Parameter Space Exploration for VisRSeq

Joseph Pober
University of Vienna

ABSTRACT

We propose an extension to the software VisRSeq [5], which allows non-programmers to use R through a GUI, provides explanatory tool-tips and default values to further support inexperienced users. The previous version of VisRSeq only allowed the user to enter a single set of parameters and therefore to generate a single output. This extension enables users to input ranges of parameters instead of only a single set of parameters to visualize their data. The program samples the provided parameter space and generates multiple outputs. To present these outputs to the user VisRSeq clusters it using hierarchical agglomerative clustering. These resulting clusters can be interactively explored by the user.

Index Terms: Clustering, Interactive Visualization, Parameter Space Exploration, VisRSeq

1 MOTIVATION

VisRSeq allows users to visualize datasets through R-Apps. This helps people with little to no experience in programming to utilize R. Instead of actually programming the desired plot, one simply selects a visualization method (i.e. histogram) and enters the parameters into the provided text boxes. The problem is that this whole workflow only works with exactly one set of parameters at a time. Many users will want to test multiple sets of parameters to find the desired result. Entering a set of parameters and then having to wait for the computer to finish the calculations does not only waste a lot of time it also interrupts the workflow of the user considerably.

Solving this problem requires users to be able to enter ranges of parameters instead of only a single set. With this, a problem in presentation arises, since many results in form of complex objects may appear and have to presented to the user. This will be done by clustering the results and only showing the median of those clusters to the user. They can then choose to inspect a cluster, which either shows all results of that cluster or shows again some sub-clusters of this cluster. Following the design principle of showing first an overview and only later zooming in and providing details.

1.1 Data

The data to be used with the bioconductor packages is a table consisting mainly of floats. Each row represents a gene and each column a cell type. The floats try to show the likelihood of a gene being responsible for a cell to form a certain cell type. Since this is real world data, there is quite a lot of noise and often multiple samples of the same type must be taken and recorded to be sure that the results are not distorted too much by noise.

Since the program delivers a general solution, most Apps work with vastly different data, but the bioconducter packeges (i.e. edgeR) require a specific structure as described above.

1.2 Users

The users will have little to no programming knowledge, but want to have powerful visualizations of their data. Therefore a GUI is provided and default values to support the users performing their tasks.

1.3 Tasks

Discover Users want to discover the underlying structure of their data. Either by finding clusters or a good set of parameters to be used for similar data.

Explore Neither the location, nor the exact plot is known to the user. Therefore plots are clustered based on similarity to support the user in their exploration task.

Identify The parameters used to generate the median of a cluster are shown when it is selected.

Compare Often a user will want to compare multiple clusters with each other to find differences or to figure out which cluster they should explore next.

Summarize Clustering of the output plots is used to provide a general overview of the visualizations.

Targets The distribution of all parameters used to generate the output is shown. This can be compared to the distribution of parameters in a specific cluster.

The variance inside of each cluster and its size in relation to its parent is shown to support users finding outliers and avoid exploring homogeneous clusters.

2 RELATED WORK

[6] provided many different visualization techniques and the task abstraction used in section 1.3.

VAICo [7] allows users to compare many images and highlight areas that are different. It searches for pixels in all pictures, which are different and groups them together based on proximity. This tool makes it easy to compare images, which are mostly the same with slight differences. Their pixel grouping algorithm and analysis of image differences can be used for image based comparison of different plots for VisRSeq. Even though VisRSeq also provides images of plots it is not desired to cluster them according to their pixel values. Instead it is much more preferred to use data, generated by R, and the similarity measure discussed in [10].

Clustering and similarity measures are also discussed in [12, 2, 1] but those lack simplicity or flexibility provided by [10], which allows comparisons of completely different clusterings. It might be a bit simplistic, but achieves good results, since plots that are visually similar get clustered together.

[8] discusses various general methods to visually explore a parameter space. Since informed trial and error strategies are often extremely time and resource inefficient, providing the program with a parameter space is a more desired approach. Using this approach a problem in presentation arises, since not only one but multiple outputs have to be presented to the user. Juxtaposing them works for small sets, but scales poorly.

The Global-to-Local strategy proposed in [8] was used to address exactly this problem, by providing the user first with an overview, achieved through clustering, and letting them drill down to specific, individual outputs.

