BioVis Parameterspace exploration of Parameters in force directed RNA secondary structure visualization.

Verena Grilnberger (a1306379) Theresa Frühwürth (a0703052) Viktoria Mayer (a1304613)

1 Introduction

Forna is a tool for RNA secondary structure visualization. The aims of secondary structure visualizations are to generate consistent visualizations minimizing certain artifacts that are commonly introduced by current RNA secondary structure visualization algorithms. These artifacts include but are not limited to overlaps, stretches and positioning errors in the force directed layout visualization implemented by Forna. Scientific RNA secondary structure visualizations aim to produce visualizations that are both consistent with the scientific standard and reproducible, such that the interpretation of such structures is facilitated. Forna can be seen as an improvement over historical visualization algorithms for example the one proposed by [8]. Forna heavily builds on this idea, however the force directed layout used in Forna introduces parameters that help alleviate overlaps by pushing structures apart using a repulsion term that decays with the square of the distance ([9]). Regardless of the improvement visualizations generated by Forna still suffer partly from artifacts introduced by the force directed layouts tendency to get caught in a local minimum. This problem is primarily encountered in larger structures, however it even smaller structures can be suffering from these drawbacks. The force directed layout furthermore suffers drawbacks because of the non deterministic nature, which leads to the issue that small changes in the initial starting conditions, can lead to disproportionally large differences in the final layout. It follows that this works major purpose was to find parameters that minimize artifacts usually introduced and help find parameter combinations that generates consistent RNA secondary structure visualizations.

2 Users

Users of this application are biologists and chemists who deal with RNA sequence data and use Forna to generate secondary structure visualizations. In order to avoid the unwanted characteristics previously mentioned the target application should help to find a combination of input parameters which leads to a good visualization of the input sequence. These parameters should then be used in the Forna interface that can be found online here http://nibiru.tbi.univie.ac.at/forna/, such that post processing efforts of the scientists are minimized. This would usually take a disproportionally long amount of time, and we hope that the parameter combination found with our tool can provide a remedy or at least reduce problems encountered while using Forna.

3 Tasks

Exploring the parameter space: The tool should provide a means to systematically explore the dependence between different input parameters in FORNA and the visualizations it produces. As scientists are interested in the best solutions it should provide an overview of the best groups of visualizations and the parameters that determined these.

Find optimal parameters: The user should be able to find optimal parameters for a particular sequence length, by being able to filter the data and look at particular subregions of the space.

Exploring Dependencies between parameters: The user should be able to explore dependencies between different subsets of parameters.

Zoom: Not only through filtering but also through clicking on cells of the matrix the user should be able to view details of the visualization such as the related image, the parameters used to generate the visualization and penalty scores associated with that particular visualization. I.e. if the user wants to explore why nodes in the graph layout are located close together, he should be able to explore this in a detailed view.

4 Data

4.1 Preprocessing of the raw data

Our project handles a variety of Data at the different steps of our tool. Firstly in order to find measures for the quality of certain secondary structure representations that are generated by Forna we needed to simulate a large number of such visualizations on the same structure. We'd like to emphasize the fact that there is a large number of structure sizes, and the tool is meant to explore combinations that would be set as defaults in Forna. Hence we decided to develop a tool that helps the developers of the application at TBI to find such default values and provide them within their application. Originally this project was developed for analyzing one single RNA structure, but the user is able to simulate data with more RNA structures of different sizes, then the size would be an additional parameter. For the simulation headless chrome was used, after initial memory problems using phantom.js.

Penalty values were calculated with a C++ program. First backbones are read in as vector lines, taking into account the translation of the svg. Position penalties i.e. penalties for the visualization not being contained in the svg are computed by adding a score of 1 each time a node is located below the minimum x or y or above the maximum x or y value of the size of the svg size. Overlaps are incremented by 1 if any of the vector lines intersect. Because stretches are elongated links in the backbone backbone and usually links are set at quite regular intervals between the nodes stretches are defined as links that exceed the minimum link length by 2 times the minimum length. This is heuristic and probably there are more appropriate ways to compute this penalty value, however due to time constraints we could not implement better penalty computations.

After the simulation step we yielded data of three penalty values and the initial parameter settings associated with them as well as the corresponding visualization. This implies that the data we used further was numerical except for the image of the visualization.

4.2 Clustering

After the preprocessing step, we needed to cluster the data. Generally we cluster over the penalty values to find visualizations that are similar in their drawbacks and benefits i.e. similar values of all penalty values will yield visualizations to be in the same cluster

Table 1: Table 1 shows results for DBScan and hierachical clustering respectively.

Results non parametric clustering	
Silhouette Coefficient:	0.317
Silhouette Coefficient:	0.854

because of their neighborhood in the space. Density based clustering was chosen because the data showed a non gaussian distribution (Figure 1) which implied that a non parametric method would be necessary for clustering. Furthermore standardization of the penalty values was necessary as there were values for stretches that differed from the other penalty values by 3 orders of magnitude. Thus to facilitate the clustering we squareroot transformed the features.

