Integration of Two RShiny Applications into the `growclusters` Package with an Example Implementation

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 revisits the two R Shiny applications that were introduced in our 2019 paper and will focus on additional functionality added to both of them, as well as the integration of the R Shiny applications into the `growclusters` package. We will also present an example where the R Shiny applications are used to analyze the text from past papers presented at the United Nations Economic Commission for Europe workshops.

Key words: growclusters, 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`. This paper will describe the functionality of these two applications in detail as well as show an implementation of the `clustering` application using a real dataset.

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` 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 simple plot and 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.

After the data is generated (see Figure 2), the user can click "Select Variables." This produces a drop down menu of variables for the user to select from. Once the user selects the variables, they can the click "Visualize Data" and the scatterplot matrix is produced (see Figure 3). The user can click the "Download File" link to save the synthetic data file to their computer for future use.

A second tab entitled "Hierarchal Plot and Table" is currently under construction. When completed, this tab will enable the user to produce a hierarchal version of a synthetic dataset that accounts for local dependencies, as opposed to the simple plot tab, which only has k-means global clustering.

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. When the user opens the application, the Load Data tab is displayed. The user clicks "Browse" to browse their local drives in search of the dataset they wish to load. Once the dataset is loaded, 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 5). At this point, clustering is not performed, but instead allows the user to examine the data and look for groupings.

On the second tab, "dpCluster," the user can alter three inputs, including the expected number of cluster, which governs the number of clusters which are formed; the larger the lambda value, the more clusters that are discovered. Here we're using a function called lambda_range4() to generate a vector of lambda penalty parameters for a range of 1:max_clusters. The user is able to change the expected number of clusters input, which impacts what the max_clusters parameter can be. When ready, the user presses a "Run" button which allows a bar plot of the cluster counts to be displayed (see Figure 6). 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, "Scatter Plot," 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 7).

The fourth tab, "Parallel Plot," produces parallel plots of the data (see Figure 8), 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 is 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. Each line represents an observation – in this case a five dimensional observation. The user can then visually examine the degree of clustering in the data. The colors correspond to the cluster ID. We see 'good' clusters when lines of the same color have similar trajectories.

4. Real World Data Background

After hosting a series of 12 workshops stating in 2000, the United Nations Economic Commission for Europe wanted to better understand what methods or topics they've discussed over the years. [5] A classification of existing methods for data editing and imputation based on papers presented in previous UNECE work sessions on data editing was conducted. Papers approximately 10 pages in length were used, thus giving much more content than if only abstracts were being used. The goal was to cluster documents so and hopefully this would result in each cluster describing similar methods and ideas.
Before encoding, the data was pre-processed using a number of cleaning steps, including removal of header and reference sections, deletion of special characters and end of sentence punctuation, removal of short, long, stop, and infrequent words, and conversion of all text to lower case.

The text was then converted to numerical format to allow for computing. A widely used method is to encode the documents in a term-document matrix (TDM). This is also known as the bag-of-words approach. The ij th entry in the TDM corresponds to the number of times the i-th word appears in the j-th document. Each column of the TDM represents a word frequency distribution for a given document or paper. For further details, please see Martinez and Savitsky. [5]

The above techniques produced n = 427 papers or documents to cluster. There are p = 10,737 unique words in the corpus after the removal of the domain-specific stop words, with two types of encodings – raw frequencies and binary. Through dimension reduction techniques, the 10,307 variables were reduced to five. The analysis that follows examines the binary dataset.

5. Implementation of the `clustering` Application Using the UNECE Dataset

As described in Section 3, we load the binary dataset into the `clustering` application. We can also choose which variables we wish to do an initial overview for in a scatterplot. What we're looking at in the scatterplot is a pairwise comparison of the variables. For example, in the first row, we're comparing variable V1 to each of the four other variables, with V1 on the vertical axis and the other variables on each of the horizontal axes. (See Figure 8.)

In Figure 9, we see the clustering results in Tab 2 for our binary dataset when we set the expected number of clusters equal to both six and ten, Note, the scales on the vertical axes are different. So this is not meant as a direct comparison. Additionally, when we say "expected number of clusters," the results are not always exactly as specified. For this data, when we had an expected number of clusters set to 10, we actually got 11. This is not a problem.

In the Scatter Plot tab (See Figure 10), we can really see the clustering is very observable. Each dot represents a paper. Note for example how the greens are clustered together, especially for V1 and V3. So we're seeing clustering based on similar features of clusters of papers. There is less separability and some over plotting going on in V3 and V5. This is ok because we're dealing with five dimensions, and sometimes it's not possible to fully see the clustering in a 2D plot.

The Parallel Plot Tab (See Figure 11) shows quite a bit of clustering going on. Remember, each line represents one element of the population and each color is a cluster. So we can clearly see patterns across each observation. We see bundles of lines for each color that sort of stick together. For example, the light greens all tend to dip for variable/dimension two.

6. Future Work and Final Comments

The project is still very much a work in progress. We have a number of tasks to complete. These include completing the hierarchal tab of the `gendata` application, integrating both apps into the `growclusters` package, expanding the expanding the `clustering` application to also include the hierarchal clustering algorithm from the `growclusters` package, publishing the package on CRAN and eventually publishing another paper with another real life dataset.

7. 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] Martinez, Wendy L., and Terrance Savitsky (2019). Towards a Taxonomy of Statistical Data Editing Methods. Working Paper presented at the UNITED NATIONS ECONOMIC COMMISSION FOR EUROPE CONFERENCE OF EUROPEAN STATISTICIANS. https://www.unece.org/fileadmin/DAM/stats/documents/ece/ces/ge.44/2018/T_USA_Martinez_Paper.pdf

[6] Savitsky, Terrance D. (2019). About `growclusters`. Documentation for `growclusters` package (to be published on CRAN)