[3] proposed an algorithm to generate as square as possible tree maps to combat very long and thing rectangles that can appear using traditional tree maps. Those long rectangles are undesired, since they make interacting with them difficult and almost impossible to

read if they convey more then just area information. It is also difficult to compare sizes of elongated rectangles.

We did not encode the size (amount of plots) of the cluster in its size on the screen, because that could potentially lead to clusters that are difficult to read and compare. The idea of keeping everything as square as possible was used instead in the arrangement on screen of the clusters themselves.

[4] provided insight into the task of comparing complex objects with each other. The different comparison techniques were used for the task of comparing clusters in VisRSeq, since those are complex objects.

Scented widgets as presented in [11] are a valuable tool in visualizing interface elements. They add an additional visual scent to a widget, which supports users in navigating the visualized information space. This approach was used in VisRSeq to display additional histograms above input ranges to show the user the distribution of the chosen parameters.

3 Approach

The most important question always was how to present many complex objects, which all need a lot of screen space to be readable, to the user. Clustering was used to reduce the potential amount of simultaneous objects on the screen. Using the Global-to-Local strategy by [8].

The visualization design focuses on allowing the user to interactively and efficiently exploring the plots generated by the VisRSeq.

For this design study [9] was used as a guideline. The work was done iteratively with frequent meetings with the collaborator to gather feedback. This was used to improve upon previous results.

4 Results

4.1 Input

The previous version of VisRSeq only allowed for a single set of parameters, we present the updated version, which also enables the user to input ranges of parameters. Figure 1 shows the parameter panel when the user just wants a single output in comparison to the parameter panel when the user desires ranges of parameters.

The input method, group1 and group2, has not changed, since the program will use the same input for each run. The attributes selected here will be the ones used from the provided input dataset for the visualization.

The drop-down-menu was changed to a series of check-boxes, to allow for multiple selection.

All numerical inputs, integers and doubles, where changed from a spinner to double sliders. These are used to select the min and max values of this parameter. The slider with the lower value will be automatically the minimum and the slider with the larger value will be the maximum. The values next to the slider display the current minimum and maximum.

The drop-down-menu on to bottom left allows users to switch between these two views. Next to it the amount of runs can be entered.

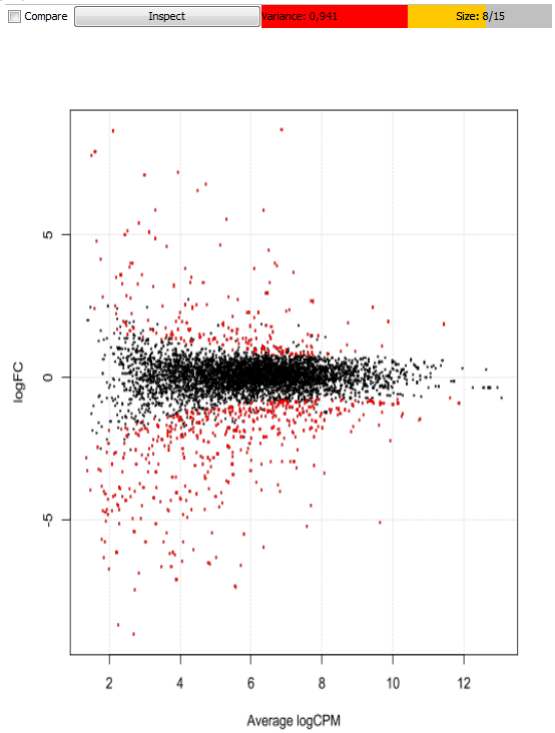
4.2 Presentation

Figure 2 shows the program, before generating output. The panel ”A” stores all imported input files and the data generated by R. ”B” keeps track of the users history and supports them in navigating through the output. ”C” is the workspace where the current clusters are displayed. ”D” is the apps panel from which the user chooses an app to visualize their data. ”E” is the previously discussed parameter panel. ”F” shows the console and displays output messages.

Figure 3 shows how the results are displayed after the user has entered the desired ranges for the parameters and hit the ”Run” button. The output is clustered based on a similarity measure, provided by [10]. The resulting clusters are again clustered to form a hierarchy until a sufficient amount of clusters is reached. The resulting top level clusters of this non-binary tree are shown to the user in the workspace panel.