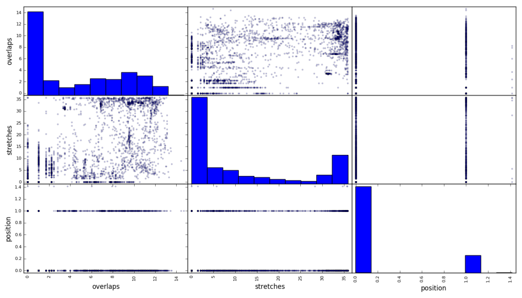


Figure 1: Shows the distribution of the penalty variables.

Hierachical clustering an alternative (Table 1.) that we considered did yield a better results than DBScan with the data. However the hierachical clustering procedure yielded only 2 Clusters, which was not a sustainable result considering that we had simulated a multitude of parameter settings during the penalty calculation and the three dimensional feature vector we clustered over showed more variation. Hence we decided to proceed using DBScan as a clustering algorithm and implement a functionality to automatically choose a good value for the neighborhood range parameter Epsilon. The optimal value for the silhouette coefficient is 1, hence our result of .317 is quite low and could indicate that the structure found was introduced artificially. One can also see that the data has areas of differing density which DBScan has allegedly problems handling, so OPTICS might have been a better choice but could not be implemented anymore due to time constraints.

Epsilon (eps) is a parameter that defines the neighborhood range i.e. the radius that is searched for Minpoints using euclidian distance in order to form a cluster or find density connected regions. A sensible value for eps can be chosen by looking at the nearest neighbor distances of all points to all other points. For eps to be appropriate we should pick a value for the nearest neighbor distance that includes most of the dataset such that one has a small number of outliers. Because we are querying the dataset on itself for the nearest neighbor distance, the minimum distance of each point will be the distance to itself and the nearest neighbor parameter has been set to 2 such that one can find the distances to all other points. By creating a histogram from the distance data one yields an array with frequencies for the bins, we chose 0.1 for the bins.

4.3 Server sided integration

Our tool furthermore provides a flexible way of choosing parameters. For the userinterface that leads to the clustering the option was

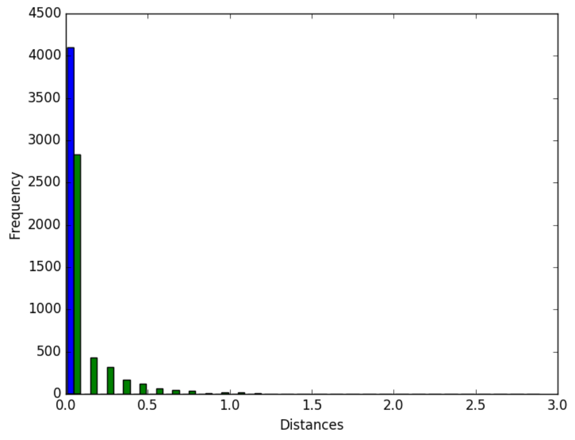


Figure 2: The histogram shows the nearest neighbors binned into bins with stepsize 0.1

implemented to choose the variables that the user wants to use as input features for the clustering algorithm. This was implemented using knockout.js a flexible javascript library that enables such tasks. The variables chosen to cluster over, are passed as a json object using an ajax call (jquery) which passes the variables to the flask application which in turn is calling the python script to cluster the data, passing the variables as a list.

Furthermore communication between the interface and the clustering file was implemented using flask by building a Rest Server. Flask enables to send json objects to scripts that run in the background. Static files (e.g. the javascript files associated with our visualization) are served via flask as well, and redirection from one step to the next happens automatically through javascript functions communicating with the server through flask and passing arguments to the python script which runs on the terminal in the background.

5 Implementation challenges

The biggest implementation challenge was that we neither knew about server sided integration and Restful APIs nor d3.js which made the work for this course difficult as the learning curve was extremely steep. The biggest issue was the time constraint and the expectation for a fully integrated app (server sided integration) as well as the preprocessing steps of the data i.e. penalty calculations (datageneration) and clustering as opposed to a visualization which by itself is already an extensive task when using D3. All these steps took extensive amount of time. Our code grew large and involved very quickly through the many responsive elements, which made it difficult to provide seemingly easy functions such as axis labels as very minor changes influenced other parts of the visualization immediately. Moreover, one major difficulty was also to get everything on one screen since we had a lot of visualizations and options.

6 Task Solutions

This section provides a reasoning of how we achieved the different tasks and why we chose particular designs.

Exploring the complete parameter space for Forna: The barchart should give an overview of the best clusters, and in combination with linking and brushing can give an overview as well as more specific insight. Similarly the scatter plot matrix can be used as an overview, but different functionalities enable the user to explore the data on different levels.

Finding optimal parameters by filtering: Through the filtering implemented in the scented widgets the user can select different subspaces and parameter settings which will reflect in the scatter plot matrix through brushing. This enables the user to filter for low values of particular penalties. Furthermore the user can decide if a particular parameter is particularly detrimental to the users RNA visualization and weigh the penalties accordingly.

Exploring Dependencies between parameters: Because changes in filtering apply to the scatterplotmatrix reducing the number of datapoints, it enables the user to see the influence of limiting multiple parameter values and explore the effects on visualizations. Deciding on the x axis and y axis for the scatterplot enables an enlarged view of the dependencies between the selected parameters and penalties or alternatively between two parameters or two penalties.