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

[8] 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

[9] 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

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

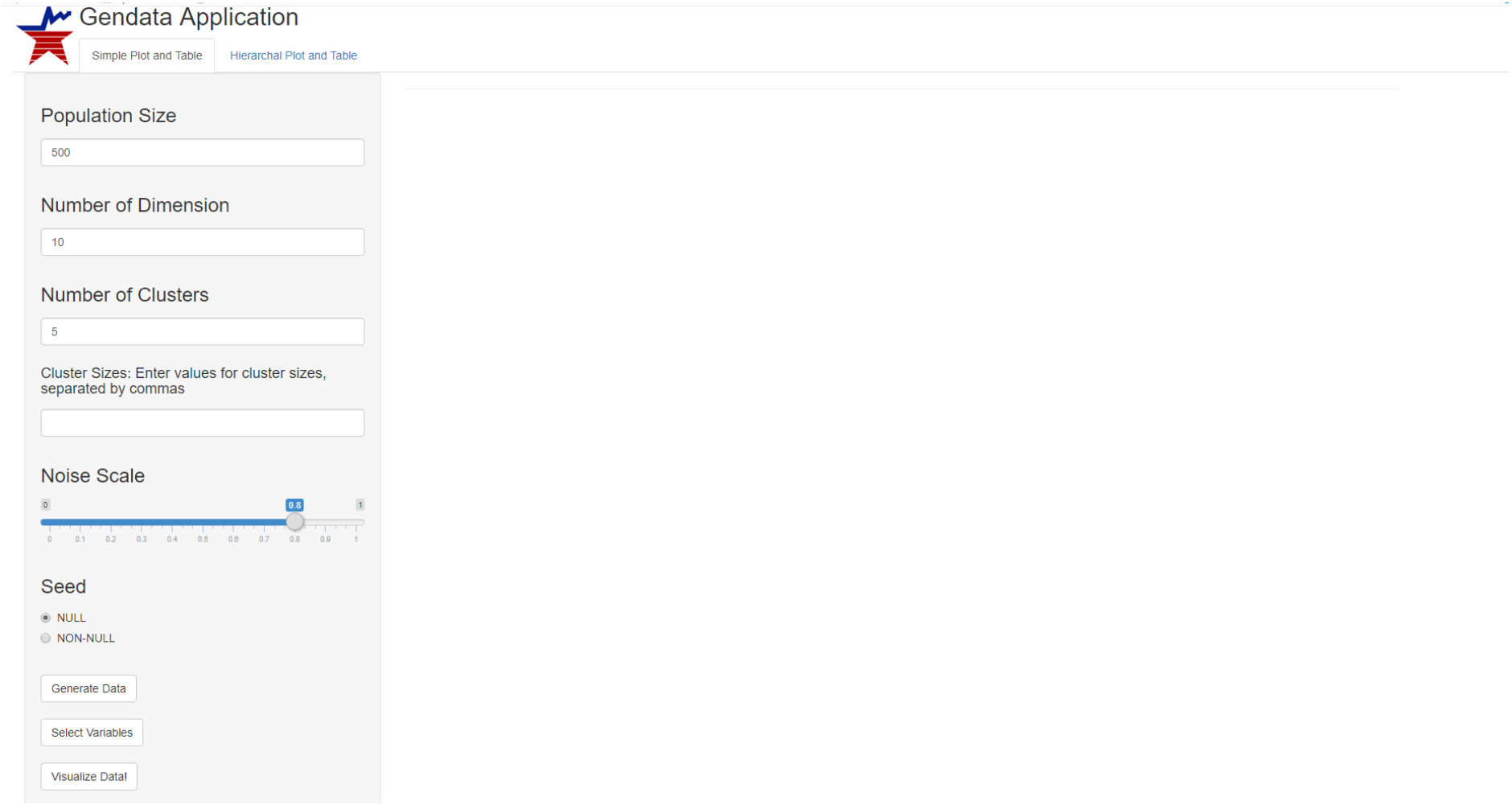


Figure 1. This is a screenshot of the `gendata` application at the point where the application is first opened and is awaiting user input. There are six possible user inputs. The only requirements regarding the inputs are that the cluster sizes sum to 1.0 and the number of cluster sizes must be equal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

