*GigaScience*, 7, 2018, 1–9

doi: 10.1093/gigascience/giy083
Advance Access Publication Date: 11 July 2018
Research

RESEARCH

# Clustering trees: a visualization for evaluating clusterings at multiple resolutions

Luke Zappia [1,2] and Alicia Oshlack [1,2,*]

[1]Bioinformatics, Murdoch Children's Research Institute, Flemington Road, Parkville, Victoria 3052, Australia and [2]School of Biosciences, Faculty of Science, The University of Melbourne, Parkville, Victoria 3052, Australia

[*]**Correspondence address.** Alicia Oshlack, E-mail: alicia.oshlack@mcri.edu.au http://orcid.org/0000-0001-9788-5690 Address: Alicia Oshlack, Murdoch Children's Research Institute, Flemington Road, Parkville, Victoria 3052, Australia

## Abstract

Clustering techniques are widely used in the analysis of large datasets to group together samples with similar properties. For example, clustering is often used in the field of single-cell RNA-sequencing in order to identify different cell types present in a tissue sample. There are many algorithms for performing clustering, and the results can vary substantially. In particular, the number of groups present in a dataset is often unknown, and the number of clusters identified by an algorithm can change based on the parameters used. To explore and examine the impact of varying clustering resolution, we present clustering trees. This visualization shows the relationships between clusters at multiple resolutions, allowing researchers to see how samples move as the number of clusters increases. In addition, meta-information can be overlaid on the tree to inform the choice of resolution and guide in identification of clusters. We illustrate the features of clustering trees using a series of simulations as well as two real examples, the classical iris dataset and a complex single-cell RNA-sequencing dataset. Clustering trees can be produced using the clustree R package, available from CRAN and developed on GitHub.

*Keywords:* clustering; visualization; scRNA-seq

## Introduction

Clustering analysis is commonly used to group similar samples across a diverse range of applications. Typically, the goal of clustering is to form groups of samples that are more similar to each other than to samples in other groups. While fuzzy or soft clustering approaches assign each sample to every cluster with some probability, and hierarchical clustering forms a tree of samples, most methods form hard clusters where each sample is assigned to a single group. This goal can be achieved in a variety of ways, such as by considering the distances between samples (e.g., *k*-means [1–3], PAM [4]), areas of density across the dataset (e.g., DBSCAN [5]), or relationships to statistical distributions [6].

In many cases, the number of groups that should be present in a dataset is not known in advance, and deciding the correct number of clusters to use is a significant challenge. For some algorithms, such as *k*-means clustering, the number of clusters must be explicitly provided. Other methods have parameters that, directly or indirectly, control the clustering resolution and therefore the number of clusters produced. While there are methods and statistics (such as the elbow method [7] and silhouette plots [8]) designed to help analysts decide which clustering resolution to use, they typically produce a single score that only considers a single set of samples or clusters at a time.

An alternative approach would be to consider clusterings at multiple resolutions and examine how samples change groupings as the number of clusters increases. This has led to a range of cluster stability measures [9], many of which rely on clustering of perturbed or subsampled datasets. For example, the model explorer algorithm subsamples a dataset multiple times, clusters each subsampled dataset at various resolutions, and then calculates a similarity between clusterings at the same resolution to give a distribution of similarities that can inform the

**Received:** 7 March 2018; **Revised:** 21 May 2018; **Accepted:** 27 June 2018

choice of resolution [10]. One cluster stability measure that is not based on perturbations is that contained in the SC3 package for clustering single-cell RNA-sequencing (scRNA-seq) data [11]. Starting with a set of cluster labels at different resolutions, each cluster is scored, with clusters awarded increased stability if they share the same samples as a cluster at another resolution but penalized for being at a higher resolution.

A similar simple approach is taken by the clustering tree visualization we present here, without calculating scores: (i) a dataset is clustered using any hard clustering algorithm at multiple resolutions, producing sets of cluster nodes; (ii) the overlap between clusters at adjacent resolutions is used to build edges; and (iii) the resulting graph is presented as a tree. This tree can be used to examine how clusters are related to each other—which clusters are distinct and which are unstable. In the following sections, we describe how we construct such a tree and present examples of trees built from a classic clustering dataset and a complex scRNA-seq dataset. The figures shown here can be produced in R using our publicly available clustree package. Although clustering trees cannot directly provide a clustering resolution to use, they can be a useful tool for exploring and visualizing the range of possible choices.

## Building a Clustering Tree

To build a clustering tree, we start with a set of clusterings and allocate samples to groups at several different resolutions. These could be produced using any hard-clustering algorithm that allows control of the number of clusters in some way. For example, this could be a set of samples clustered using $k$-means with $k = 1, 2, 3$ as shown in Fig. 1. We sort these clusterings so that they are ordered by increasing resolution ($k$), then consider pairs of adjacent clusterings. Each cluster $c_{k,\,i}$ (where $i = 1, \ldots, n$ and $n$ is the number of clusters at resolution $k$) is compared with each cluster $c_{k+1,\,j}$ (where $j = 1, \ldots, m$ and $m$ is the number of clusters at resolution $k + 1$). The overlap between the two clusters is computed as the number of samples that are assigned to both $c_{k,\,i}$ and $c_{k+1,\,j}$. Next, we build a graph where each node is a cluster and each edge is an overlap between two clusters. While we refer to this graph as a tree for simplicity, it can more correctly be described as a polytree, a special case of a directed acyclic graph where the underlying undirected graph is a tree [12].

