Creation of Two R Shiny Applications to Illustrate and Accompany the growClusters Package

Randall Powers, Terrance Savitsky and Wendy Martinez

Office of Survey Methods Research, U.S. Bureau of Labor Statistics

Powers.Randall@bls.gov

Abstract: GrowClusters is an R package that estimates a clustering or partition structure for multivariate data. Estimation is performed under a penalized optimization derived from Bayesian non-parametric formulations. This is done either under a Dirichlet process (DP) mixing measure or a hierarchical DP (HDP) mixing measure in the limit of the global variance (to zero). The latter set-up allows for a collection of dependent, local partitions. This paper describes the growClusters algorithm, but will focus on the creation of an R Shiny application designed to visually illustrate the operation and functionality of the growClusters package. Examples of the utility and functionality of the R Shiny application will be highlighted.

Key words: clustering; R Shiny; partition structure; multivariate data

1. Introduction

Cluster analysis is the grouping of data in a way such that data records assigned to the same group (or cluster) are more similar to each other than data in other groups. Clustering is often used for dimension reduction and to perform inference. It is an iterative process that can be achieved using a number of algorithms.

The growClusters package for R is a package designed to estimate a clustering or partition structure for relatively high-dimensional multivariate data. Estimation is performed under a penalized optimization derived from Bayesian non-parametric formulations in the limit that the model noise variance of a hierarchical Dirichlet process (HDP) process model goes to 0.

Given that clustering is exploratory data analysis, creation of an interactive data visualization tool to accompany the package seemed an obvious goal and was the inspiration for this project. Building an R Shiny application that would allow growClusters package users to visualize clustering outcomes was determined to be the best solution to achieve this goal.

The project is ongoing; currently we have designed two applications to accompany the package. The first, currently called `gendata`, allows the user to create a customized input dataset to be used in the second application, called `clustering`. The remainder of this paper will describe the functionality of these two applications in detail.

2. The `gendata` Application

There may be occasional times where the growClusters package user may wish to generate synthetic data under a clustering structure to explore the performance of growClusters to utilize for other purposes, but may not have a properly

formatted dataset to use as input. In this case, it would be helpful to have an application that generates a dataset that may be used as input for the `clustering` application. For this reason, we designed the `gendata` application.

The user opens the `gendata` application (see Figure 1) to find the variable data tab. There are several input options that allow the user to customize the dataset that is produced. The user may leave the defaults in place for most of the inputs, however the user is required to specify the number of clusters and the size of the clusters in order for a dataset to be generated.

The "number of clusters" is the number of clusters the user wishes to be displayed. Each cluster receives a roughly equal number of assigned observations. The user can specify any numbers of clusters, with a minimum number of two. In conjunction with the number of clusters is the "size of the clusters". Here the user must input x comma separated values which sum to one. The number of values must equal the specified number of clusters.

Additionally, there are a number of optional user inputs. These include the "population size," which allows the user to control the number of observations n that are produced. The user may specify a population size in increments of five. The "number of dimensions" is the number of variables in the dataset that is being created. The user may specify the noise_scale, which can be any value between 0 and 1 inclusive. The noise scale is the global standard deviation, a factor that affects the elements that are generated. The user may specify whether the seed used for generating the dataset is null or non-null. If the default null option is left as is, a null seed will produce a completely random dataset, whereas a non-null seed will create a dataset where the results can be repeated.

Once the user has set the required inputs (having the total number of comma separated cluster values equal the number of clusters on the number of clusters input), they can press the "Generate Data" button, and a dataset is produced (see Figure 2). If the null setting is chosen, a new dataset is created each time this button is pressed. The dataset is captured and held in the working space (unless manually cleared) for use as input for the `clustering` application. The output is displayed on the screen.

The user can choose to click the second tab and view a scatterplot matrix of the first four variables (see Figure 3). The user can also generate a new dataset while in this tab as well, since the input remains visible and active while on this tab. As mentioned above, the newly generated dataset remains in the working space and is accessible to the clustering application.

3. The `clustering` Application

The `clustering` application is the application that actually allows the user to perform clustering analysis. The user can do this with an input dataset generated by the `gendata` application, or arguably more usefully (to them), with their own properly formatted dataset.

The `clustering` application contains four tabs. The user calls the clustering function and specifies the dataset that will be analyzed. When the user opens the application, the Plot tab is opened. The first three variables are shown as a matrix plot by default. The user may select any of the variables in the dataset from the drop down menu to plot. Once this is done, the user clicks the "Produce Matrix Plot" button, and the user defined matrix plot is created (See Figure 4). At this

point, clustering is not performed, but instead allows the user to examine the data and look for groupings.