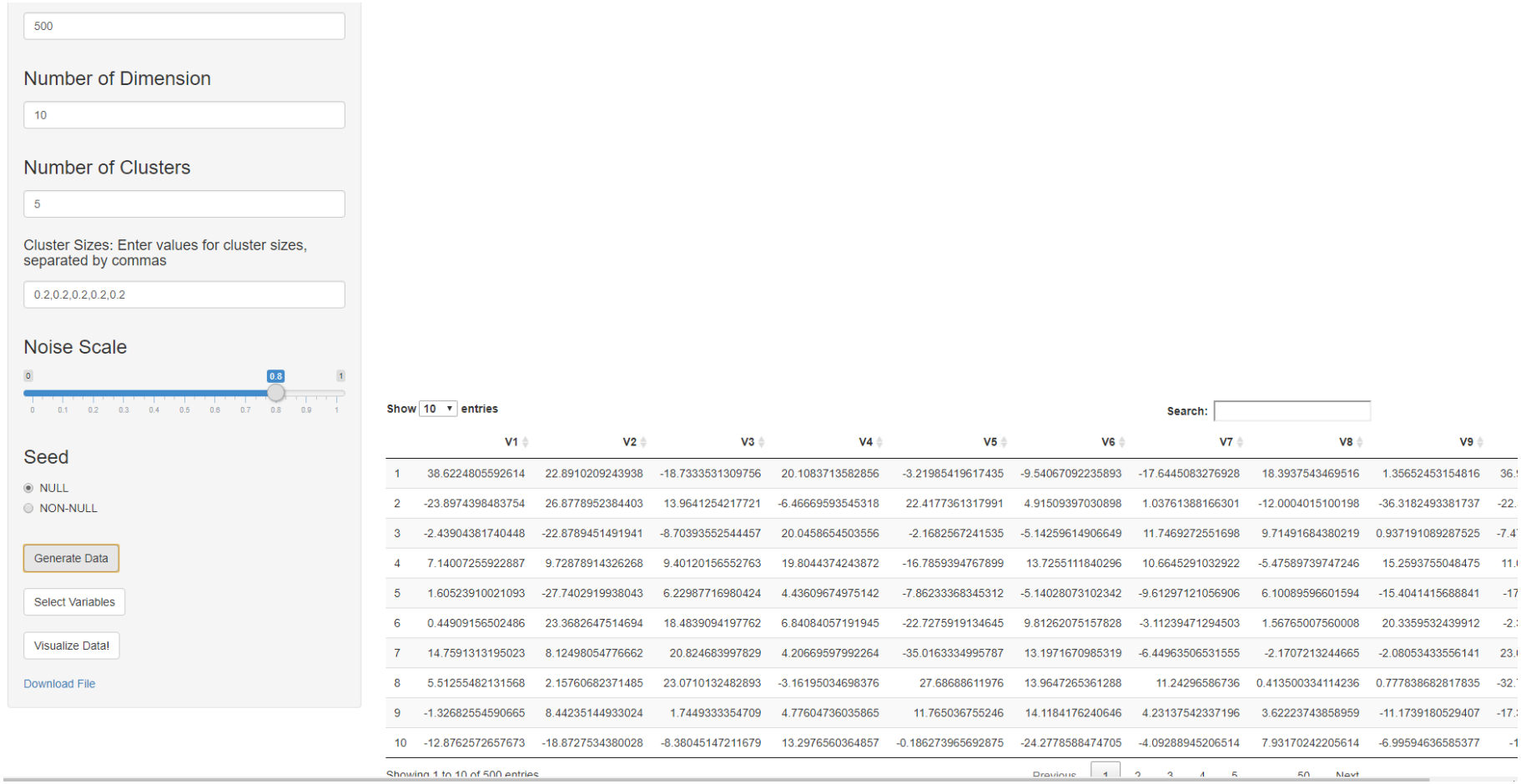
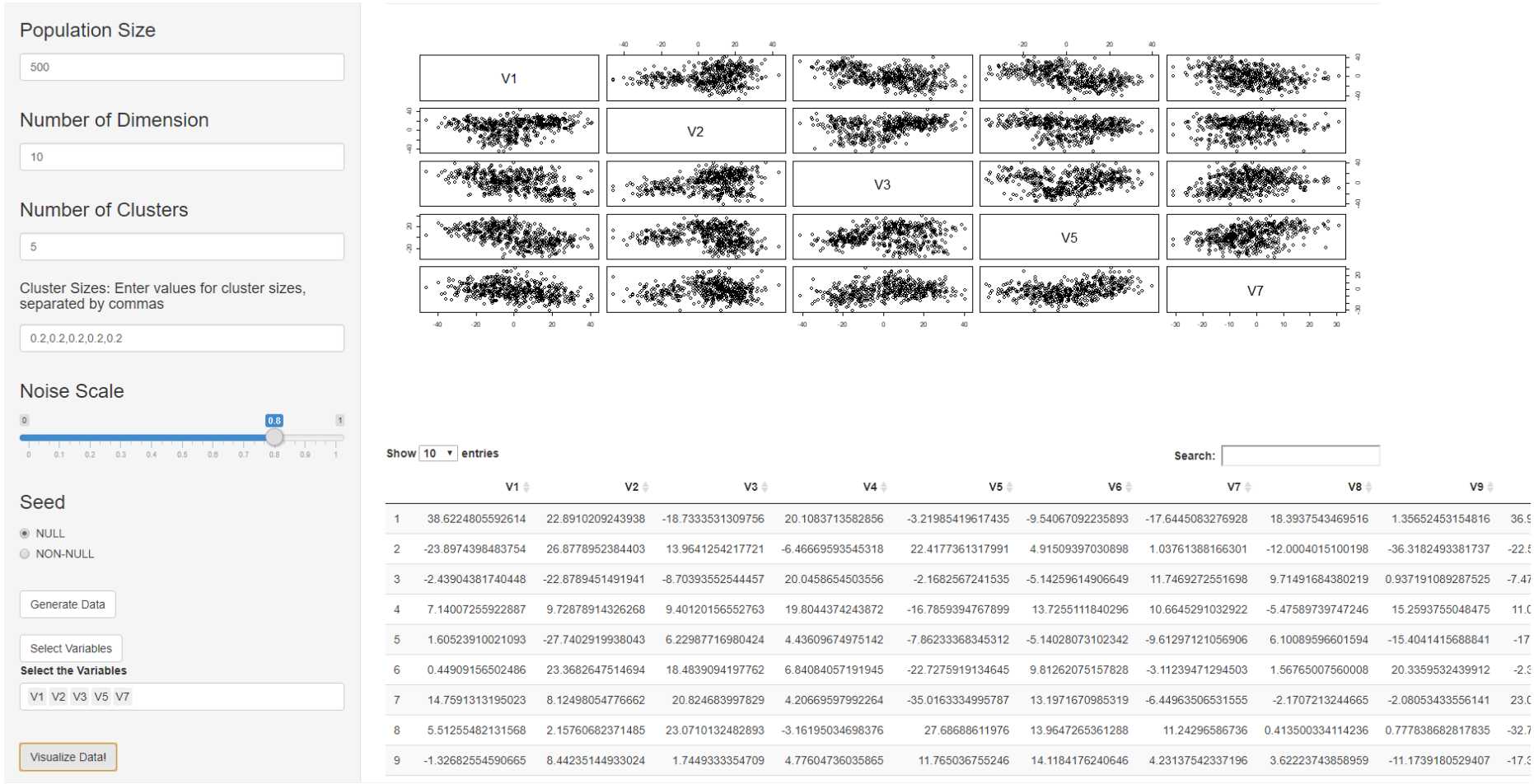


Figure 2: This is the `gendata` application after user has submitted inputs and clicked "Generate Data." A data set is produced and saved in the working space. Given the inputs shown, the dataset will have 500 observations and 10 variables. Five clusters will be produced, and the dataset will be completely random, since the "null" option was used.



	V1	V2	V3	V4	V5	V6	V7	V8	V9	
1	38.6224805592614	22.8910209243938	-18.7333531309756	20.1083713582856	-3.21985419617435	-9.54067092235893	-17.6445083276928	18.3937543469516	1.35652453154816	36.9
2	-23.8974398483754	26.8778952384403	13.9641254217721	-6.46669593545318	22.4177361317991	4.91509397030898	1.03761388166301	-12.0004015100198	-36.3182493381737	-22.8
3	-2.43904381740448	-22.8789451491941	-8.70393552544457	20.0458654503556	-2.1682567241535	-5.14259614906649	11.7469272551698	9.71491684380219	0.937191089287525	-7.47
4	7.14007255922887	9.72878914326268	9.40120156552763	19.8044374243872	-16.7859394767899	13.7255111840296	10.6645291032922	-5.47589739747246	15.2593755048475	11.0
5	1.60523910021093	-27.7402919938043	6.22987716980424	4.43609674975142	-7.86233368345312	-5.14028073102342	-9.61297121056906	6.10089596601594	-15.4041415688841	-17
6	0.44909156502486	23.3682647514694	18.4839094197762	6.84084057191945	-22.7275919134645	9.81262075157828	-3.11239471294503	1.56765007560008	20.3359532439912	-2.3
7	14.7591313195023	8.12498054776662	20.824683997829	4.20669597992264	-35.0163334995787	13.1971670985319	-6.44963506531555	-2.1707213244665	-2.08053433556141	23.0
8	5.51255482131568	2.15760682371485	23.0710132482893	-3.16195034698376	27.68688611976	13.9647265361288	11.24296586736	0.413500334114236	0.777838682817835	-32.7
9	-1.32682554590665	8.44235144933024	1.7449333354709	4.77604736035865	11.765036755246	14.1184176240646	4.23137542337196	3.62223743858959	-11.1739180529407	-17.3

Figure 3: This is the simple plot and table tab of the `gendata` application. After the data is generated (see Figure 2), the user can click "Select Variables." This produces a drop down menu of variables for the user to select from. Once the user selects the variables, they can the click "Visualize Data" and the scatterplot matrix is produced.