Many of the edges will be empty, e.g., in Fig. 1 no samples in cluster A at $k = 2$ end up in cluster B at $k = 3$. In some datasets there may also be edges that contain few samples. These edges are not informative and result in a cluttered tree. An obvious solution for removing uninformative, low-count edges is to filter them using a threshold on the number of samples they represent. However, in this case, the count of samples is not the correct statistic to use because it favors edges at lower resolutions and those connecting larger clusters. Instead, we define the in-proportion metric as the ratio between the number of samples on the edge and the number of samples in the cluster it goes toward. This metric shows the importance of the edge to the higher-resolution cluster independently of the cluster size. We can then apply a threshold to the in-proportion in order to remove less-informative edges.

The final graph can then be visualized. In theory, any graph layout algorithm could be used. However, for the clustree package, we have made use of the two algorithms specifically designed for tree structures available in the igraph package [13]. These are the Reingold-Tilford tree layout, which places parent nodes above their children [14], and the Sugiyama layout, which places nodes of a directed acyclic graph in layers while minimizing the number of crossing edges [15]. Both of these algorithms can produce attractive layouts; as such, we have not found the need to design a specific layout algorithm for clustering trees. By default, the clustree package uses only a subset of edges when constructing a layout, specifically the highest in-proportion edges for each node. We have found that this often leads to more interpretable visualizations; however, users can choose to use all edges if desired.

Regardless of the layout used, the final visualization places the cluster nodes in a series of layers where each layer is a different clustering resolution and edges show the transition of samples through those resolutions. Edges are colored according to the number of samples they represent, and the in-proportion metric is used to control the edge transparency, highlighting more important edges. By default, the node size is adjusted according to the number of samples in the cluster, and their color indicates the clustering resolution. The clustree package also includes options for controlling the aesthetics of nodes based on the attributes of samples in the clusters they represent, as shown in the following examples.

While a clustering tree is conceptually similar to the tree produced through hierarchical clustering, there are some important differences. The most obvious are that a hierarchical clustering tree is the result of a particular clustering algorithm and shows the relationships between individual samples, while the clustering trees described here are independent of clustering method and show relationships between clusters. The branches of a hierarchical tree show how the clustering algorithm has merged samples. In contrast, edges in a clustering tree show how samples move between clusters as the resolution changes and nodes may have multiple parents. While it is possible to overlay information about samples on a hierarchical tree, this is not commonly done but is a key feature of the clustree package and how clustering trees could be used in practice.

## A Demonstration Using Simulations

To demonstrate what a clustering tree can look like in different situations and how it behaves as a dataset is overclustered, we present some illustrative examples using simple simulations (see Methods). We present five scenarios: random uniform noise (simulation A), a single cluster (simulation B), two clusters (simulation C), three clusters (simulation D), and four clusters (simulation E). Each cluster consists of 1,000 samples (points) generated from a 100-dimensional normal distribution, and each synthetic dataset has been clustered using $k$-means clustering with $k = 1, \ldots, 8$. We then use the clustree package to produce clustering trees for each dataset (Fig. 2).

Looking at the first two examples (uniform noise [Fig. 2A] and a single cluster [Fig. 2B]), we can clearly see how a clustering tree behaves when a clustering algorithm returns more clusters than are truly present in a dataset. New clusters begin to form from multiple existing clusters, and many samples switch between branches of the tree, resulting in low in-proportion edges. Unstable clusters may also appear and then disappear as the resolution increases, as seen in Fig. 2E. As we add more structure to the datasets, the clustering trees begin to form clear branches and low in-proportion edges tend to be confined to sections of the tree. By looking at which clusters are stable and where low in-proportion edges arise, we can infer which areas of the tree