Zoom: In each part of the visualization the user can explore contents in more detail. Zooms enable the exploration of particular dependencies in more detail. Hovering over the dots in the scatterplot shows details on the parameter settings and penalty values of a particular visualization. Clicking on a tile in the scatter-rectangle-matrix shows a detailed view of the Forna visualization in a small window to the left of the main window. By clicking on it a pop up window enables a detailed exploration of the corresponding svg image.

7 Usability Tests and User feedback

We assessed the usability of our app at an intermediate stage to incorporate changes that the user felt would be useful to improve the ease of use.

We surveyed 4 Persons, among them 2 members of TBI, one student and one post Doc. Furthermore a student of scientific computing and a biology student were asked for their opinion. The testpersons agerange was 22 to 31. We have sorted the feedback in a way that feedback for a particular part of the visualization will be mentioned in groups. Furthermore we explain which changes we implemented in the subsequent improvement phase. Furthermore weaknesses that are not addressed could also be counted towards our discussion of the weaknesses later on, but in order to not repeat anything these weaknesses are mentioned here exclusively.

Similar tools were only known to one testperson. The goal of testpersons was to find the best visualization using the tool, but also to find patterns in the data to identify good parameter settings.

Clustering: Clustering was clear for most testpersons, but it was critisized that we should cluster over the parameters rather than the penalty values. The granularity of the clustering was critisized and now the selection of the best clustering result and corresponding number of clusters is available.

Barchart: The color coding was not recognized by the test persons, and we implemented the color coding by coloring corresponding values in the tooltip accordingly. The tooltip furthermore was a bit confusing bevcause it was unclear what was referred to as node, while we mean a datapoint (i.e. visualization), it could also refer to the nodecount in the sequence. Furthermore the tooltips are sometimes overlapping with the bars such that it makes inspection of both the tooltip and the bar difficult, this could be solved by showing the tooltip on a fixed position, however this also implies that the connection between a particular group of bars and the corresponding tooltip is not visually encoded anymore.

All testpersons said that the x axis needed a label, which we included later on, but was not yet implemented at the time of testing due to difficulties in the implementation.

2 testpersons could not identify the correspondence between the different y axis and the bars heights. The stacked layout was not liked by one test person. Testpersons also did not like the colors used mentioning that the colors were too loud and confusing because of the repetition of the colors in the scatterplotmatrix and the bars of the barchart, this was changed in our final visualization. Furthermore we implemented the percent value in the tooltip after cirtisizm that one should be able to identify which portion of the complete data one looks at.

Heatmap scatterplot matrix: Labels were not readable on smaller screens. The color selection was critisized for both not being clear on the color coding and not being appropriate for color blind users. This led us to provide two additional color sets. Maximizing one element of the scatterplot matrix was difficult for most users, as the drop down elements were not implying the use for maximization, so we added a small visual element helping users to maximize the particular subplot.

Scatterplots: After cirtisizm of overlapping datapoints, we jittered the data randomly around the value such that overlaps are minimized as much as possible. The color coding was also critisized but we kept the color coding, because in face if there are less clusters the color coding is unequivocal, and the confusion arises only because of the granularity of the clustering. It was suggested that we could implement tooltip values that show how many values are overlapping at one position. Only one testperson said it would be good to implement a functionality to access the values of a specific datapoint, this was implemented after the usability test.

Weighting of the penalties: Most users did not find the functionality obvious, and were not certain how the weighting of the penalties was used. One test person mentioned that negative weights could be useful as the weights were confined to be between 0 and 1 before the test, negative weights were made possible in the final version.

Parallel coordinates: Parallel coordinates were unintuitive for most testpersons. This corresponds to the ongoing discussion about the usefulness of parallel coordinate charts, that are arguably complicated and involved the more lines are present. Some users mentioned redundancy of the parallel coordinates, that encode the same information as the scatterplot matrix. It was implied that brushing and linking would be important to highlight corresponding elements in the scatterplot matrix, thus this was improved in the final version.

Scented widgets : Scented widgets were initially not recognized to be a filter option, and the correspondence between barcharts and filtering was not clear. One testperson preferred a table instead of the scented widgets in which datapoints could be selected in a checkbox.

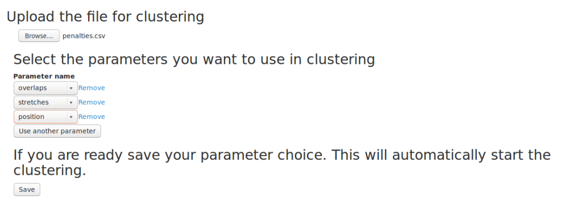
General: One testpersons mentioned that tooltips could appear more immediate. Testpersons with insufficient knowledge of the data had difficulties interpreting the visualization. Summarizing the critisizm all users seemed some time to come to grips with the visualization, but eventually found the tool to be useful. Because the tool was tailored for the specific application not all users found it to be useful.

8 Visualization implementation details and scenario of use

Figure 3 shows the userinterface that lets users select the features for the clustering. These features are passed on to the python file and clustering is simulated. We will ommit the "best clustering selection" step because it is covered in the database report.

