JAguc – User's Manual

Michael Holzhauser Lars Hüttenberger Raphael Reitzig

Matthias Sperber Sebastian Wild

April 26, 2010

JAguc is an application developed for biologists using the 454 Sequencing method to extract RNA sequences. It provides fast and comprehensive tools for analyzing these sequences, including the computation of pairwise alignments, using the resulting similarities to cluster the sequences, using the *BLAST*[1] database to identify the clusters, and finally providing the user with graphical analyses of the results. It is written in Java and optimized to scale to multiple processor cores.

Contents

[1] `http://www.ncbi.nlm.nih.gov/BLAST/`

1 Introduction

JAguc is a tool helpful for analyzing samples of sequence data. It enables the user to find out what species are represented in the sample and learn about their distribution. The basic processing of sequences is done as follows. As an initial step, the sequences are read and filtered. Next, pairwise alignments are computed, and the resulting similarity values are used to put the sequences into clusters. For each of these clusters of similar sequences, one representative is determined and its species identified using BLAST. This is performed using a *best fit* approach (only known species are considered), or *max fit* (all RNA is considered). The resulting data can then be analyzed in detail.

1.1 Requirements

Because of the large number of expected sequences, as well as time-consuming calculations of pairwise alignments and clusters, high-performant computers are recommended. This includes both CPU speed (for the calculations) and RAM size (for efficient clustering, all pairwise similarity values should fit into memory). The following items are required:

- Graphical login
- Java 6
- Local installation of BLAST
- Database server (MySQL or PostgreSQL; local installation preferrable)
- As much main memory as possible (1GB is required for 30,000 unique tags, scaling to its square). *Important:* In general, Java does not automatically allow using the complete main memory. It must be told so by using the `-Xmx` command or the accompanying batch script.
- As much CPU speed as possible (JAguc scales to both speed and number of cores)
- A 64 Bit processor is recommended, because for the 32 Bit version, the usable main memory is limited to 2GB.
- As much disk space as possible

1.2 Some Philosophy

JAguc is a tool that, beyond analyzing individual samples, also enables the user to compare different analyses (with different parameters) of the same sample to one another. Furthermore, different samples can be analyzed and compared. To make this convenient, JAguc is a database driven application, hence requires a running database installation.

To start with, JAguc requires a Fasta[2] file containing the sample of sequences. These are then loaded into the database, which holds all the sequence information of all the samples that have been imported so far. For each sample, multiple analyses can be carried out. For each of these, a number of parameters can be defined (see Chapter 2 for more details):

- The minimum length of sequences of interest.
- Allowed symbols (any sequence with illegal symbols will be sorted out).
- Primers can specify a prefix (A-Primer) or suffix (B-Primer) that all sequences of interest must match. As an option, these can be omitted for the aligning step.
- The alignment score, including substitution penalties and affine gap penalties.
- The clustering threshold.
- The clustering algorithm can be customized to use minimum, maximum, or average similarity when merging clusters.

After all parameters have been set, the aligning & clustering process can be started. This may take a while, depending on the number of unique tags (i. e. different sequences) and computer performance. Estimates are displayed to give the user an idea of how long it might take.

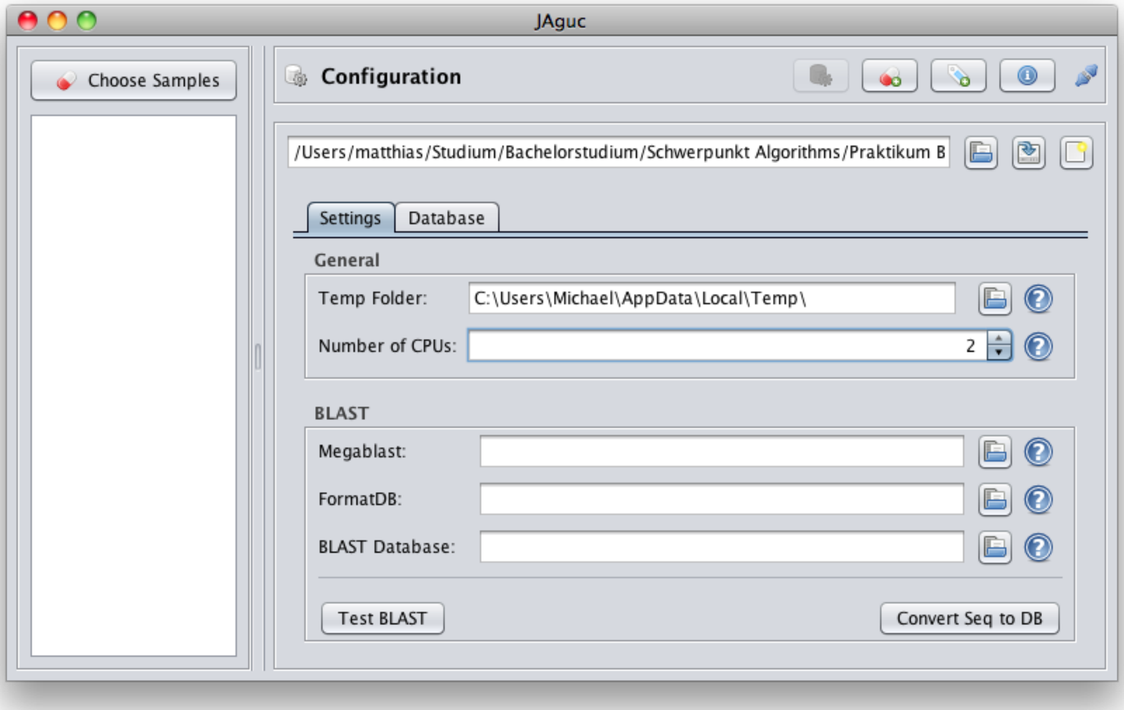
When all is said and done (and all sequences are finally clustered), several possibilities for analysis are given, including a systematics tree, sample saturation, and rank abundance. Also, several export functions are available: Graphs can be exported as images, and nodes in the systematics tree can be exported to FASTA files for further analysis.