Each cluster is represented by the plot which is the most similar to all other plots in its clusters, which is the median after sorting them by inter-similarity to each other.

The clusters are arranged in a grid that is as square as possible, sorted by their size. A square grid has two benefits: most plots are square themselves and therefore do not appear squished or stretched, while using the provided space efficiently; the workspace area is very often square which leads to a square grid using its space well.



Additional information about each cluster is located at the very top of it. The first two elements ”Compare” and ”Inspect” only appear when the user selects or hovers over a cluster to avoid cluttering. The last two, ”Variance” and ”Size” are always visible to allow for easy comparison since eyes beat memory as discussed in [6]. This is also the reason why all child clusters are presented in juxtaposition to each other. Since this does not scale very well the user can force the program to always have at most a certain number of clusters visible, by allowing a more generous clustering.

The compare check-box marks a cluster for comparison. If it is checked, the box it self, without the text, stays always visible to allow the user to easily see which clusters are marked for comparison and uncheck them quickly again.

Clicking the ”Inspect”-Button moves the user one layer down the cluster tree and displays the children of the this cluster.

The variance field displays the variance found inside of this cluster. It helps the user in deciding which clusters are worth exploring further and which ones should be left alone. A cluster with only very little variance has very homogeneous plots, which are very similar to the median, which is already presented as the image of the cluster. Therefore the user does not need to explore this cluster any further. If a cluster has a high variance the opposite is true and the plots are very heterogeneous. This means that this cluster is potentially worth exploring.

Colour encoding was used to give the user an immediate intuition for the meaning of the variance. The colour has five levels of saturation, ranging from grey to red. Grey maps to a low variance, being a ”boring” colour and not being very noticeable. It is