The user will be able to upload the file on the uploadscreen (Figure 4) by browsing the folder resultsclustering the user can select the file of his choice. Upon clicking on the upload file only csv files are shown within the directory. The user should browse to the results file. After which he is prompted to input the number of penalties and the path to the SVG files, if no SVG files are present the user should type no or n and remove this function.

The initial screen will show a barchart (Figure 5) of the 20 best clusters sorted by the their penalty values. The best cluster is shown first. Hovering over the bars will give you detailed information on the content of the selected cluster in a tooltip. The color coding explains the content of a particular bar. Furthermore the number



Upload the file for clustering

Browse... penalties.csv

Select the parameters you want to use in clustering

Parameter name

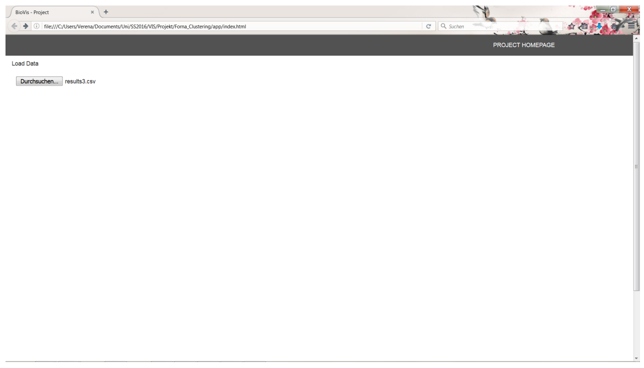
overlaps	Remove
stretches	Remove
position	Remove

Use another parameter

If you are ready save your parameter choice. This will automatically start the clustering.

Save

Figure 3: Userinterface for feature selection.



BioVis - Project

file:///C:/Users/Verena/Documents/Uni/SS2016/VIS/Projekt/Forma_Clustering/app/index.html

PROJECT HOMEPAGE

Load Data

Durchsuchen... results3.csv

Figure 4: Uploadscreen for uploading the file.

of datapoints contained in the cluster are shown in lilac. The default ordering of the scatterplot matrix shows the best datapoints on the front. Green datapoints are better and red data points are worse visualizations. As will be later described in the options menu you could choose to look at the worst datapoints. However these datapoints are mostly outliers, this funcitonality was implemented due to the large amount of overlapping datapoints such that one could explore the best and optional if needed the worst datapoints that are produced by a particular combination of input parameters. This might be useful for the user to understand the limitations of a parameter combination. If only the best results would be shown, he could not easily explore these limitations. Our usability studies also showed that users wanted to keep track how many datapoints were inside a cluster relative to the whole dataset, such that we implemented a percent value in the tooltip to give the user a chance to put the results in perspective with respect to the whole dataset.

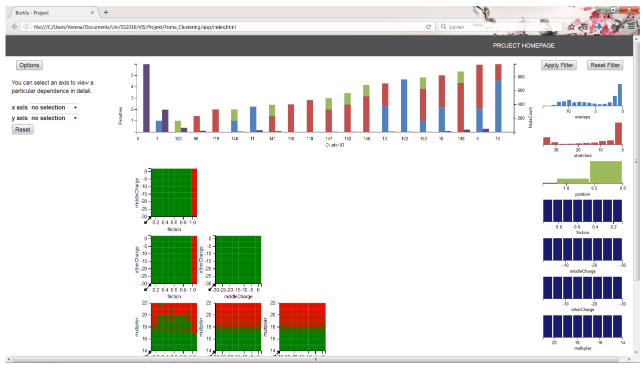
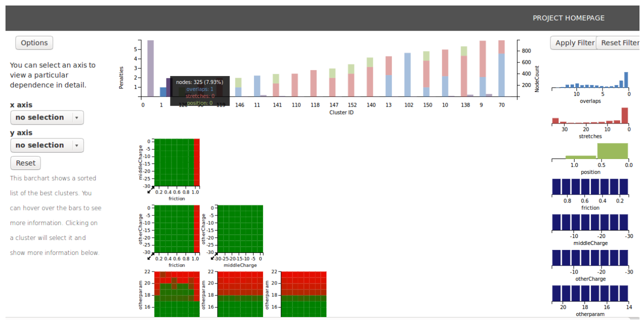


Figure 5: Initial visualization of the clustering results.



PROJECT HOMEPAGE

Options

You can select an axis to view a particular dependence in detail.

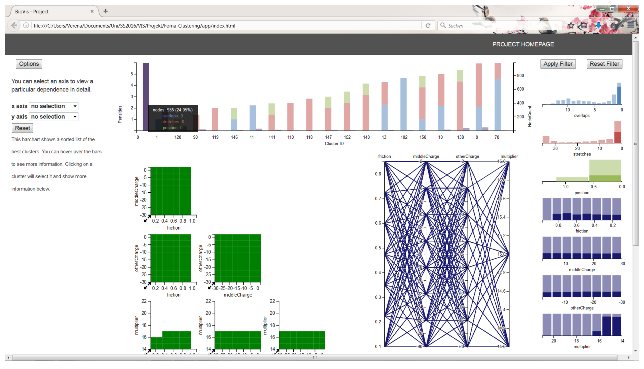
x axis no selection

y axis no selection

Reset

This barchart shows a sorted list of the best clusters. You can hover over the bars to see more information. Clicking on a cluster will select it and show more information below

Figure 6: Initial visualization of the clustering results.