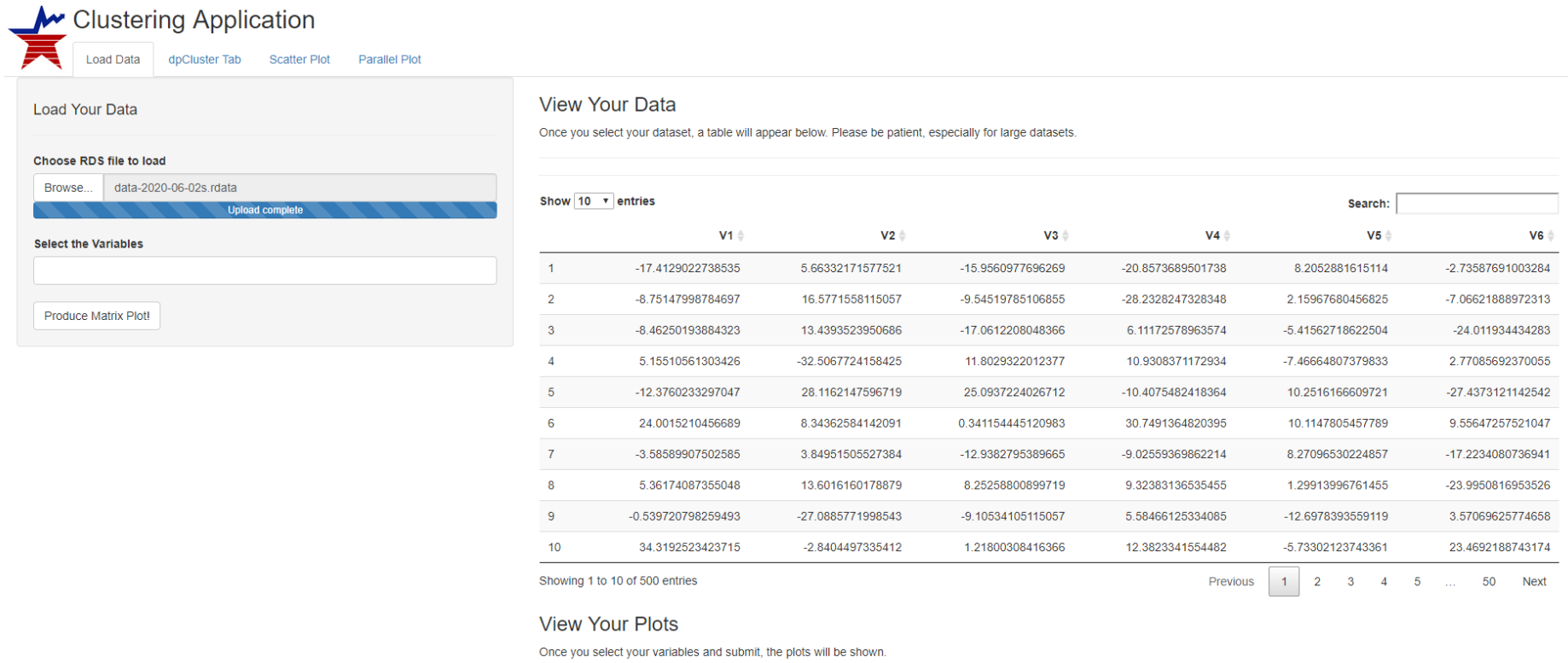


Figure 4: This is the `clustering` app. The load data tab is shown. The user browses their personal drives and loads the .rds data file they wish to view.