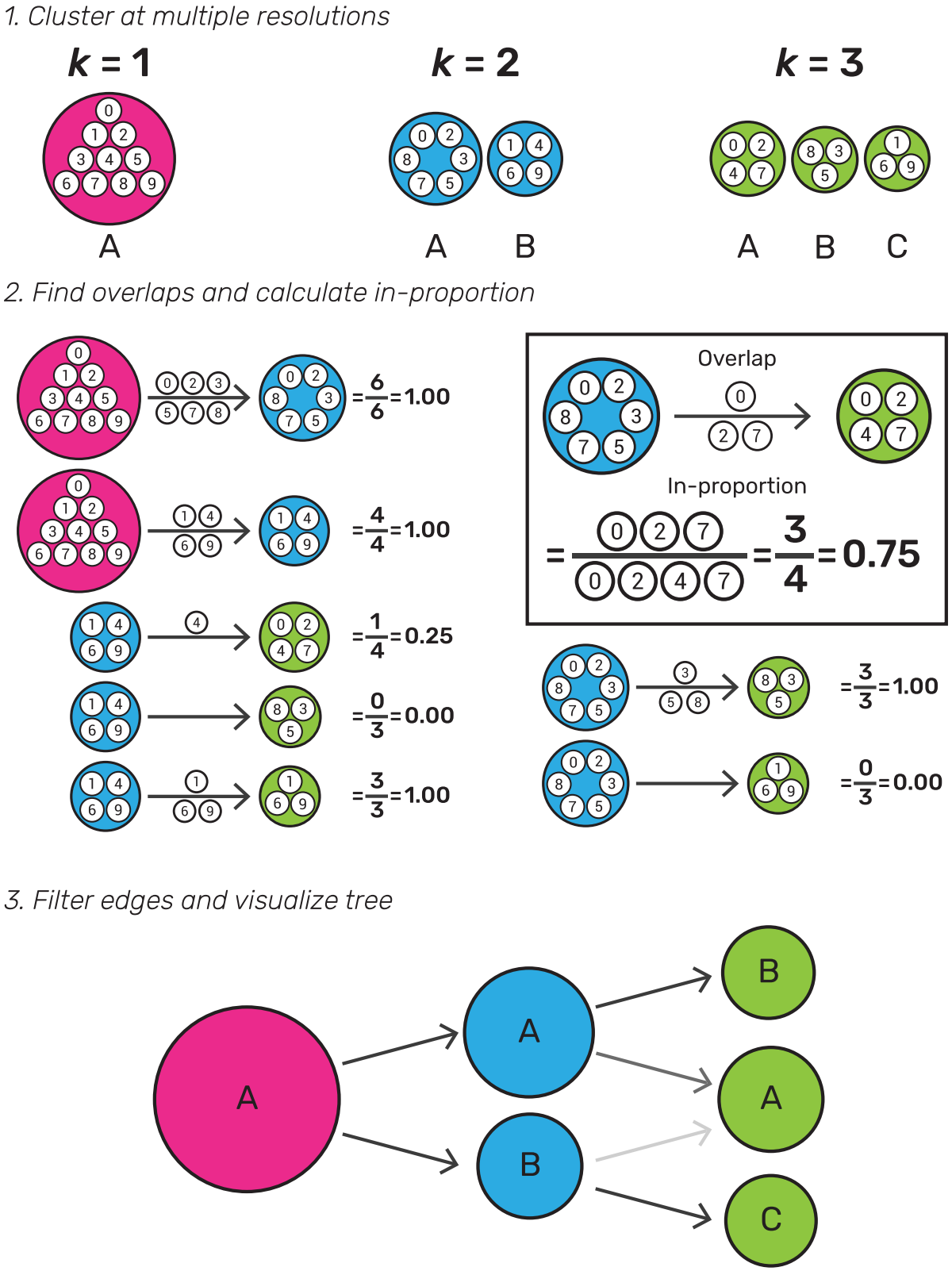


**Figure 1:** Illustration of the steps required to build a clustering tree. First, a dataset must be clustered at different resolutions. The overlap in samples between clusters at adjacent resolutions is computed and used to calculate the in-proportion for each edge. Finally, the edges are filtered and the graph visualized as a tree.

are likely to be the result of true clusters and which are caused by overclustering.

The second clustering tree for each dataset shows nodes colored according to the SC3 stability index for each cluster. As we

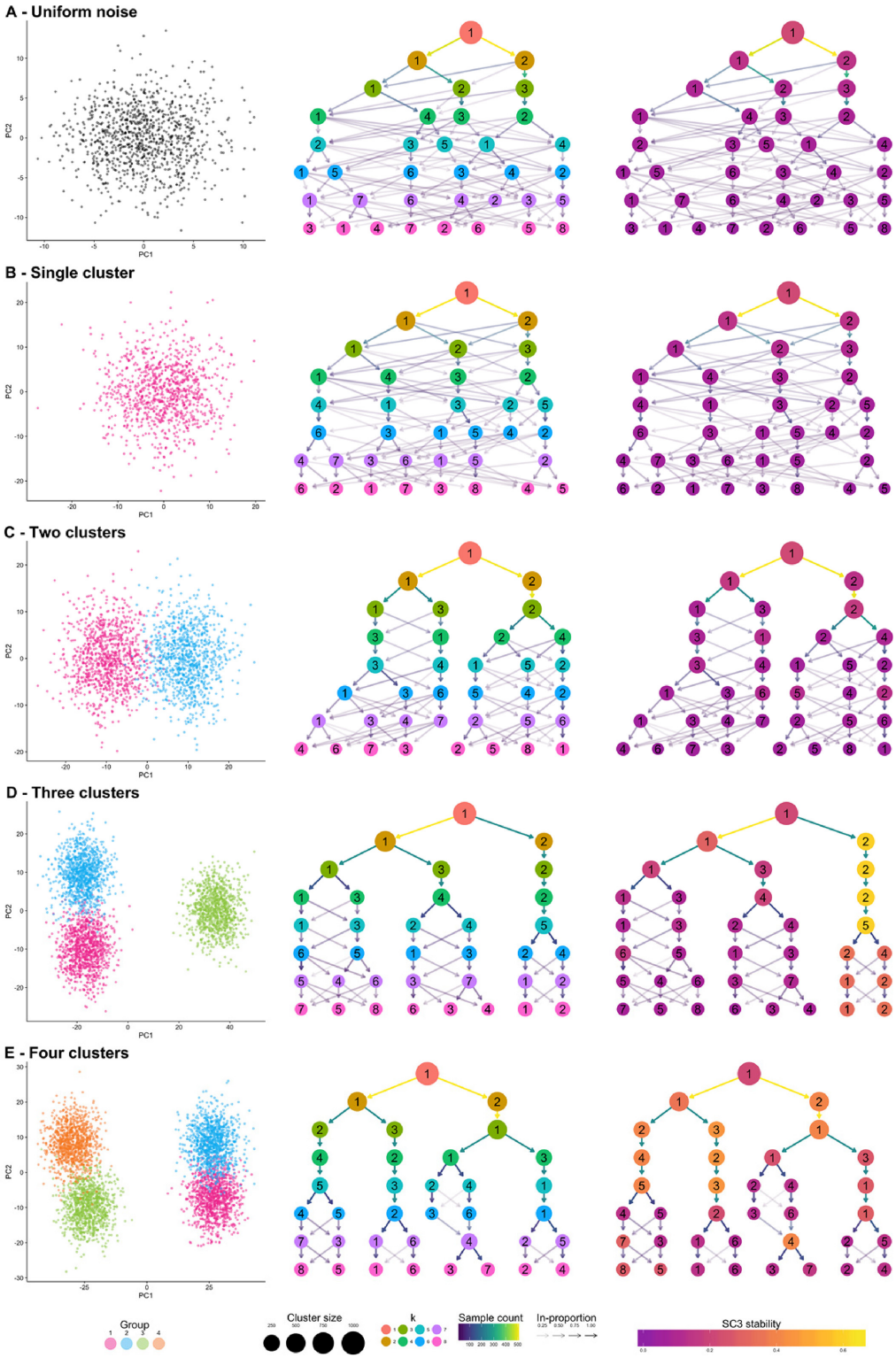


**Figure 2:** Five synthetic datasets used to demonstrate clustering trees. For each dataset, a scatter plot of the first two principal components, a default clustering tree, and clustering tree with nodes colored by the SC3 stability index from purple (lowest) to yellow (highest) are shown. The five datasets contain: **(A)** random uniform noise, **(B)** a single cluster, **(C)** two clusters, **(D)** three clusters, and **(E)** four clusters.