You can select an axis to view a particular dependence in detail.

x axis no selection

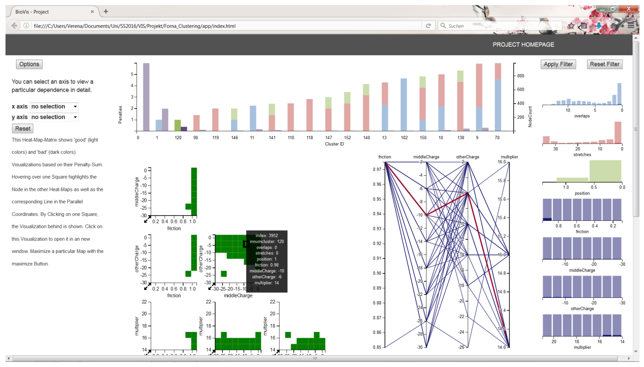
y axis no selection

Reset

This barchart shows a sorted list of the best clusters. You can hover over the bars to see more information. Clicking on a cluster will select it and show more information below

Figure 7: Initial visualization of the clustering results.

There is also a functionality to zoom into one element of the scatterplotmatrix. This is useful if many matrizes look quite similar and if the dataset would pertain to many different parameter values. From the perspective of the scenario of use, one can see that if one selects the results for the best cluster by clicking, that values of 14 to 16 for the other parameter should be choosen in order to get visualizations with 0 penalty values in all 3 penalties. This follows from the reduction of the scatteplotmatrizes values to the values contained only in the selected cluster. Furthermore the parallel coordinates pop up and show the distribution of parameters in the selected cluster. Hovering over a value in the matrix will highlight the corresponding line in the parallel coordinate diagram.



You can select an axis to view a particular dependence in detail.

x axis no selection

y axis no selection

Reset

This Heat-Map-Matrix shows 'good' (light colors) and 'bad' (dark colors) Visualizations based on their Penalty-Sum. Hovering over one Square highlights the Node in the other Heat-Maps as well as the corresponding Line in the Parallel Coordinates. By Clicking on one Square, the Visualization behind is shown. Click on this Visualization to open it in an new window. Maximize a particular Map with the maximize Button.

Figure 8: Highlighting the datapoint in the parallel coordinate chart.

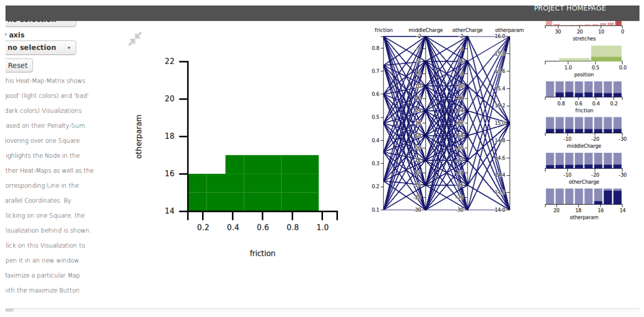


Figure 9: Maximizing the matrix.

Figure 10 shows the functionality that enables the user to inspect the RNA structure visualization by clicking on a datapoint in the heatmap or in the parallel coordinate chart. Clicking on the structure in the pop up window on the left in Figure 10 will direct the user to a new window where he can inspect the structure in more detail and use zoom to inspect the structure closely.

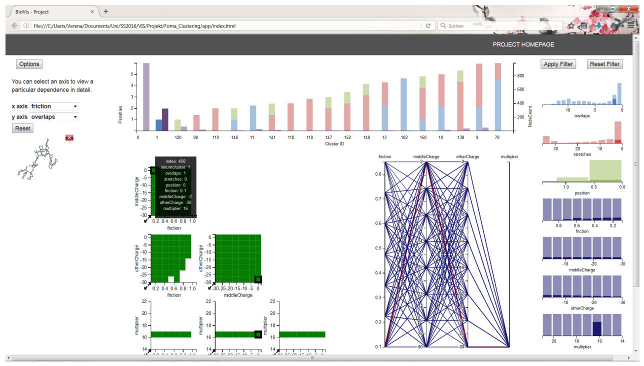


Figure 10: Detailed view 1.

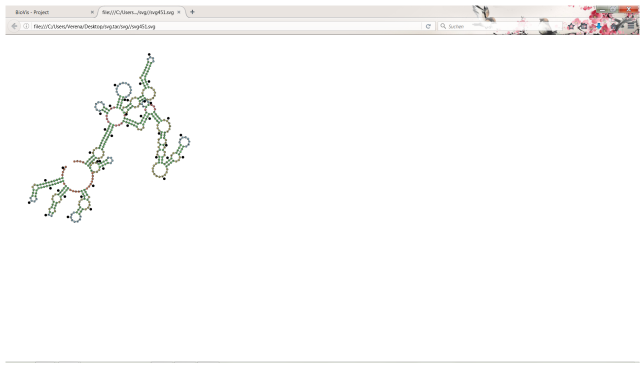


Figure 11: Detailed view 2.