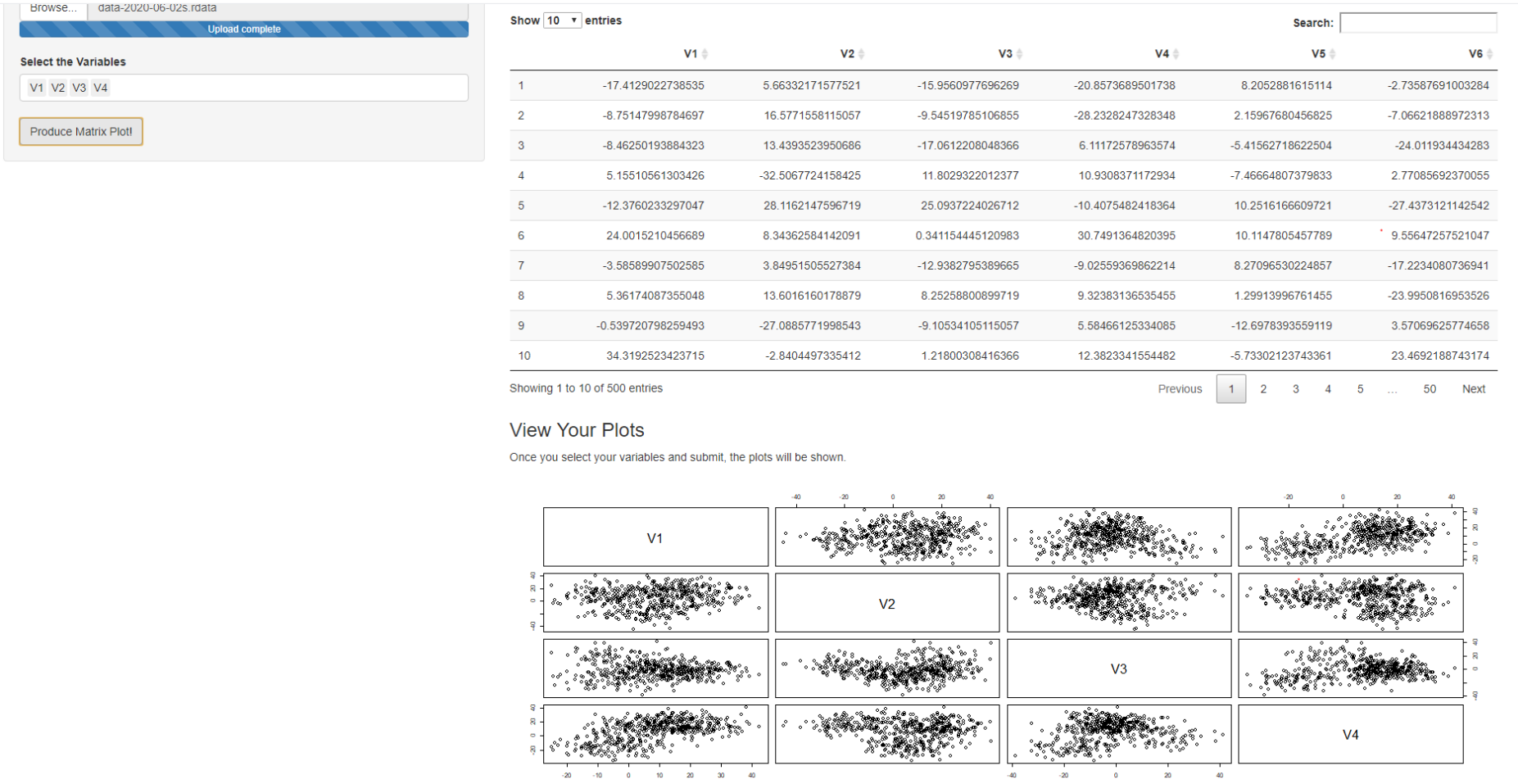


Figure 5: This is the `clustering` application Load Data tab. After loading and viewing the data, the user can press the "Produce Matrix Plot" button to view a simple matrix plot of the loaded data.

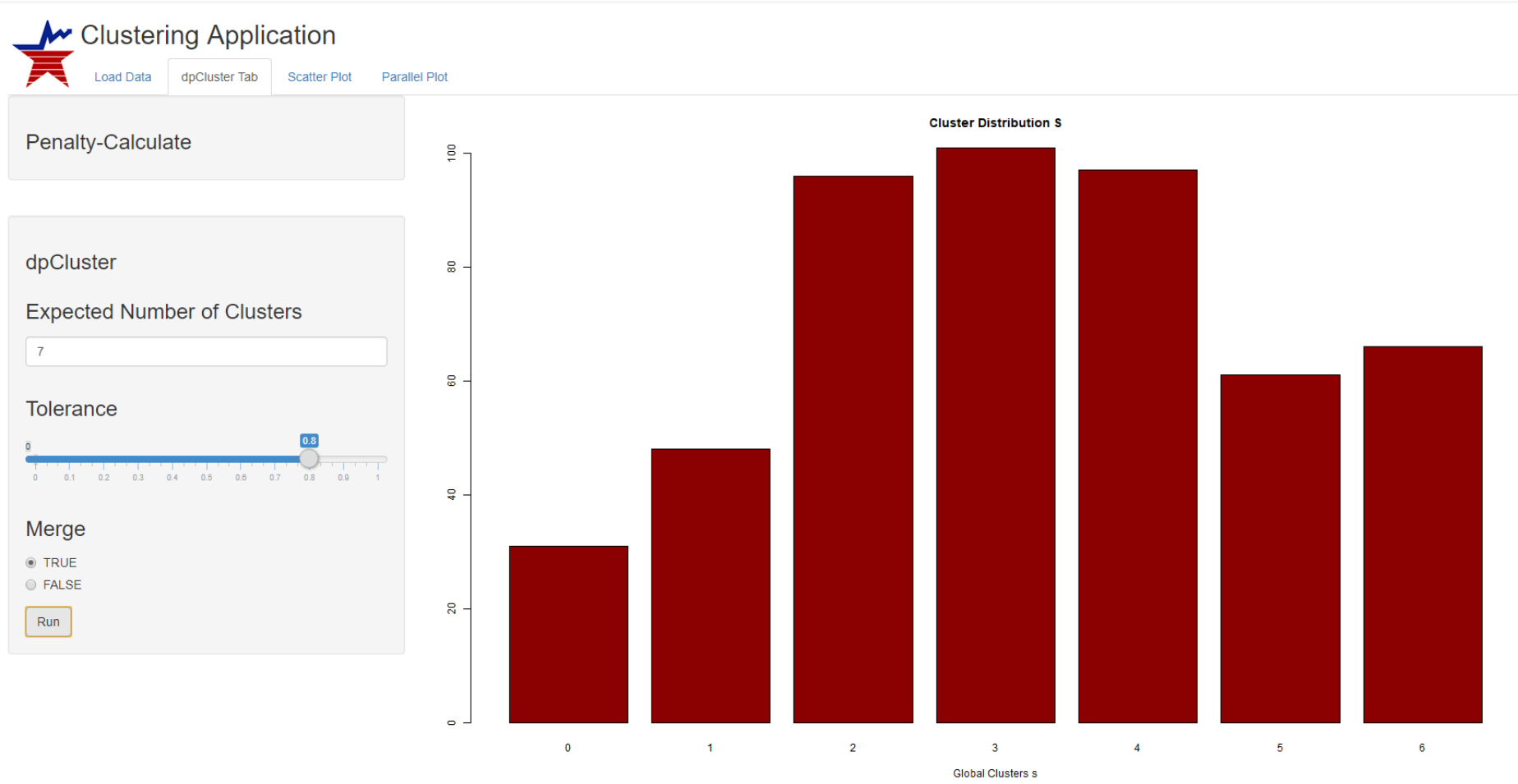


Figure 6: This is the dpcluster tab of the `clustering` application. The user can tailor the various inputs to match the clustering needs. Once the run button is clicked, a bar plot of the cluster distribution is produced. The unequal distribution of this bar plot reflects to unequal cluster sizes that were used when producing the input dataset, using the `gendata` application.