would expect, no cluster receives a high stability score in the first two examples. However, while we clearly see two branches in the clustering tree for the two-cluster example (simulation C), this is not reflected in the SC3 scores. No cluster receives a high stability score, most likely due to the large number of samples moving between clusters as the resolution increases. As there are more true clusters in the simulated datasets, the SC3 stability scores become more predictive of the correct resolution to

use. However, it is important to look at the stability scores of all clusters at a particular resolution as taking the highest individual cluster stability score could lead to the incorrect resolution being used, as can be seen in the four-cluster example (simulation E). These examples show how clustering trees can be used to display existing clustering metrics in a way that can help to inform parameter choices.

## A Simple Example

To further illustrate how a clustering tree is built, we will work through an example using the classic iris dataset [16, 17]. This dataset contains measurements of the sepal length, sepal width, petal length, and petal width from 150 iris flowers, 50 from each of three species: *Iris setosa*, *Iris versicolor*, and *Iris virginica*. The iris dataset is commonly used as an example for both clustering and classification problems with the *I. setosa* samples being significantly different from, and linearly separable from, the other samples. We have clustered this dataset using $k$-means clustering with $k = 1, \ldots, 5$ and produced the clustering tree shown in Fig. 3A.

We see that one branch of the tree is clearly distinct (presumably representing *I. setosa*), remaining unchanged regardless of the number of clusters. On the other side, we see that the cluster at $k = 2$ cleanly splits into two clusters (presumably *I. versicolor* and *I. virginica*) at $k = 3$. However, as we move to $k = 4$ and $k = 5$, we see clusters being formed from multiple branches with more low in-proportion edges. As we have seen in the simulated examples, this kind of pattern can indicate that the data have become overclustered and we have begun to introduce artificial groupings.

We can check our assumption that the distinct branch represents the *I. setosa* samples and that the other two clusters at $k = 3$ are *I. versicolor* and *I. virginica* by overlaying some known information about the samples. In Fig. 3B we have colored the nodes by the mean petal length of the samples they contain. We can now see that clusters in the distinct branch have the shortest petals, with cluster 1 at $k = 3$ having an intermediate length and cluster 3 having the longest petals. This feature is known to separate the samples into the expected species, with *I. setosa* having the shortest petals on average, *I. versicolor* an intermediate length, and *I. virginica* the longest.

Although this is a very simple example, it highlights some of the benefits of viewing a clustering tree. We get some indication of the correct clustering resolution by examining the edges, and we can overlay known information to assess the quality of the clustering. For example, if we observed that all clusters had the same mean petal length, it would suggest that the clustering has not been successful as we know this is an important feature that separates the species. We could potentially learn more by looking at which samples follow low-proportion edges or by overlaying a series of features to try and understand what causes particular clusters to split.

## Clustering Trees for scRNA-seq Data

One field that has begun to make heavy use of clustering techniques is the analysis of scRNA-seq data. scRNA-sequencing is a recently developed technology that can measure how genes are expressed in thousands to millions of individual cells [18]. This technology has been rapidly adopted in fields such as developmental biology and immunology where it is valuable to have information from single cells rather than measurements that are averaged across the many different cells in a sample using older RNA-seq technologies. A key use for scRNA-seq is to discover and interrogate the different cell types present in a sample of a complex tissue. In this situation, clustering is typically used to group similar cells based on their gene expression profiles. Differences in gene expression between groups can then be used to infer the identity or function of those cells [19]. The number of cell types (clusters) in an scRNA-seq dataset can vary depending on factors such as the tissue being studied, its developmental or environmental state, and the number of cells captured. Often, the number of cell types is not known before the data are generated, and some samples can contain dozens of clusters. Therefore, deciding which clustering resolution to use is an important consideration in this application.

As an example of how clustering trees can be used in the scRNA-seq context, we consider a commonly used peripheral blood mononuclear cell (PBMC) dataset. This dataset was originally produced by 10x Genomics and contains 2,700 peripheral blood mononuclear cells, representing a range of well-studied immune cell types [20]. We analyzed this dataset using the Seurat package [21], a commonly used toolkit for scRNA-seq analysis, following the instructions in their tutorial with the exception of varying the clustering resolution parameter from zero to 5 (see Methods). Seurat uses a graph-based clustering algorithm, and the resolution parameter controls the partitioning of this graph, with higher values resulting in more clusters. The clustering trees produced from this analysis are shown in Fig. 4.

The clustering tree covering resolutions zero to 1 in steps of 0.1 (Fig. 4A) shows that four main branches form at a resolution of just 0.1. One of these branches, starting with cluster 3 at resolution 0.1, remains unchanged, while the branch starting with cluster 2 splits only once at a resolution of 0.4. Most of the branching occurs in the branch starting with cluster 1, which consistently has subbranches split off to form new clusters as the resolution increases. There are two regions of stability in this tree—at resolution 0.4–0.5 and resolution 0.7–1.0 where the branch starting at cluster 0 splits in two.