If the user wants to filter according to one or more penalty values he can use the scented widgets and apply the filter, which acts on the whole dataset such that all subplots will be changed accordingly. The scented widgets show the distribution of the penalties and parameters, because of the nature of our data simulation our data on parameters is obeying a uniform distribution. This may help him to constrain the data further and only inspect clusters with low values on each penalty, such as shown here. The scented widgets can be inspected by hovering over them. A small tooltip will show the particular frequency value of a bar. Another axis would have potentially confused the user thus we decided to minimize the data ink ratio while first looking at the data.

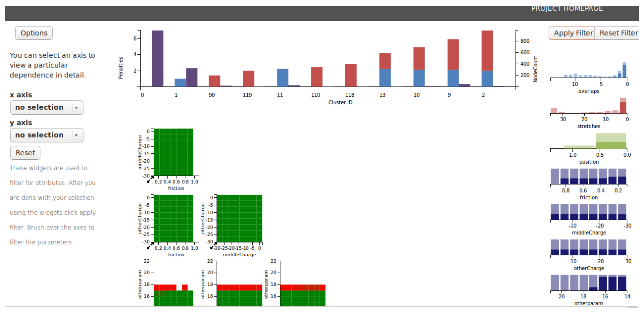


Figure 12: Filtering using the scented widgets.

If one wants to look at the dependency of a particular parameter in detail, the user could select a penalty value and a parameter. Here he could see that position mistakes are introduced if friction is too large. One can also see that outliers in Figure 13 cover most datapoints. If one clicks the option exclude outliers on the left outliers are removed. We introduced random jitter to alleviate the problem of overlapping points in the scatterplot. Furthermore clicking on the datapoint will pop up the structure corresponding to the datapoint as well.

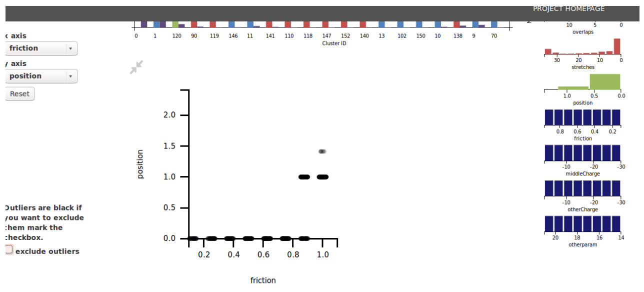


Figure 13: Scatterplot to view a dependency.

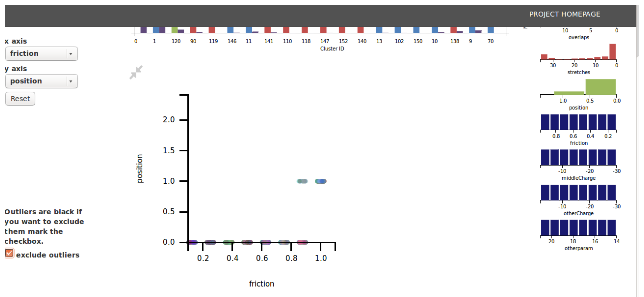
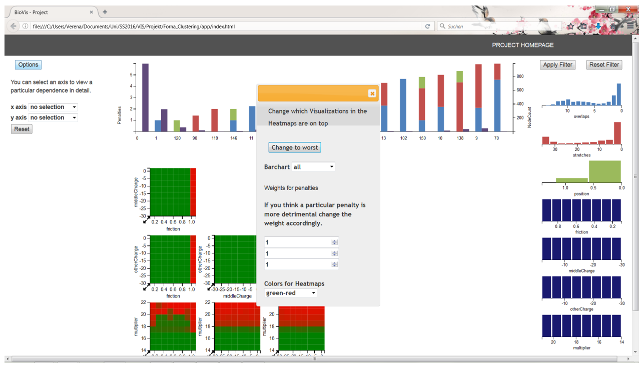


Figure 14: Scatterplot to view a dependency with outliers removed.

We also gave the user the option to select weights for the penalties, if he deems particular errors more detrimental to the visualization. Giving negative weights will show you the worst clusters as shown in Figure 17. Similarly the change to worst option will show the worst datapoints in front (but not the whole cluster it follows that Figure 16 and 17 are distinct). He can furthermore choose to only show a particular penalty in the barchart. The stacked barchart is helpful to view the relative values of different dimensions.

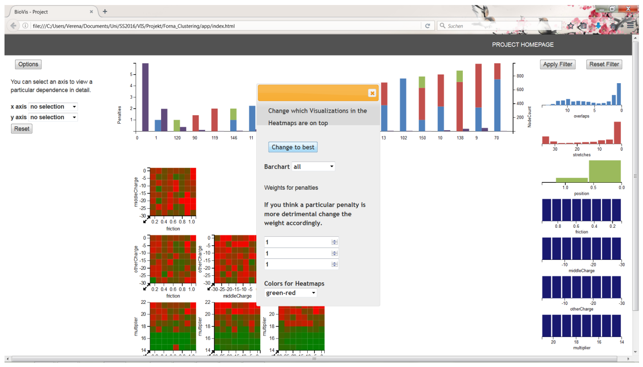
But in order to compare only one dimension it helps to remove the dimensions that are not currently under investigation by setting a penalty weight to 0 or if only one dimension is needed selecting it in the Barchart dropdown menu. Furthermore this image shows the option to choose colors other than green and red for color blind users.