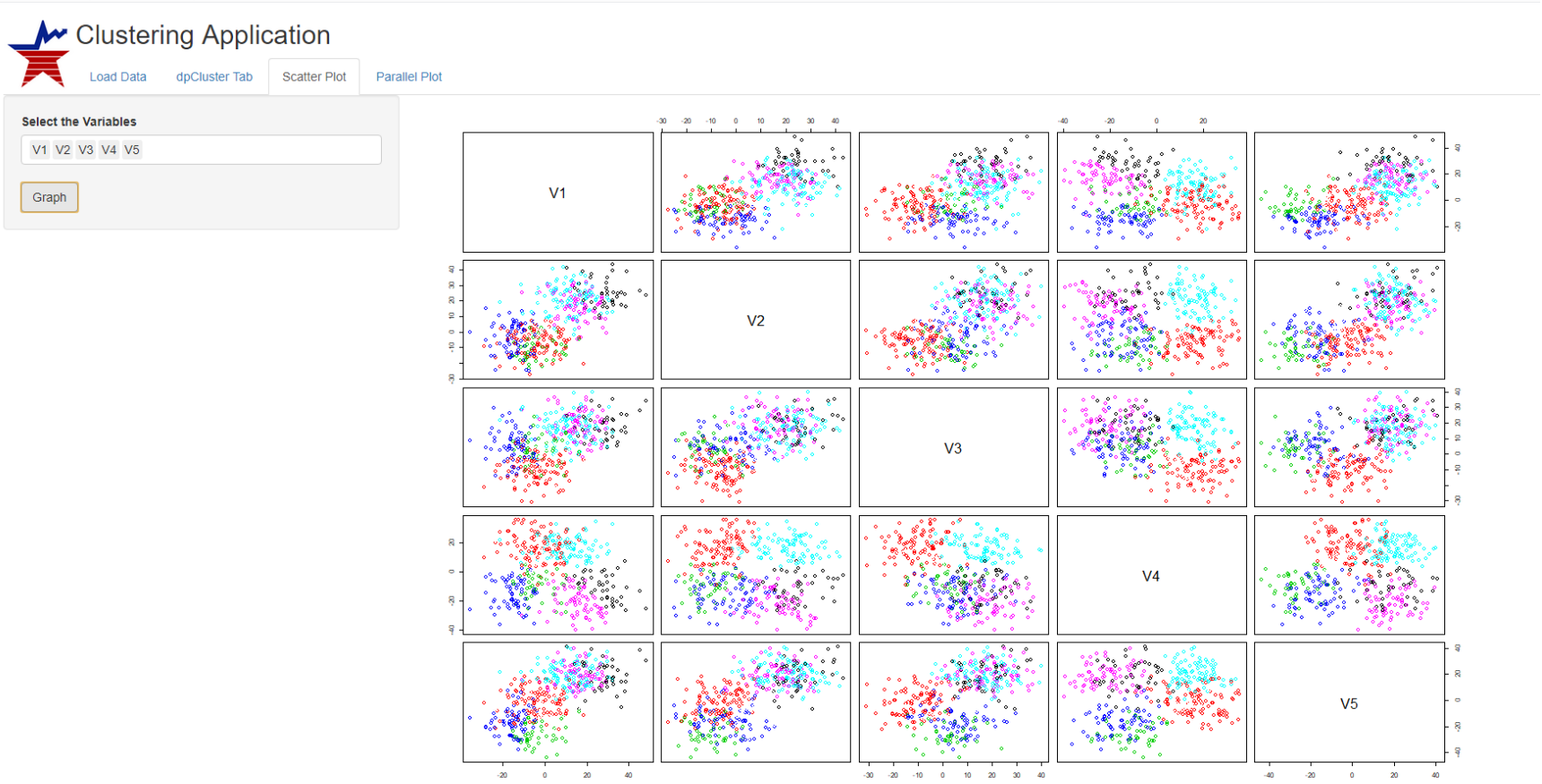


Figure 5: This is the scatter plot tab of the `clustering` application. The user selects which variables they wish to see and a customized color-coded scatterplot is produced. The user can examine the degree of clustering for each of the chosen variables.