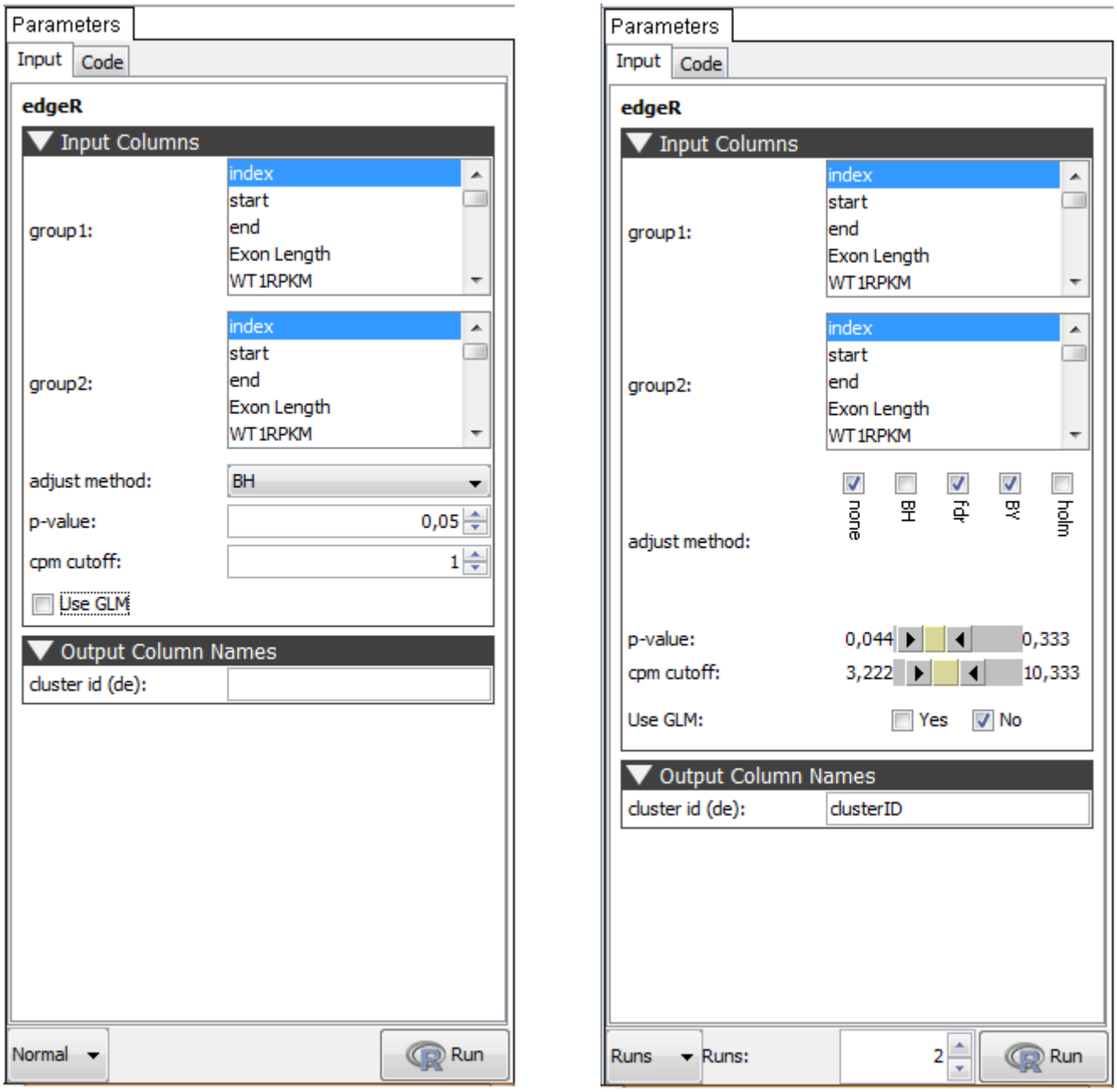


Figure 1: Parameter panels: On the left for a single set of parameters, on the right for ranges.

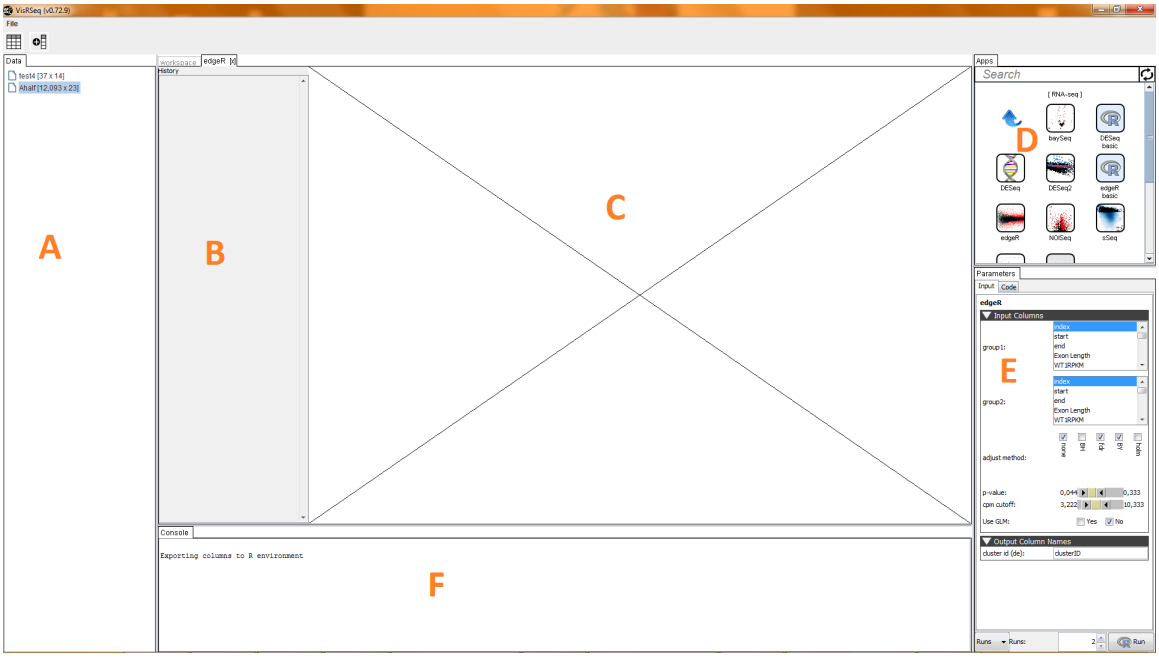


Figure 2: Program while entering parameter ranges. A: Input, B: History, C: Workspace, D: Apps, E: Parameters, F: Console