You can select an axis to view a particular dependence in detail.

If you think a particular penalty is more detrimental change the weight accordingly.

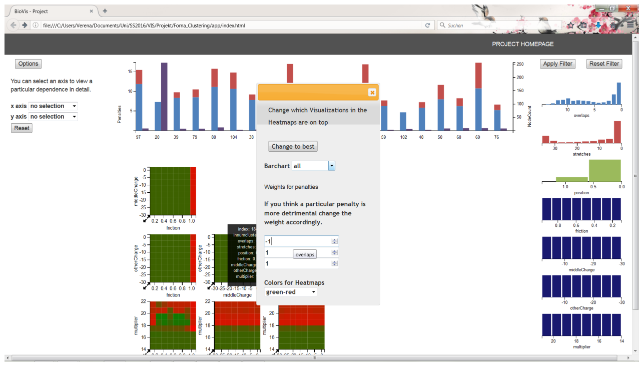
Figure 15: Options



You can select an axis to view a particular dependence in detail.

If you think a particular penalty is more detrimental change the weight accordingly.

Figure 16: Change to worst option

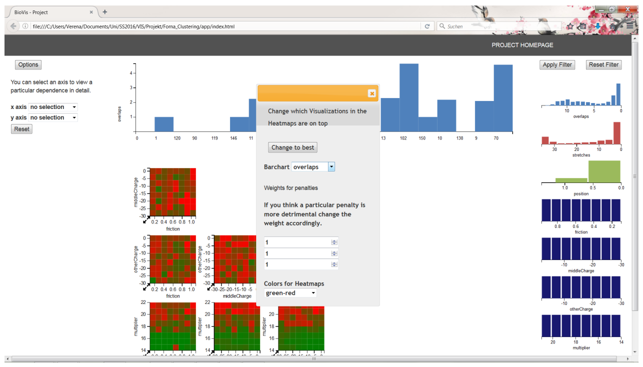


You can select an axis to view a particular dependence in detail.

If you think a particular penalty is more detrimental change the weight accordingly.

Figure 17: Weights option.

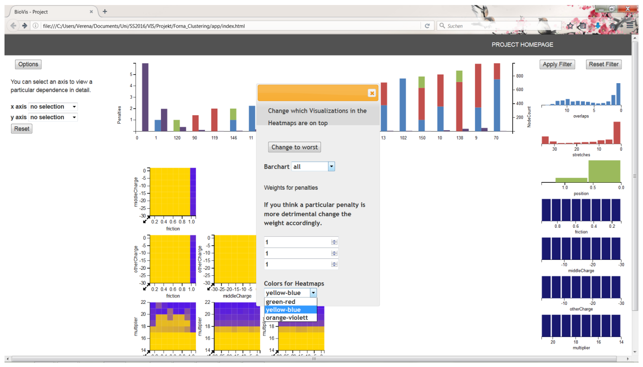
Going back looking at the best cluster, one can also see that the selection is represented in the scented widgets as well. The parallel coordinate chart highlights the selected datapoint and shows detailed information. This was implemented to facilitate navigation in the parallel coordinate chart. Furthermore the option of selecting a subspace in the parallel coordinate was implemented, and the



You can select an axis to view a particular dependence in detail.

If you think a particular penalty is more detrimental change the weight accordingly.

Figure 18: Overlaps only options.

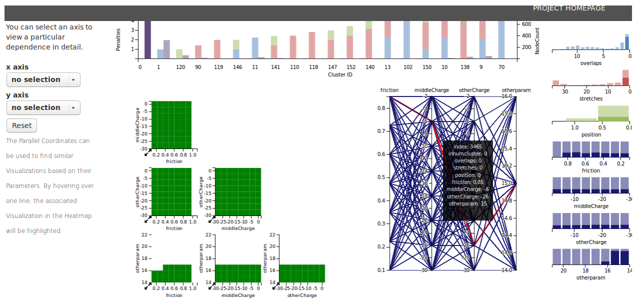


You can select an axis to view a particular dependence in detail.

If you think a particular penalty is more detrimental change the weight accordingly.

Figure 19: Option for color blindness.

user can look at particular dependencies selecting subsets of datapoints(nodes), this reduces the clutter that is present when looking at all datapoints in a cluster and facilitates following trajectories through the chart, please note that this filtering function has no effect on the actual dataset, but filters only within the subgraph (Figure 21).



You can select an axis to view a particular dependence in detail.