On the second tab, the user can alter a number of inputs, including the lambda value, which governs the number of clusters which are formed; the larger the lambda value, the more clusters that are discovered. When ready, the user presses a “Run” button which allows a bar plot of the cluster counts to be displayed (See Figure 5). This tells the user how many clusters are found. A merge step is incorporated that reduces sensitivity for the estimated number of clusters to the penalty parameters. This step occurs when the user chooses the “True” input, and is skipped when the user opts for “False” Tolerance is another variable input. This is the maximum amount of change in parameter values between iteration to declare convergence.

The third tab allows us to produce a scatter plot with color coded clusters. Like Tab 1, the user chooses the variables they want to analyze. This matrix plot differs from the one produced in Tab 1 in that the clustering is done for the user (See Figure 6).

The fourth tab produces parallel plots of the data (See Figure 7), In the Cartesian coordinate system where the axes are orthogonal, the most points we can view is three dimensions. If we instead draw the axes parallel to each other, we can view many axes on the same two-dimensional display. In our example, each of the variables in displayed as an axis, each cluster is color coded, and each broken line segment represent the collection of variables for a particular row of the data. The user can then visually examine the degree of clustering in the data.

4. Future Work and Final Comments

The project is still very much a work in progress. We have a number of tasks still to complete. These include perfecting the apps and integrating into the growClusters package, expanding the `gendata` application beyond simple random sampling to also include complex sample designs, expanding the `clustering` application to also include the hierarchal clustering algorithm from the growClusters package, and publishing the package on CRAN.

5. References

[1] Auguie, Baptiste (2017). gridExtra: Miscellaneous Functions for "Grid" Graphics. R package version 2.3. https://CRAN.R-project.org/package=gridExtra

[2] Bailey, Eric (2015). shinyBS: Twitter Bootstrap Components for Shiny. R package version 0.61. https://CRAN.R-project.org/package=shinyBS

[3] Chang, Winston, Joe Cheng, JJ Allaire, Yihui Xie and Jonathan McPherson (2019). shiny: Web Application Framework for R. R package version 1.3.2. https://CRAN.R-project.org/package=shiny

[4] Martinez, Wendy L., Angel R. Martinez and Jeffrey L. Solka. Exploratory Data Analysis with MATLAB. CRC Press, 2011. Print.

[5] Savitsky, Terrance D. (2019). About growClusters. Documentation for growClusters package (to be published on CRAN)

[6] Savitsky, Terrance D. (2016). Scalable Approximate Bayesian Inference for Outlier Detection under Informative Sampling, Journal of Machine Learning Research 17(225):1−49.

[7] Schloerke, Barret Schloerke, Jason Crowley, Di Cook, Francois Briatte,Moritz Marbach,Edwin Thoen, Amos Elberg and Joseph Larmarange. (2018). GGally: Extension to 'ggplot2'. R package version 1.4.0. https://CRAN.R-project.org/package=GGally

[8] Wickham, Hadley, Roman Francois, Lionel Henry and Kirill Müller (2019). dplyr: A Grammar of Data Manipulation. R package version 0.8.3. https://CRAN.R-project.org/package=dplyr

[9] Wickham, Hadley, Winston Chang, Lionel Henry, Thomas Lin Pedersen, Kohske Takahashi, Claus Wilke, Kara Woo and Hiroaki Yutani (2019). ggplot2: Create Elegant Data Visualisations Using the Grammar of Graphics. R package version 3.2.1. https://CRAN.R-project.org/package=ggplot2

