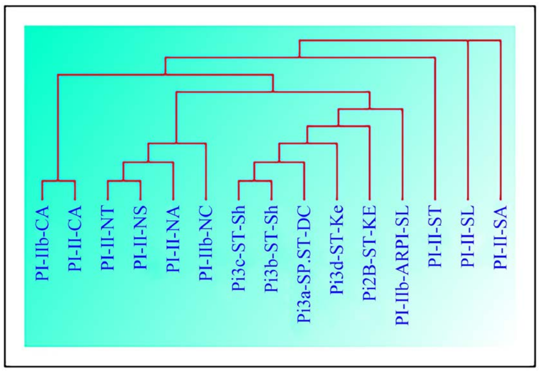

Figure 8. The consecutive clustering dendrogram drawn to the upright with the free-style bottom-up/upward orientation using the real data TreeView.tre.

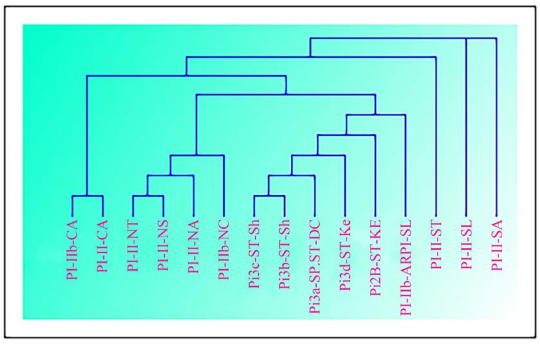

Figure 9. The successive clustering dendrogram drawn to the upright with the free-style italic and colored labels using the real data TreeView.tre.

4. Conclusions and Discussion

One of the requirements for data science at the big data era is capable of showcasing the computing results of both static and dynamic data but most of the peer programs fail to possess such features. Therefore the better algorithmic solution as above described is in great demand and it performs well to meet the needs of dynamic outcome presentation. With this algorithm, it is possible to generate the dynamic representation of hierarchical clustering results, which makes a smoothly changeable visualization of hierarchical cluster analysis. In addition, a free-style resultant vision is helpful to users, and together they are integrated as a much better view from the varying perspectives than other programs. This includes drawing to the right (forward view), to the left (mirrored view), and to the upright (upward view). The core code given also is very concise compared to the R- and Python-based programs. It is shared for drawing and has no extra code to cost when drawing orientation is switched. The algorithm can be generalized to use in any image drawing beyond tree drawing, particularly for the mode of remainder pixels rearranging in resizing and rescaling a picture. The workings behind the algorithm are analogous when extended to the otherwise drawings using the remainder pixels incremented or decremented. This makes possible the seamless pixel rearrangements to resize, rescale and restyle any imaging. The algorithm can also be applied to the dynamic data visualization from data input all the way through graphical output of the analytical outcome. The data can be of any type and category but is required to be clusterable such as unsupervised data or classifiable such as supervised data. Aside from the phylogenetic data, the phylogenomic or bioinformatics data such as DNA banding data, gene (family) clusterings, gene or protein homologs, genomic microarray data, etc can also be used for this purpose in the algorithm.

With additional algorithm built in the software ParCluster v.3.0, it is possible to save and store the dynamic representation of hierarchical clustering results wherever appropriate. The saved work can be a picture in any of the common formats such as jpeg, tiff and png. This way has a great advantage over other peer applications for a picture saved that has the highest view or publication quality, provided that the right size, scale and style are well adjusted for a demand. This is just the work to do by the algorithm discussed, which is able to make a perfect adjustment for whatever an image quality required by resizing, rescaling and restyling a dendrogram. The information about image size and scale (width x height) can be displayed according to adjustment at the upper-left corner of the window. There is no need for such a quality picture saved to be digitally enlarged or shrunken for a required quality, say, for publication, since it has the preset view quality of the right size, scale and style.

For usage of the dendrogram drawing program built in the software ParCluster v.3.0, the software is available upon request at rli@alumni.lsu.edu. It is the best window to showcase the algorithm with resizable, rescalable and free-style graphical visualization of hierarchical cluster analysis.

Conflicts of Interest

The author declares no conflicts of interest regarding the publication of this paper.

References

[1] Falcaro, M. and Pickles, A. (2010) Riskplot: A Graphical Aid to Investigate the Effect of Multiple Categorical Risk Factors. *STATA Journal*, **10**, 61-68. https://doi.org/10.1177/1536867X1001000107

[2] Jobb, G., von Haeseler, A. and Strimmer, K. (2004) TREEFINDER: A Powerful Graphical Analysis Environment for Molecular Phylogenetics. *BMC Evolutionary Biology*, **4**, 18. https://doi.org/10.1186/1471-2148-4-18

[3] Kumar, S., Tamura, K. and Nei, M. (1994) MEGA: Molecular Evolutionary Genetics Analysis Software for Microcomputers. *Bioinformatics*, **10**, 189-191. https://doi.org/10.1093/bioinformatics/10.2.189

[4] Kumar, S., Nei, M., Dudley, J.T. and Tamura, K. (2008) MEGA: A Biologist-Centric Software for Evolutionary Analysis of DNA and Protein Sequences. *Briefings in Bioinformatics*, **9**, 299-306. https://doi.org/10.1093/bib/bbn017

[5] Olsen, G.J., Matsuda, H., Hagstrom, R. and Overbeek, R. (1994) fastDNAml: A Tool for Construction of Phylogenetic Trees of DNA Sequences Using Maximum Likelihood. *Bioinformatics*, **10**, 41-48. https://doi.org/10.1093/bioinformatics/10.1.41

[6] Page, R.D. (1996) TreeView: An Application to Display Phylogenetic Trees on Personal Computers. *Bioinformatics*, **12**, 357-358. https://doi.org/10.1093/bioinformatics/12.4.357

[7] Page, R.D. (2003) Visualizing Phylogenetic Trees Using TreeView. *Current protocols in Human Genetics*, **10**, 6.2.1-6.2.15.

[8] Saitou, N. and Nei, M. (1987) The Neighbor-Joining Method: A New Method for Reconstructing Phylogenetic Trees. *Molecular Biology and Evolution*, **4**, 406-425.

[9] Seo, J. and Shneiderman, B. (2003) Understanding Hierarchical Clustering Results by Interactive Exploration of Dendrograms: A Case Study with Genomic Microarray Data. *IEEE Computer*, **35**, 1-15.

[10] Zmasek, C.M. and Eddy, S.R. (2001) ATV: Display and Manipulation of Annotated Phylogenetic Trees. *Bioinformatics*, **17**, 383-384. https://doi.org/10.1093/bioinformatics/17.4.383

[11] Li, R., Li, X. and Wang, G. (2015) Improved and Novel Cluster Analysis for Bioinformatics, Computational Biology and All Other Data. *Proceedings of the* 16*th International Conference on Bioinformatics & Computational Biology*, Las Vegas, 131-139.

[12] Li, W. (1981) Simple Method for Constructing Phylogenetic Trees from Distance Matrices. *Proceedings of the National Academy of Sciences of the United States of America*, **78**, 1085-1089. https://doi.org/10.1073/pnas.78.2.1085