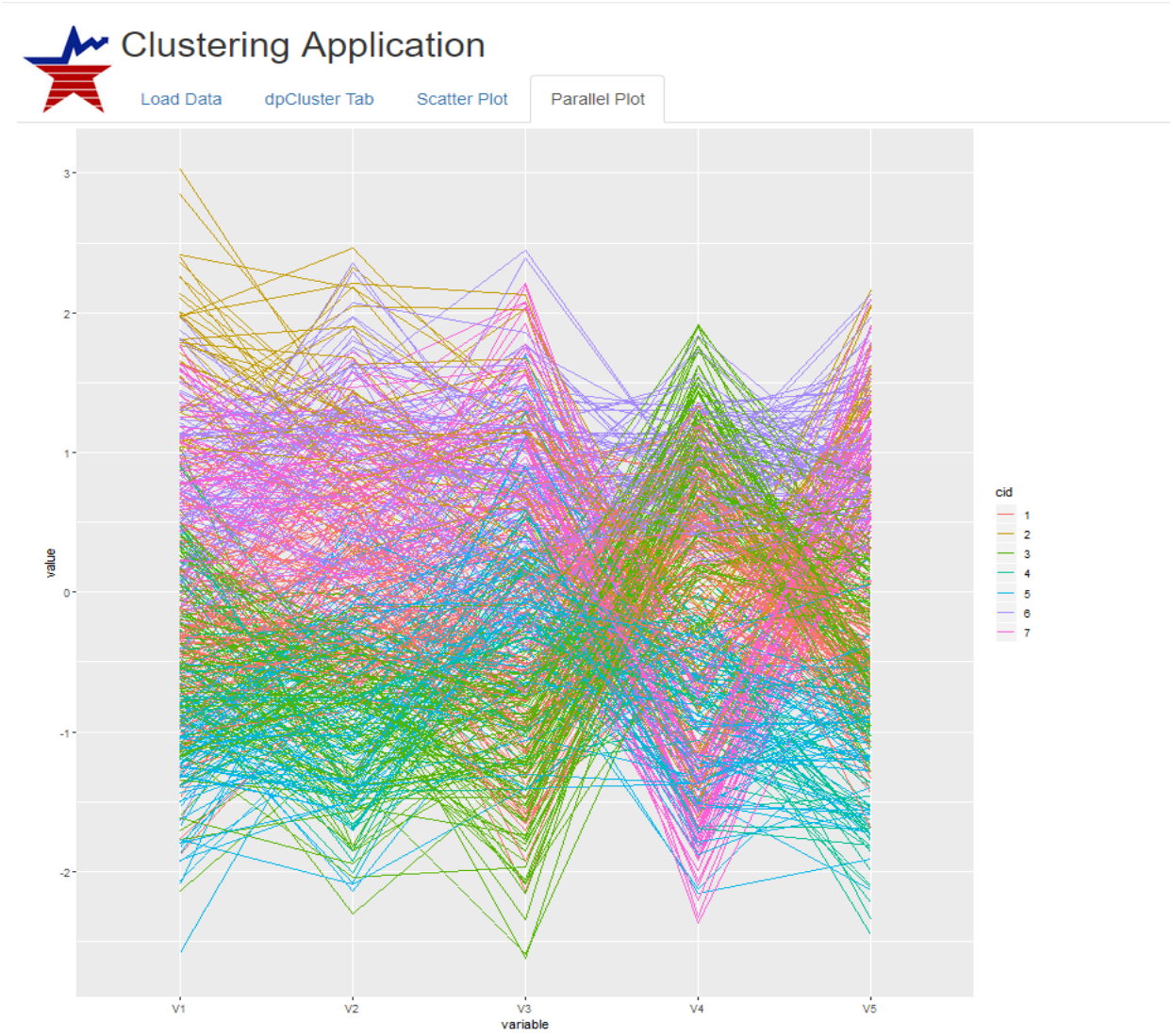
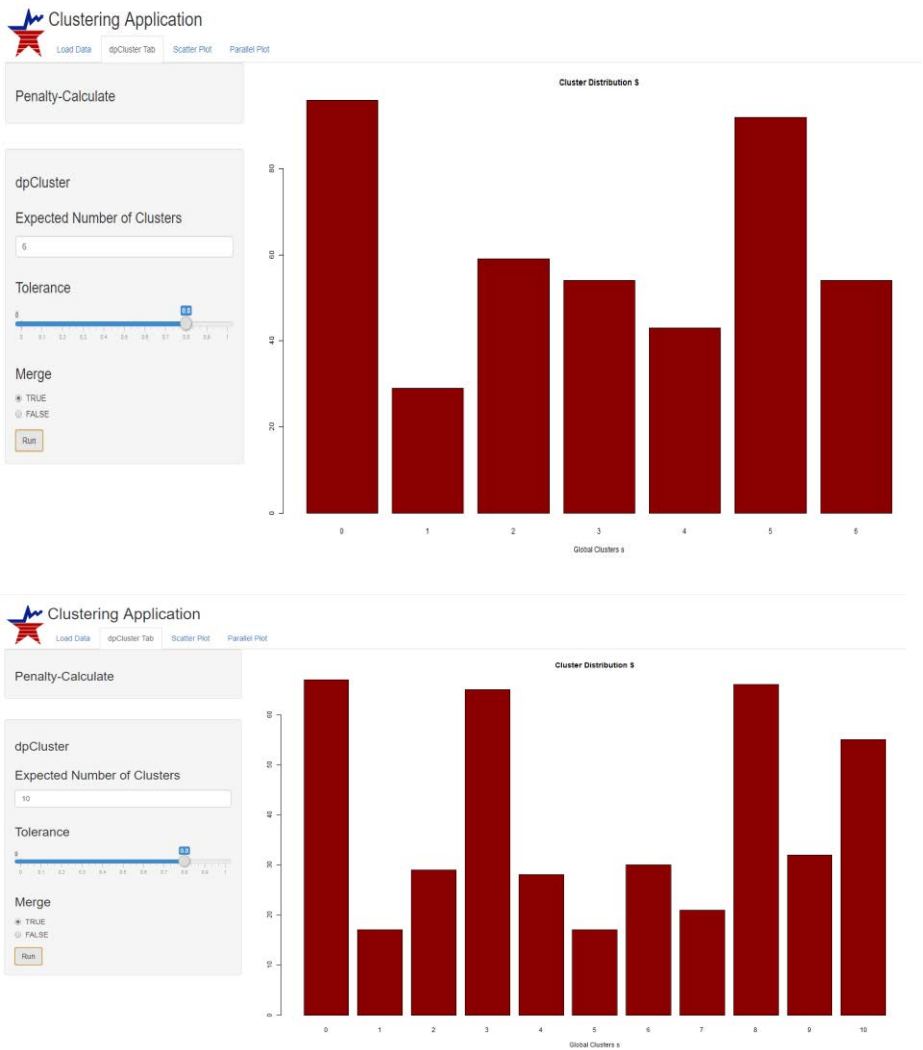


Figure 6: This is the parallel Plot tab of the `clustering` application. Each broken line segment represent the collection of variables for a particular row of the data. In the plot shown, there are seven clusters and five variables displayed.

Figure 8: This is the Load Data tab of the `clustering` application. The binary dataset has been uploaded, and the user has chosen to view a scatterplot containing each of the five variables.



Figures 9a and 9b: This is the dpcluster tab of the `clustering` application. The user has set the expected number of clusters equal to six in the first graph, and to 10 in the second graph. Note that the number of clusters that result is not always equal to the expected number of clusters (Figure 9b.)