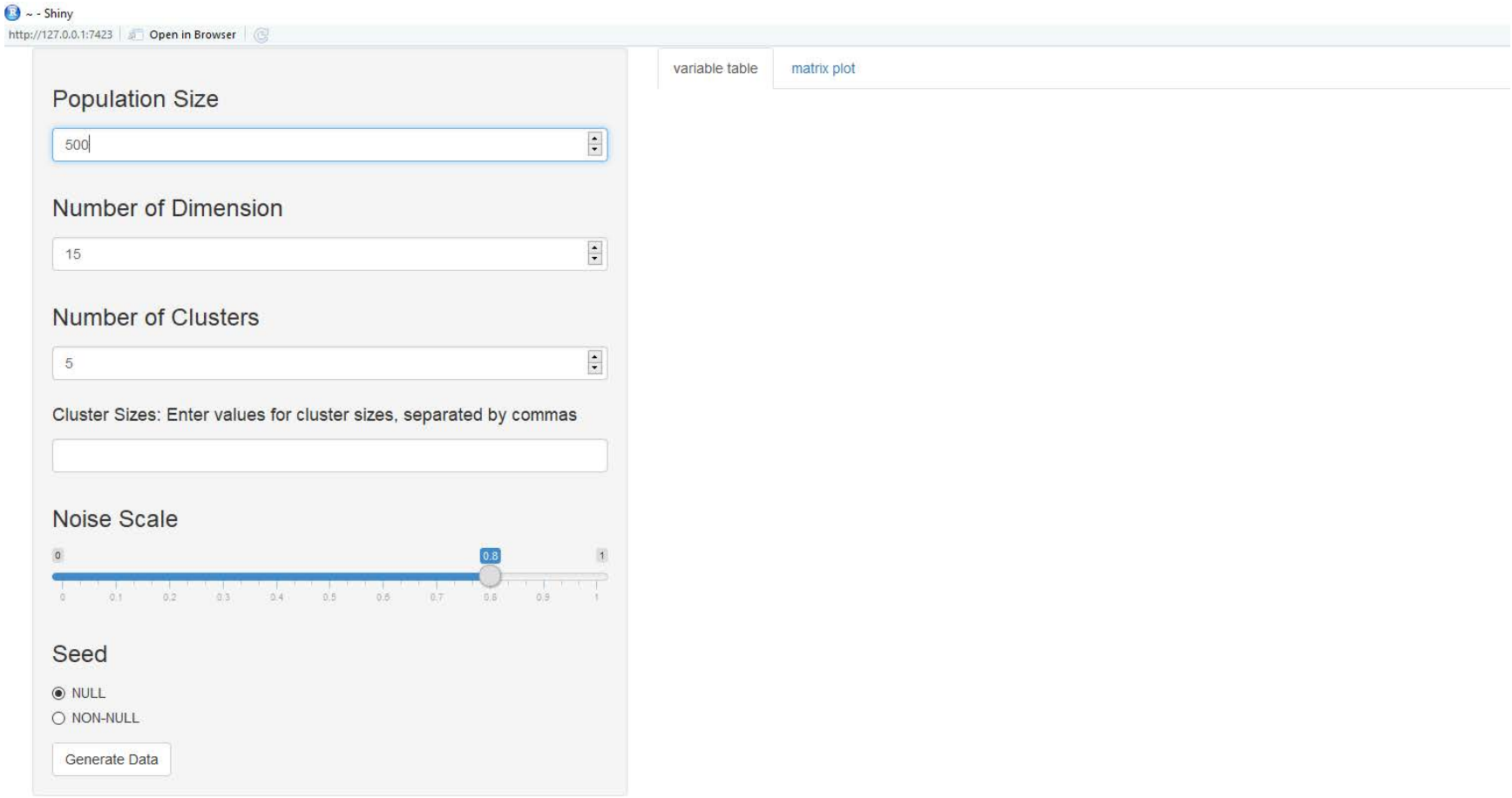

Figure 1. Screenshot of the `gendata` application at the point where the application is first opened and is awaiting user input. There are six possible user inputs. The only requirements regarding the inputs are that the cluster sizes sum to 1.0 and the number of cluster sizes must be equal to the number of clusters entry.