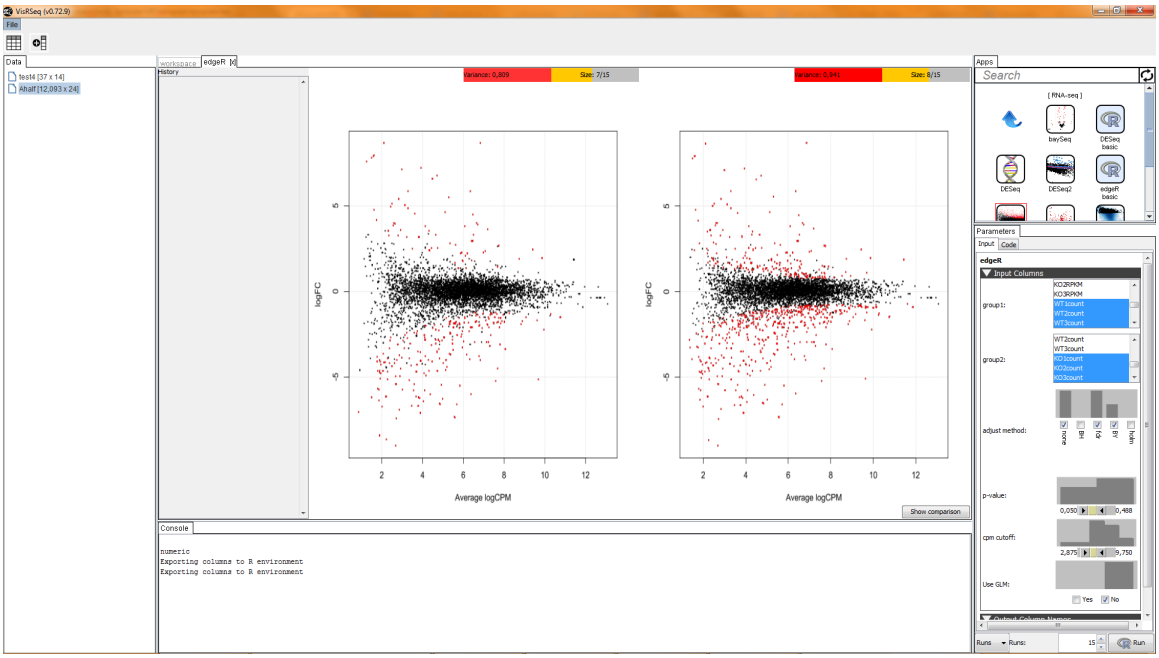


Figure 3: Program displaying results.

supposed to not make the user interested in it and not get their attention, while also providing the exact value of the variance in this cluster. Red is used to display extreme variance and guide the user to further explore such a cluster.

The last element displays the size of the cluster in comparison to the size of its parent cluster. Size is measure by the amount of plots a cluster has. A bar is drawn to show how much percent the child cluster's size takes of the parents total size. On top of that the actual values are written. The background of the bar is light grey, like all backgrounds, while the filled-in part is yellow, like the all other elements that are linked to selected individual clusters.

4.3 Parameter Distribution

Figure 4 shows the parameter panel, after the program ran and produced output. The distribution of all parameters used to generate all output is shown as grey histograms. The values next to the slider beneath the histogram show the minimum and maximum value used to generate the output. This might differ from the values entered by the user, because the program only samples the parameter space and does not explore it exhaustively. Hovering over a bar of a histogram displays its x and y value.

If a cluster is selected an additional histogram is shown above superimposed on the histogram displaying the distribution of all parameters. The new histogram shows the distribution of the parameters used in generating all plots that are contained in the selected cluster. The double sliders, previously showing the range of the whole distribution now only show a single value. This is the value that was used to generate the specific plot that was chosen to represent the selected cluster.

The color gray for the general distribution was chosen to make comparison with the selected cluster easier, which has a very saturated and bright colour. It also is supposed make the user associate it with something in the background, since the histogram's background colour itself is a lighter gray. This helps to further distinguish it from an interactable object.

4.4 History

After a the user inspects a cluster and therefore goes to the next level of the graph to view its children a history node will be created and attached to the history panel, see figure 5. This node displays the children of the parent cluster, or in other words: exactly the view the user saw before inspecting a cluster. The cluster that was chosen for inspection will have a yellow border to show the user from where they came. They can go down additional levels of the tree and for each level one node will be created.