Fig. 4B shows a clustering tree with a greater range of resolutions, from zero to 5 in steps of 0.5. By looking across this range, we can see what happens when the algorithm is forced to produce more clusters than are likely to be truly present in this dataset. As overclustering occurs, we begin to see more low in-proportion edges and new clusters forming from multiple parent clusters. This suggests that those areas of the tree are unstable and that the new clusters being formed are unlikely to represent true groups in the dataset.

Known marker genes are commonly used to identify the cell types that specific clusters correspond to. Overlaying gene expression information onto a clustering tree provides an alternative view that can help to indicate when clusters containing pure cell populations are formed. Figure 5 shows the PBMC clustering tree in Fig. 4A overlaid with the expression of some known marker genes.

By adding this extra information, we can quickly identify some of the cell types. CD19 (Fig. 5A) is a marker of B cells and is clearly expressed in the most distinct branch of the tree. CD14 (Fig. 5B) is a marker of a type of monocyte, which becomes more expressed as we follow one of the central branches, allowing us to see which resolution identifies a pure population of these cells. CD3D (Fig. 5C) is a general marker of T cells and is expressed in two separate branches, one that splits into low and high expression of CCR7 (Fig. 5D), separating memory and naive CD4 T cells. By adding expression of known genes to a clustering tree, we can see if more populations can be identified as the

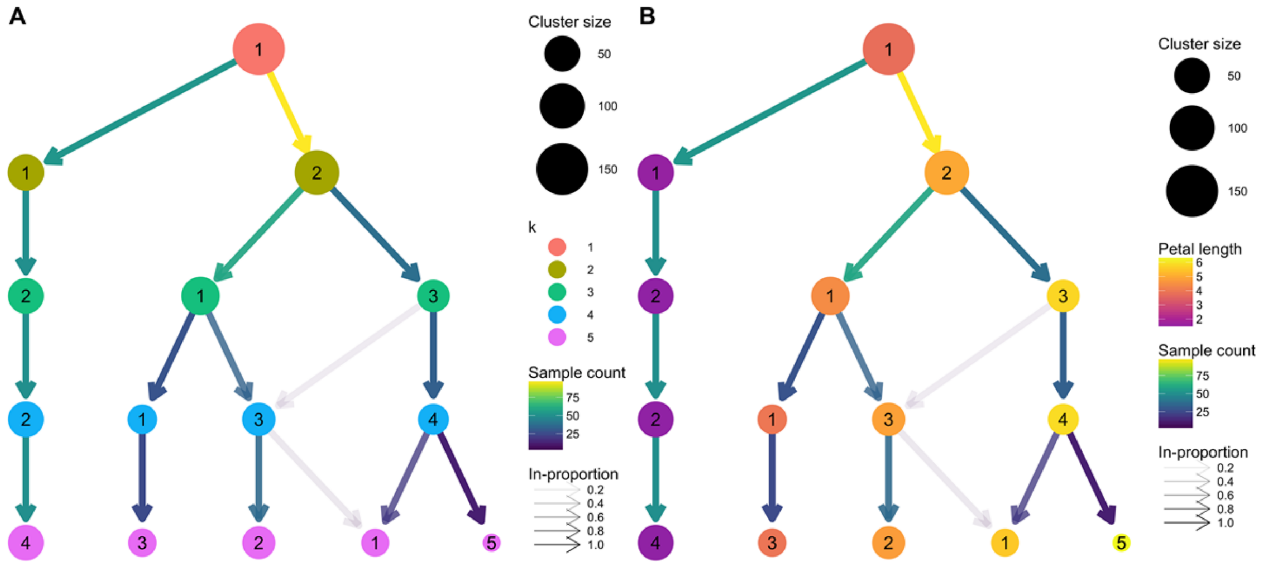


**Figure 3:** Clustering trees based on *k*-means clustering of the iris dataset. **(A)** Nodes are colored according to the value of *k* and sized according to the number of samples they represent. Edges are colored according to the number of samples (from blue representing few to yellow representing many). The transparency is adjusted according to the in-proportion, with stronger lines showing edges that are more important to the higher-resolution cluster. Cluster labels are randomly assigned by the *k*-means algorithm. **(B)** The same tree with the node coloring changed to show the mean petal length of the samples in each cluster.