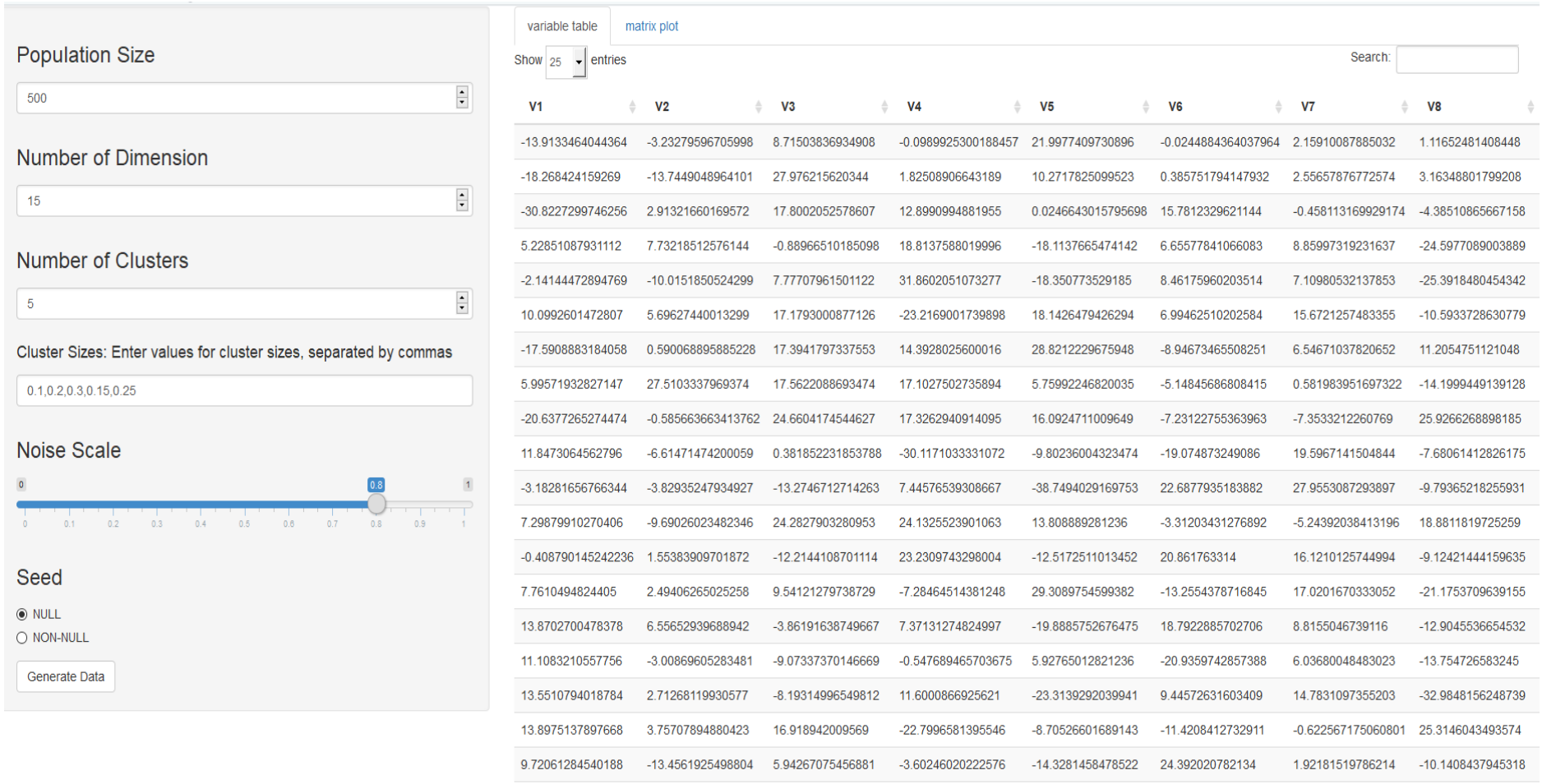

Figure 2: This is the `gendata` application after user has submitted inputs. A data set is produced and saved in the working space. Given the inputs shown, the dataset will have 500 observations and 15 variables. Five clusters will be produced, and the dataset will be completely random, since the "null" option was used.