The nodes do not only provide the user a view of their history and the path that they took exploring the data. Users are also able to quickly navigate to the desired level of the tree by clicking on the corresponding history node.

4.5 Comparison

Clusters which are marked for comparison can be compared by clicking the "Show comparison" button. This shows a view to the user only consisting of the clusters which were marked for comparison. The clusters are arranged in a as square as possible grid to allow the user easy comparison. Using the "Back" button gets the user back to the view they were previously using.

4.6 Performance

The program keeps being highly interactive, even with big amounts of data. Generating the plots takes a long time, which is expected, but could be improved. The similarity measure is not optimally calculated yet, since some redundancies occur and some values could be saved and kept for future calculations.

The generating of the plots lends itself perfectly to prallelization, which would provide a significant performance increase.

4.7 Evaluation

Since the program is not yet complete and does not support all required functionality, no real evaluation was performed yet. The frequent meetings with the collaborator provided significant feedback, which was used to continually adopt and improve the program.

One major addition due to feedback was the effort to more clearly convey to the user which parts of the program are interactable and which are not. This was mostly done by providing small borders around objects as soon as the user hovers over them or display them in a certain colour.

The presentations in class were also great feedback and helped to get the project into the right direction, especially during the prototype phase.

The evaluation process will start as soon as possible to get feedback from the actual users.

5 Implementation

The program is written in Java using mainly swing elements for its layout. R was used for the plots and Rserve as a link between Java and R.

Getting into the code and learning to understand it proved to be the most difficult task. This is due to the program still being under heavy development and therefore providing very little documentation and partially dead or unintuitive code. This is not a direct criticism but more due to the nature of code that is unfinished. Given the scope of the project it took considerable effort to actually start working and to figure out what does what.

6 Discussion

6.1 Strengths and Weaknesses

Major drawbacks of this visualization include the poor scalability of the history nodes. Those are smaller versions of the workspace at a specific time. If many clusters are shown at the same time a smaller version of this could make them too small to decipher.

The general presentation is also plagued by this problem. Too many clusters at the same time leads to very small plots.

Both of these issues can be combated by forcing the program to do more lenient clustering and therefore having less plots visible at the same time.

An other issue is the ugly design of the interface elements and a general lack of aesthetics for the whole program. Those issues will be addressed as soon all features are implemented.

A strength of this tool is that users can quickly figure out the distribution of the parameters used, due to scented widgets. It also allows users to quickly find interesting clusters to explore, because of the variance and size display of each cluster.

6.2 Lessons Learned

This was the first time I had to work with code provided by someone else and to build on top of completely unknown architecture. The time investment can be massive to get started, depending on the size of the already existing code base.

"Fail faster" truly is an important mantra. Getting feedback as quick as possible is incredibly important, because some visualization techniques I thought were great turned out to be completely useless or overshadowed by techniques I just designed as a back up, but the users really liked.

The importance of inspecting existing work flows and trying to improve them to make them more efficient. Using VisRSeq without parameter space analyses can be for some use-cases be incredibly inefficient, but providing the program with a parameter space and the user with many plots to explore at once hopefully proves to be way more efficient.