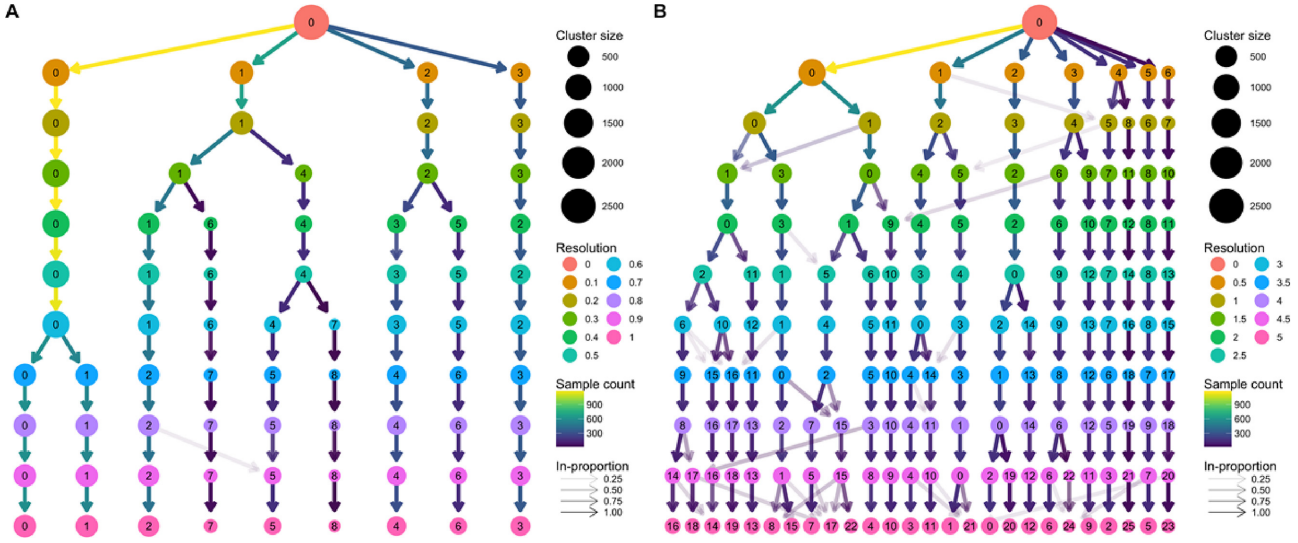


**Figure 4:** Two clustering trees of a dataset of 2,700 PBMCs. **(A)** Results from clustering using Seurat with resolution parameters from zero to 1. At a resolution of 0.1, we see the formation of four main branches, one of which continues to split up to a resolution of 0.4, after which there are only minor changes. **(B)** Resolutions from zero to 5. At the highest resolutions, we begin to see many low in-proportion edges, indicating cluster instability. Seurat labels clusters according to their size, with cluster 0 being the largest.

clustering resolution is increased and if clusters are consistent with known biology. For most of the Seurat tutorial, a resolution of 0.6 is used, but the authors note that by moving to a resolution of 0.8, a split can be achieved between memory and naive CD4 T cells. This is a split that could be anticipated by looking at the clustering tree with the addition of prior information.

## Discussion

Clustering similar samples into groups is a useful technique in many fields, but often analysts are faced with the tricky problem of deciding which clustering resolution to use. Traditional approaches to this problem typically consider a single cluster or sample at a time and may rely on prior knowledge of sample labels. Here, we present clustering trees, an alternative visualization that shows the relationships between clusterings at multiple resolutions. While clustering trees cannot directly suggest which clustering resolution to use, they can be a useful tool for helping to make that decision, particularly when combined with other metrics or domain knowledge.

Clustering trees display how clusters are divided as resolution increases, which clusters are clearly separate and distinct, which are related to each other, and how samples change groups as more clusters are produced. Although clustering trees can appear similar to the trees produced from hierarchical clustering, there are several important differences. Hierarchical clus-

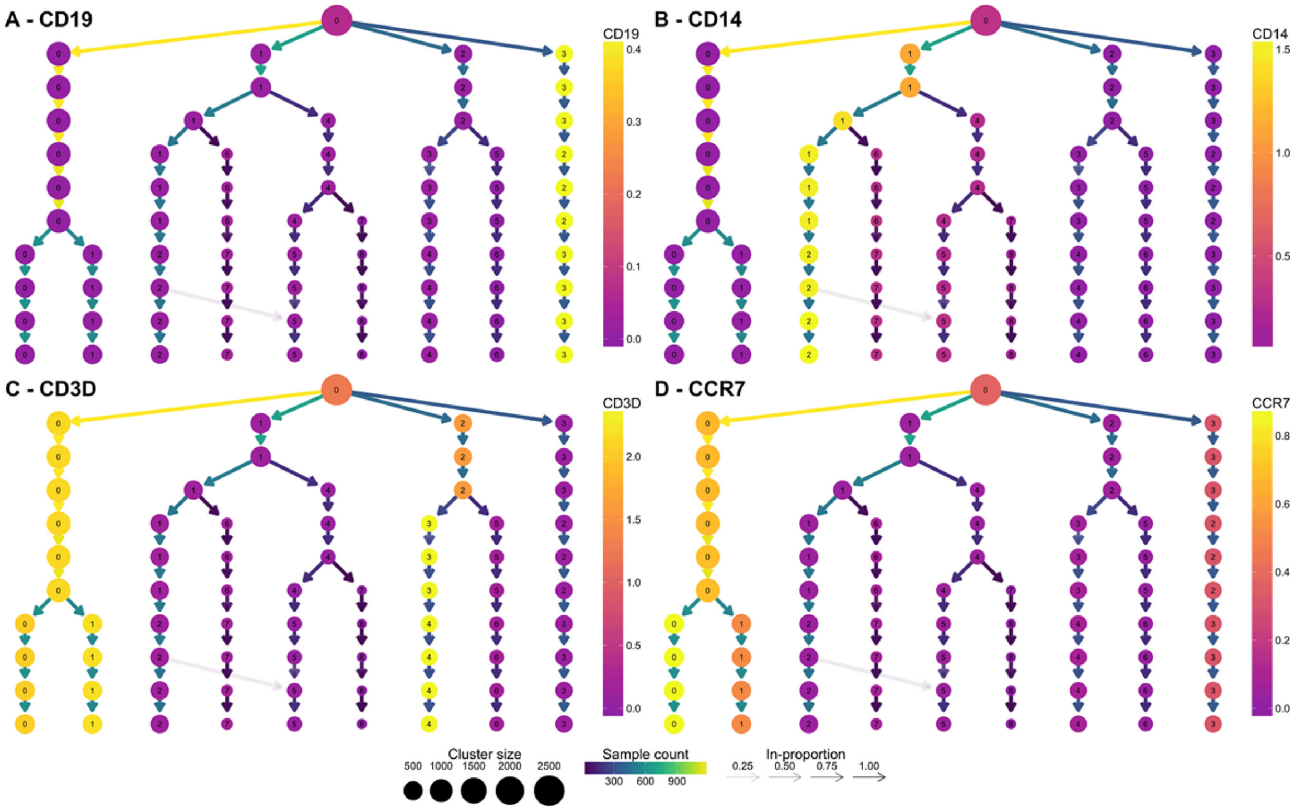


**Figure 5:** Clustering trees of the PBMC dataset colored according to the expression of known markers. The node colors indicate the average of the $\log_2$ gene counts of samples in each cluster. CD19 **(A)** identifies B cells, CD14 **(B)** shows a population of monocytes, CD3D **(C)** is a marker of T cells, and CCR7 **(D)** shows the split between memory and naive CD4 T cells.