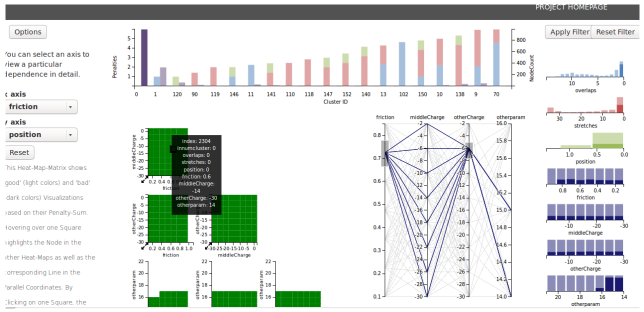
The Parallel Coordinates can be used to find similar Visualizations based on their Parameters. By hovering over one line, the associated Visualization in the Heatmap will be highlighted.

Figure 20: Selecting the best cluster and tooltip in parallel coordinate chart.

After the usability tests it was also clear that the user needed help navigating the visualizations. This was implemented by adding small help texts that explain the part of the visualization the user hovers over.

9 Lessons Learned

Since our application was aiming at providing a flexible frame work for parameter space exploration not only for a force directed visualization of RNA secondary structure but also a tool for RNA design



ou can select an axis to iew a particular lependence in detail.

his Heat-Map-Matrix shows good' (light colors) and 'bad' dark colors) Visualizations ased on their Penalty-Sum. Hovering over one Square ighlights the Node in the ther Heat-Maps as well as the orresponding Line in the arallel Coordinates. By licking on one Square, the

```
index: 2304
innumcluster: 0
overlaps: 0
stretches: 0
position: 0
friction: 0.6
middleCharge:
-14
otherCharge: -30
otherparam: 14
```

Figure 21: Selecting the best cluster and tooltip in parallel coordinate chart.

work done by the TBI group at the University of Vienna. It turned out to be of utmost importance to keep the tool flexible regarding its input and output. D3 turns out to be an excellent tool in terms of flexibility however this also comes at a cost in terms of the effort it takes to program visualizations with D3. This flexibility of the tool was important to the group at TBI in order to reuse the design for RNA design questions and make it available for public use. Secondly in the course of the project it became apparent that visualization in D3 indeed needs a lot of expertise in front end development while the preprocessing steps needed expertise in data analysis. The visualization discipline is highly interdisciplinary and this project showed exactly this aspect of this discipline to all of our groups members. The cooperation with TBI in general was pleasant, and we did not face any of the problems mentioned to us in the beginning such as for example unavailability of the Users. We believe that it is crucial for this interaction between the user and the developers to reach an understanding of the possibilities and limits and iteratively update the Users on the progress to gather feedback, which was made possible by the collaboraters as well and it presumably helped that the Users are involved in software development themselves. Finally it was interesting to see, how when the users had an image of the application, suddenly new areas of application appeared for the Users. Specifically the wish to use the tool for new datasets was mentioned. We tried to keep most of the application flexible, however the barchart and tooltips would need some revision in order to work with new datasets. Furthermore we believe that the Data was not optimal for our application, and thus the full potential of the visualization is not yet visible. However trying to load Data from RNA Design showed much more interesting patterns to explore.

10 Strenghts and weaknesses

Furthermore to address the critique of visualizing only one sequence length, our computational means were limited, and with respect to this we generated only one sequence length, it is as mentioned up to the user to generate sufficient data and use the tool for different sequence lengths. Furthermore in general clustering might not have been the appropriate approach for the nature of the data. However it might be that other data works better with our tool, and other clustering algorithms might work better.

One strength of the tool is that one could adjust the tool for other datasets by using a different clustering approach and make some parts for example the barchart handle negative values, such that it could be flexibly used for other data if some changes are implemented. Furthermore when using other data one needs to always limit the variables displayed as d3.js does not scale well for large amount of visualizations for example in the scatterplot matrix. The script will become unresponsive, thus a subset of variables should be selected for display and highly correlated features need to be excluded.

Another strength of the application is that since there is a complete integration of all steps, it should be easy to implement on a production server with minor changes.

11 References for the Implementation

For the implementation of the clustering algorithmus [10] was used as well as knockout.js and flask.

For the scatterplotmatrix [1] and [2] were used, for the barchart [3] and for the parallel coordinates [4].

Other references used are [5] and [6].

The source code is on github: [7].

References

[1] http://bl.ocks.org/tjdecke/5558084. Accessed: 2016-06-30.

[2] https://bl.ocks.org/mbostock/3808218. Accessed: 2016-06-30.

[3] https://bl.ocks.org/mbostock/3885304. Accessed: 2016-06-30.

[4] https://bl.ocks.org/jasondavies/1341281. Accessed: 2016-06-30.

[5] http://mounirmesselmeni.github.io/2012/11/20/reading-csv-file-with-javascript-and-html5-file-api/. Accessed: 2016-06-30.

[6] http://stackoverflow.com/questions/11903709/adding-drop-down-menu-using-d3-js. Accessed: 2016-06-30.

[7] https://github.com/Cricket156/Forna_Clustering.

[8] R. E. Bruccoleri and H. Gerhard. An improved algorithm for nucleic acid secondary structure display. *Comput. Appl. Biosci.*, 4(1):167–173, 1988.

[9] P. Kerpedjiev, S. Hammer, and I. L. Hofacker. Forna (force-directed rna): Simple and effective online rna secondary structure diagrams. *Bioinformatics*, 31(20):3377–3379, Oct. 2015.

[10] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.