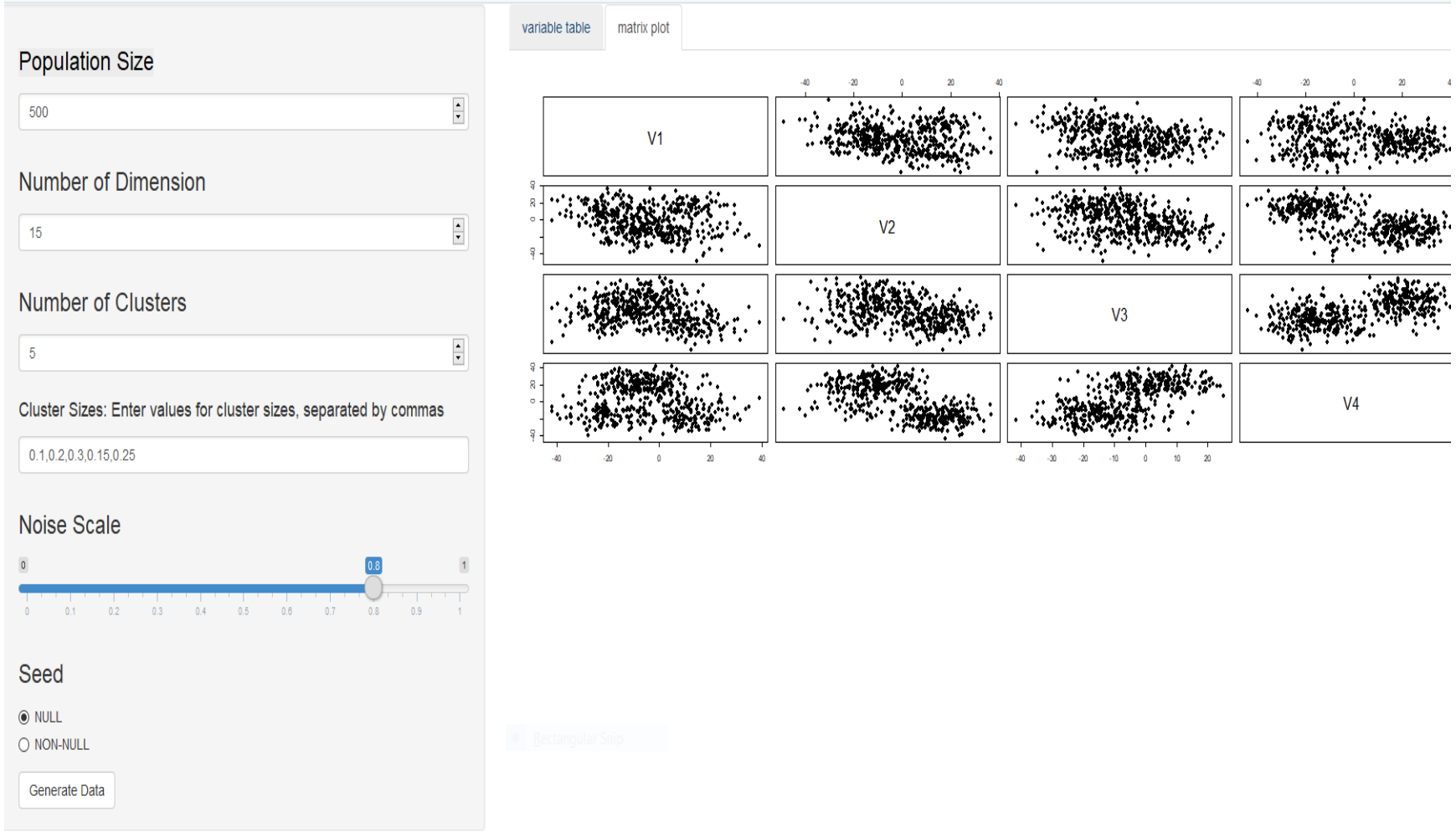

Figure 3: This is the scatterplot matrix tab of the `gendata` application. This is automatically produced when the dataset is generated. The user has the option to generate a new dataset while clicked on this tab.