tering considers the relationships between individual samples and does not provide an obvious way to form groups. In contrast, clustering trees are independent of any particular clustering method and show the relationships between clusters, rather than samples, at different resolutions, any of which could be used for further analysis.

To illustrate the uses of clustering trees, we presented a series of simulations and two examples of real analyses, one using the classic iris dataset and a second based on a complex scRNA-seq dataset. Both examples demonstrate how a clustering tree can help inform the decision of which resolution to use and how overlaying extra information can help to validate those clusters. This is of particular use to scRNA-seq analysis as these datasets are often large, noisy, and contain an unknown number of cell types or clusters.

Even when determining the number of clusters is not a problem, clustering trees can be a valuable tool. They provide a compact, information-dense visualization that can display summarized information across a range of clusters. By modifying the appearance of cluster nodes based on attributes of the samples they represent, clusterings can be evaluated and identities of clusters established. Clustering trees potentially have applications in many fields and, in the future, could be adapted to be more flexible, such as by accommodating fuzzy clusterings. There may also be uses for more general clustering graphs to combine results from multiple sets of parameters or clustering methods.

## Methods

### clustree

The clustree software package (v0.2.0) is built for the R statistical programming language (v3.5.0). It relies on the ggraph package (v1.0.1) [22], which is built on the ggplot2 (v2.2.1) [23] and tidygraph (v1.1.0) [24] packages. Clustering trees are displayed using the Reingold-Tilford tree layout or the Sugiyama layout; both are available as part of the igraph package (v1.2.1).

Figure panels shown here were produced using the cowplot package (v0.9.2) [25].

### Simulations

Simulated datasets were constructed by generating points from statistical distributions. The first simulation (simulation A) consists of 1,000 points randomly generated from a 100-dimensional space using a uniform distribution between zero and 10. Simulation B consists of a single normally distributed cluster of 1,000 points in 100 dimensions. The center of this cluster was chosen from a normal distribution with mean zero and standard deviation 10. Points were then generated around this center from a normal distribution with mean equal to the center point and a standard deviation of 5. The remaining three simulations were produced by adding additional clusters. In order to have a known relationship between clusters, the center for the new clusters was created by manipulating the centers of existing clusters. For cluster 2, a random 100-dimensional vector was generated from a normal distribution with mean zero and standard deviation 2 and added to the center for cluster 1. Center 3 was the average of center 1 and center 2 plus a random vector from a

normal distribution with mean zero and standard deviation 5. To ensure a similar relationship between clusters 3 and 4 as between clusters 1 and 2, center 4 was produced by adding half the vector used to produce center 2 to center 3 plus another vector from a normal distribution with mean zero and standard deviation 2. Points for each cluster were generated in the same way as for cluster 1. Simulation C consists of the points in clusters 1 and 2; simulation D consists of clusters 1, 2, and 3; and simulation E consists of clusters 1, 2, 3, and 4. Each simulated dataset was clustered using the "kmeans" function in the stats package with values of $k$ from 1 to 8, a maximum of 100 iterations, and 10 random starting positions. The clustering tree visualizations were produced using the clustree package with the tree layout. The simulated datasets and the code used to produce them are available from the repository for this article [26].

### Iris dataset

The iris dataset is available as part of R. We clustered this dataset using the "kmeans" function in the stats package with values of $k$ from 1 to 5. Each value of $k$ was clustered with a maximum of 100 iterations and with 10 random starting positions. The clustree package was used to visualize the results using the Sugiyama layout. The clustered iris dataset is available as part of the clustree package.

### PBMC dataset

The PBMC dataset was downloaded from the Seurat tutorial page [27], and this tutorial was followed for most of the analysis using Seurat version 2.3.1. Briefly, cells were filtered based on the number of genes they express and the percentage of counts assigned to mitochondrial genes. The data were then log-normalized and 1,838 variable genes identified. Potential confounding variables (number of unique molecular identifiers and percentage mitochondrial expression) were regressed from the dataset before performing principal component analysis on the identified variable genes. The first 10 principal components were then used to build a graph that was partitioned into clusters using Louvain modularity optimization [28] with resolution parameters in the range zero to 5, in steps of 0.1 between zero and 1, and then in steps of 0.5. Clustree was then used to visualize the results using the tree layout.

## Availability of source code and requirements

Project name: clustree.
Project home page: https://github.com/lazappi/clustree.
Operating systems(s): Linux, MacOS, Windows
Programming language: R ($>= 3.4$)
Other requirements: None
License: GPL-3
Any restrictions to use by non-academics: None
RRID:SCR_016293

## Availability of supporting data

The clustree package is available from CRAN [29] and is being developed on GitHub [30]. The code and datasets used for the analysis presented here are also available from GitHub [26]. The clustered iris dataset is included as part of clustree, and the PBMC dataset can be downloaded from the Seurat tutorial page [27] or the paper GitHub repository. Snapshots of the code are available in the *GigaScience* repository, GigaDB [31].

## Abbreviations

PBMC: peripheral blood mononuclear cell; scRNA-seq: single-cell RNA-sequencing.