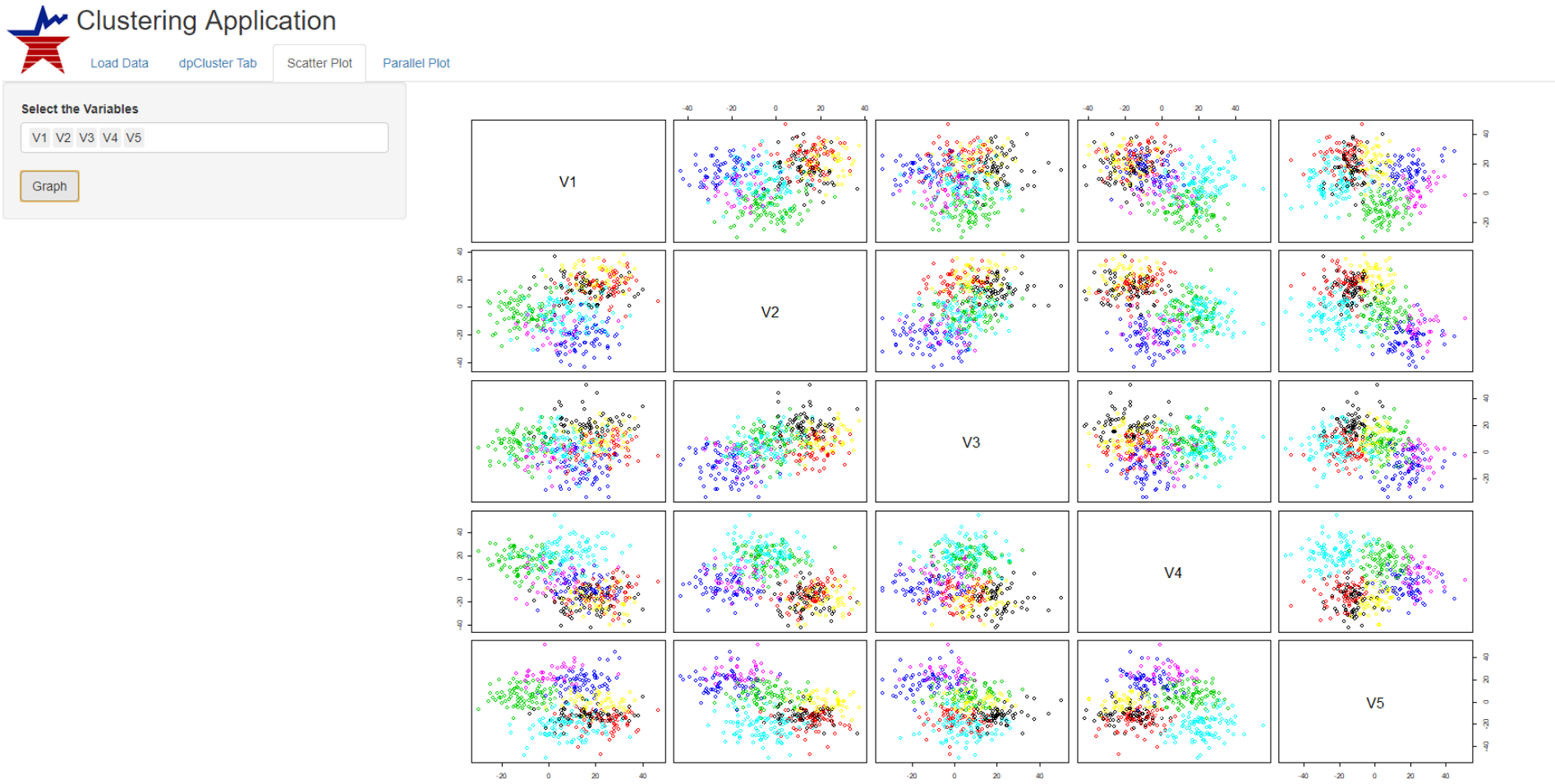


Figure 10: This is the Scatter Plot tab of the `clustering` application for the binary data. Each dot represents a paper. We observe a good deal of clustering.

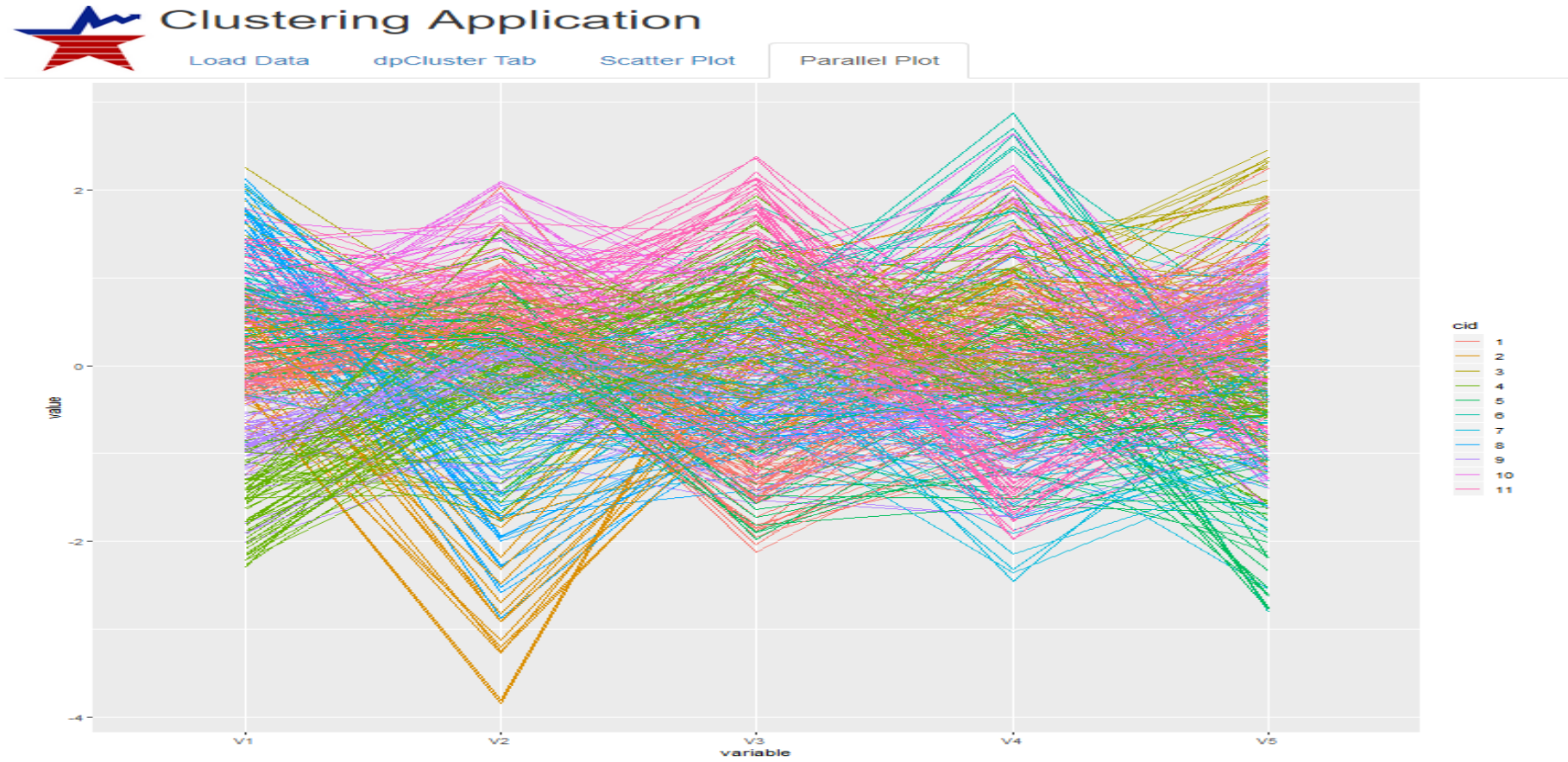


Figure 10: This is the Parallel Plot tab of the `clustering` application for the binary data. Each line represents one of the 427 elements. Each color represents a cluster, and each variable represents a dimension.