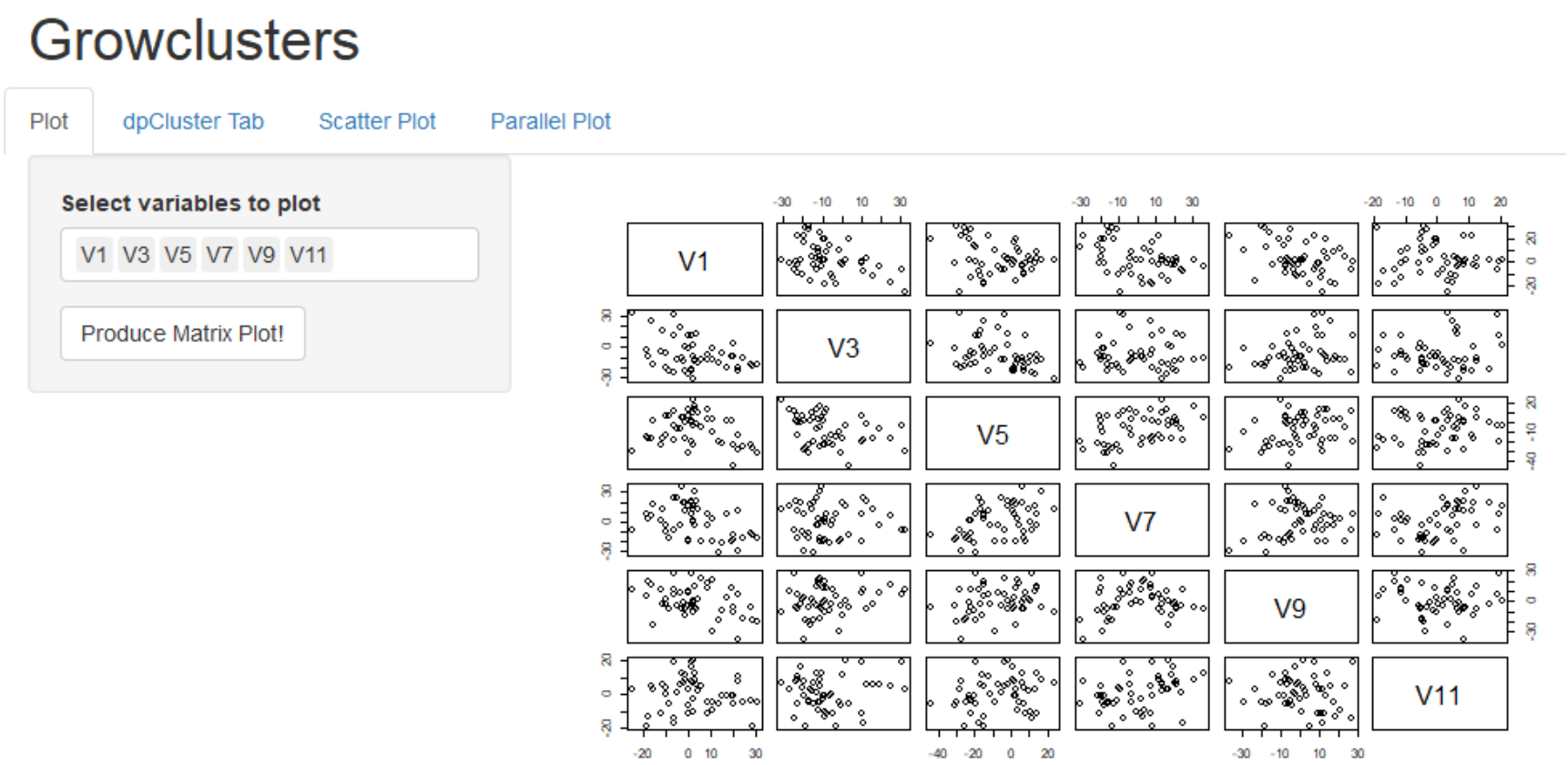

Figure 4: This is the `clustering` app. The plot tab is shown. This is a scatterplot matrix of the data. By default, the first three variables are shown. From the drop-down menu, the user selects which variables to show, and a customized scatterplot matrix is produced.