## Competing interests

The authors declare that they have no competing interests.

## Funding

L.Z. is supported by an Australian Government Research Training Program scholarship. A.O. is supported through a National Health and Medical Research Council Career Development fellowship (APP1126157). The Murdoch Children's Research Institute is supported by the Victorian Government's Operational Infrastructure Support Program.

## Author contributions

L.Z. designed the clustering tree algorithm, wrote the clustree software package, and drafted the manuscript. A.O. supervised the project and commented on the manuscript.

## Acknowledgements

Thank you to Marek Cmero for providing comments on a draft of the manuscript and the reviewers for their comments and suggestions.

## References

1. Forgy WE. Cluster analysis of multivariate data: efficiency versus interpretability of classifications. Biometrics 1965;**21**:768–9.
2. Macqueen J. Some methods for classification and analysis of multivariate observations. In 5th Berkeley Symposium on Mathematical Statistics and Probability, 1967.
3. Lloyd S. Least squares quantization in PCM. IEEE Trans Inf Theory 1982;**28**:129–37.
4. Kaufman L, Rousseeuw PJ. Partitioning Around Medoids (Program PAM). Finding Groups in Data, New Jersey, USA. John Wiley & Sons, Inc. 1990. pp. 68–125.
5. Ester M, Kriegel H-P, Sander J, et al. A density-based algorithm for discovering clusters in large spatial databases with noise. Proceedings of the Second International Conference on Knowledge Discovery and Data Mining. Portland, Oregon: AAAI Press; 1996. pp. 226–31. Available from:
6. Fraley C, Raftery AE. Model-based clustering, discriminant analysis, and density estimation. J Am Stat Assoc 2002;**97**:611–31.
7. Thorndike RL. Who belongs in the family? Psychometrika 1953;**18**:267–76.
8. Rousseeuw PJ. Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. J Comput Appl Math 1987;**20**:53–65.
9. Luxburg U von. Clustering stability: an overview. Foundations and Trends in Machine Learning 2010;**2**:235–74.
10. Ben-Hur A, Elisseeff A, Guyon I. A stability based method for discovering structure in clustered data. Pac Symp Biocomput 2002, **7**;6–17.
11. Kiselev VY, Kirschner K, Schaub MT, et al. SC3: consensus clustering of single-cell RNA-seq data. Nat Methods 2017;**14**:483–6.

12. Rebane G, Pearl J. The Recovery of Causal Poly-Trees from Statistical Data. 2013; Available from: http://arxiv.org/abs/1304.2736, Accessed May 16, 2018.
13. Csardi G, Nepusz T. The igraph software package for complex network research. Inter Journal Complex Systems 2006;**1695**:1–9.
14. Reingold EM, Tilford JS. Tidier drawings of trees. IEEE Trans Software Eng 1981;**SE-7**:223–8.
15. Sugiyama K, Tagawa S, Toda M. Methods for visual understanding of hierarchical system structures. IEEE Trans Syst Man Cybern 1981;**11**:109–25.
16. Anderson E. The irises of the Gaspe Peninsula. Bulletin of the American Iris Society 1935;**59**:2–5.
17. Fisher RA. The use of multiple measurements in taxonomic problems. Ann Eugen 1936;**7**:179–88.
18. Tang F, Barbacioru C, Wang Y, et al. mRNA-seq whole-transcriptome analysis of a single cell. Nat Methods 2009;**6**:377–82.
19. Stegle O, Teichmann SA, Marioni JC. Computational and analytical challenges in single-cell transcriptomics. Nat Rev Genet 2015;**16**:133–45.
20. Zheng GXY, Terry JM, Belgrader P, et al. Massively parallel digital transcriptional profiling of single cells. Nat Commun 2017;**8**:14049.
21. Satija R, Farrell JA, Gennert D, et al. Spatial reconstruction of single-cell gene expression data. Nat Biotechnol 2015;**33**:495–502.
22. Pedersen TL. ggraph: An Implementation of Grammar of Graphics for Graphs and Networks. 2018. Available from: https://CRAN.R-project.org/package=ggraph, Accessed 21 May, 2018
23. Wickham H. ggplot2: Elegant Graphics for Data Analysis. New York: Springer; 2010.
24. Pedersen TL. tidygraph: A Tidy API for Graph Manipulation. 2018. Available from: https://CRAN.R-project.org/package=tidygraph, Accessed May 21, 2018
25. Wilke CO. cowplot: Streamlined Plot Theme and Plot Annotations for 'ggplot2.' 2018. Available from: https://CRAN.R-project.org/package=cowplot, Accessed May 21, 2018
26. Zappia L, Oshlack A. clustree-paper GitHub repository, 2018. Available from: https://github.com/Oshlack/clustree-paper, Accessed May 21, 2018.
27. Satija Lab. Seurat PBMC3K Tutorial. ; 2018. Available from: https://satijalab.org/seurat/pbmc3k_tutorial.html, Accessed May 21, 2018
28. Blondel VD, Guillaume J-L, Lambiotte R, et al. Fast unfolding of communities in large networks. J Stat Mech. IOP Publishing; 2008;**2008**:P10008.
29. Zappia L, Oshlack A. clustree: Visualise Clusterings at Different Resolutions. 2018. Available from: https://CRAN.R-project.org/package=clustree, Accessed May 21, 2018
30. Zappia L, Oshlack A. clustree GitHub repository. ; 2018. Available from: https://github.com/lazappi/clustree, Accessed May 21, 2018.
31. Zappia L, Oshlack A. Supporting data for "Clustering trees: a visualization for evaluating clusterings at multiple resolutions." GigaScience Database 2018. http://dx.doi.org/10.5524/100478.