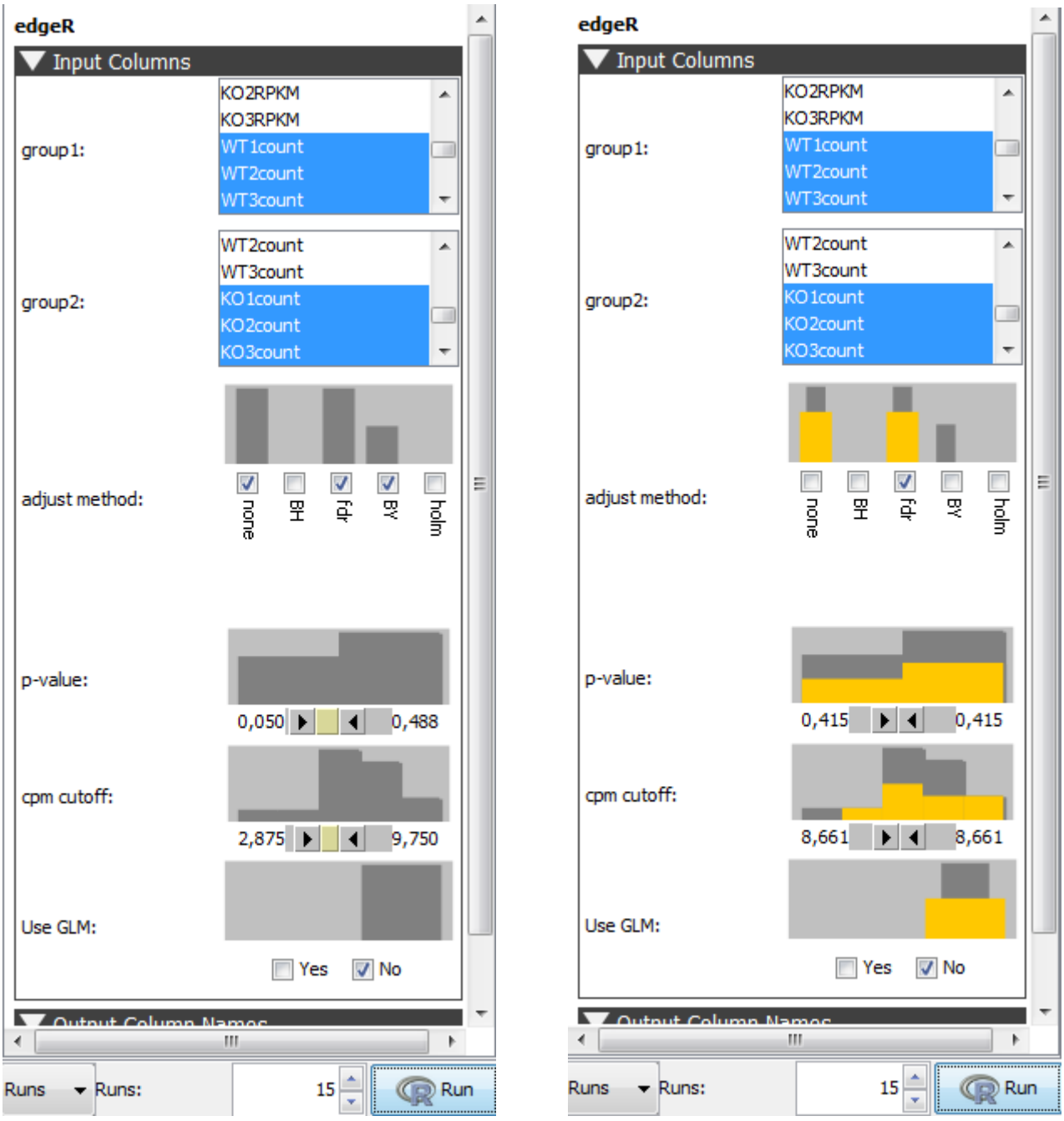


Figure 4: Parameter panels after all runs were finished: On the left no cluster is selected, on the right a cluster is selected.