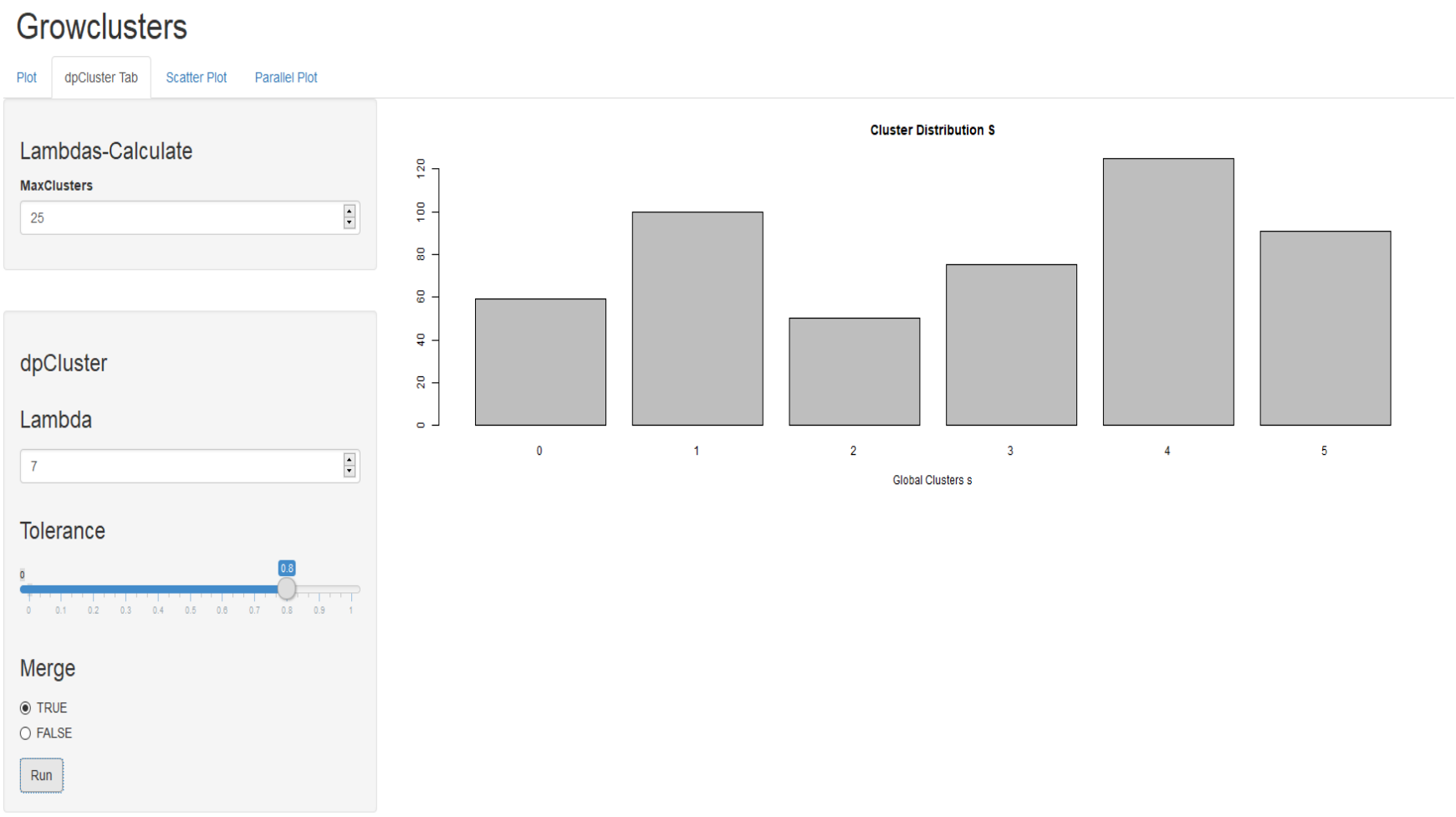

Figure 5: This is the dpcluster tab of the `clustering` application. The user can tailor the various inputs to match the clustering needs. Once the run button is clicked, a bar plot of the cluster distribution is produced. The unequal distribution of this bar plot reflects to unequal cluster sizes that were used when producing the input dataset, using the `gendata` application.

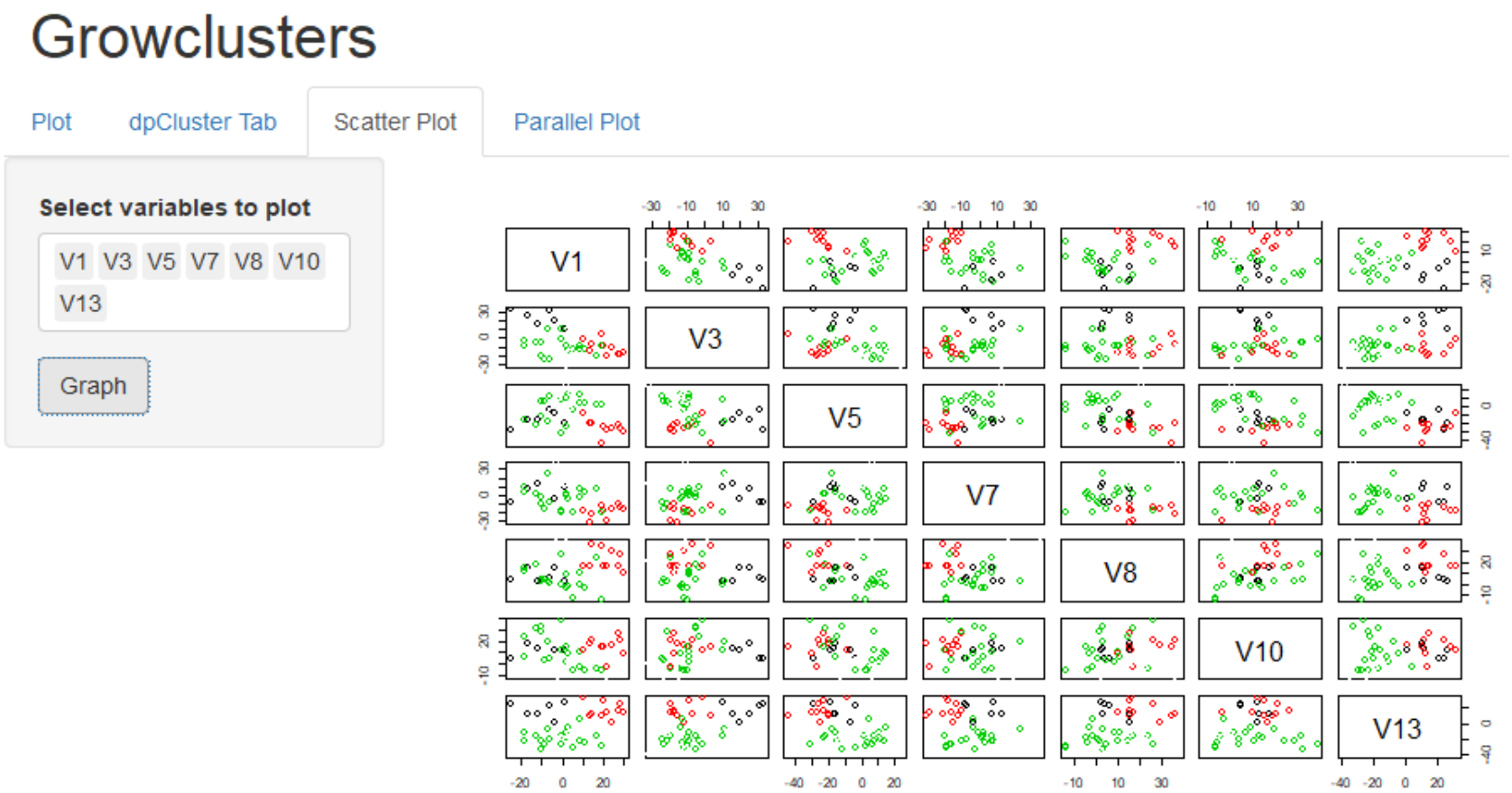

Figure 6: This is the scatter plot tab of the `clustering` application. The user selects which variables they wish to see and a customized color-coded scatterplot is produced. The user can examine the degree of clustering for each of the chosen variables.

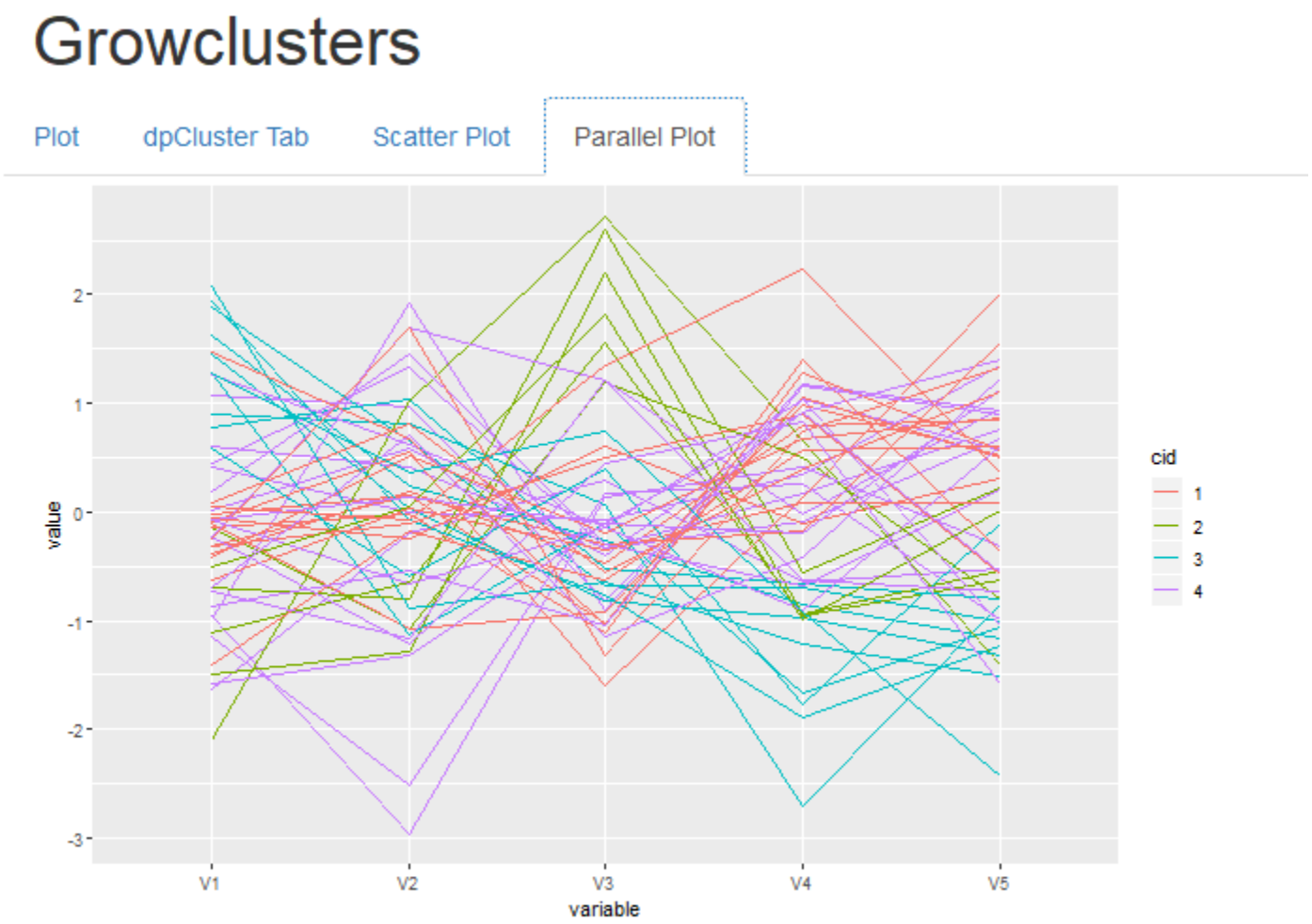

Figure 7: This is the parallel plots tab of the `clustering` application. Each broken line segment represent the collection of variables for a particular row of the data. In the plot shown, there are four clusters and five variables displayed.