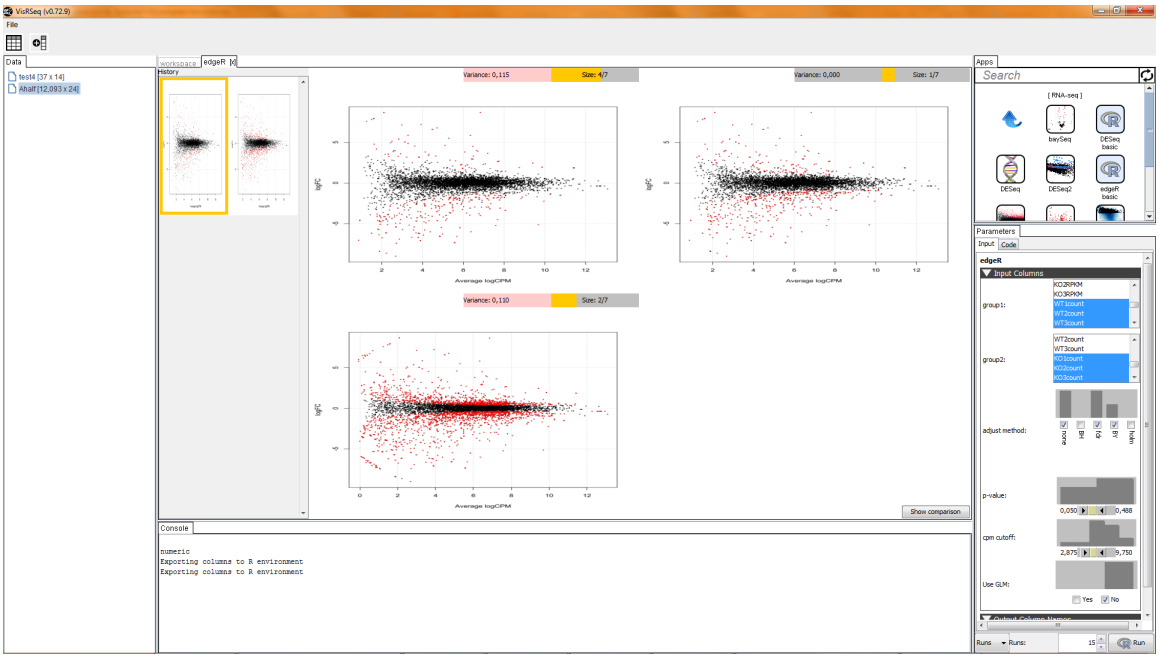


Figure 5: Program showing results after going down one level of the tree and saving the history.

REFERENCES

[1] M.-F. Balcan, A. Blum, and S. Vempala. Clustering via similarity functions: Theoretical foundations and algorithms, 2008.

[2] C. Blundell, Y. W. Teh, and K. A. Heller. Discovering non-binary hierarchical structures with Bayesian rose trees. In K. Mengersen, C. P. Robert, and M. Titterington, editors, *Mixture Estimation and Applications*. John Wiley & Sons, 2011.

[3] M. Bruls, K. Huizing, and J. van Wijk. Squarified treemaps. In *In Proceedings of the Joint Eurographics and IEEE TCVG Symposium on Visualization*, pages 33–42. Press, 1999.

[4] M. Gleicher, D. Albers, R. Walker, I. Jusufi, C. D. Hansen, and J. C. Roberts. Visual comparison for information visualization. *Information Visualization*, 10(4):289–309, Oct. 2011.

[5] Y. Hamid, M. Torsten, C. L. Matthew, M. K. Mohammad, Steven, and J. M. Jones. Visrseq: R-based visual framework for analysis of sequencing data. In *5th Symposium on Biological Data Visualization*, 2015.

[6] T. Munzner and a. Maguire. *Visualization analysis and design*. AK Peters visualization series. CRC Press, Boca Raton, FL, 2015.

[7] J. Schmidt, M. Groller, and S. Bruckner. Vaico: Visual analysis for image comparison. *Visualization and Computer Graphics, IEEE Transactions on*, 19(12):2090–2099, Dec 2013.

[8] M. Sedlmair, C. Heinzl, H. Piringer, S. Bruckner, and T. Möller. Visual parameter space analysis: A conceptual framework. *IEEE Transactions on Visualization and Computer Graphics / Proceedings IEEE InfoVis 2014*, 20(12):pp. 2161–2170, 2014.

[9] M. Sedlmair, M. Meyer, and T. Munzner. Design study methodology: Reflections from the trenches and the stacks. *Visualization and Computer Graphics, IEEE Transactions on*, 18(12):2431–2440, Dec 2012.

[10] G. Torres, R. Basnet, A. Sung, S. Mukkamala, and B. Ribeiro. A similarity measure for clustering and its applications. *Int. J. of Electrical, Computer, and Systems Engineering*, 3(3), 2009.

[11] W. Willett, J. Heer, and M. Agrawala. Scented widgets: Improving navigation cues with embedded visualizations. *IEEE Trans. Visualization & Comp. Graphics (Proc. InfoVis)*, 13:1129–1136, 2007.

[12] Y. Yang, F. Liang, S. Yan, Z. Wang, and T. S. Huang. On a theory of nonparametric pairwise similarity for clustering: Connecting clustering to classification. In Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, and K. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 145–153. Curran Associates, Inc., 2014.