[2] http://www.ncbi.nlm.nih.gov/blast/fasta.shtml

2 Using JAguc

This chapter provides a step by step guide on how to use JAguc.

2.1 Configuration



/Users/matthias/Studium/Bachelorstudium/Schwerpunkt Algorithms/Praktikum B

Temp Folder: C:\Users\Michael\AppData\Local\Temp\

Number of CPUs: 2

Figure 1: The Configuration Panel

JAguc starts up displaying the configuration panel. It allows to create new configuration files or use existing ones. New configurations can be saved using the according button on the right. A configuration can be customized using the two tabs:

Settings. This requires some general information:

- Temp folder: A folder in which all temporary files will be stored. A folder on a fast hard drive is preferrable.
- Number of CPUs: This allows computations (especially the aligning algorithm) to scale to computers with multiple CPUs. Cores count as individual CPUs, so, for two processors with two cores each, the number should be set to four.
- BLAST: Path to the megablast executable. Check the website for more information.

BLAST Database: Path to the BLAST database (in Fasta Format).

Convert Seq to DB: This button *must* be pressed after the BLAST database has been updated in order for the changes to take effect in JAguc.

Database. The requires specification of database information. JAguc will create necessary database tables automatically. The following information is required:

Database Driver: JAguc currently supports both MySQL and PostgreSQL databases.

Database Host: An URL to the database server. A local installation is recommended for performance reasons (in this case, put *localhost*).

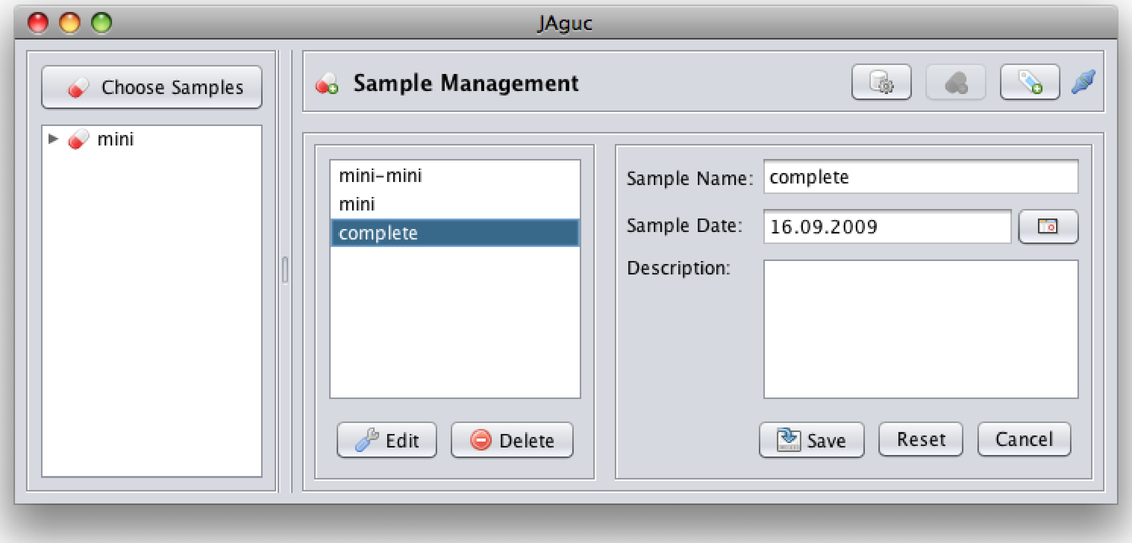
Database Port: The port under which to reach the database server. The default port is 3306.

Database Name: It is recommended to dedicate one database exclusively to JAguc.

Database User

Database Password: Can be left empty. In this case, the password will be asked for each time it is needed, instead of saving it in the configuration file.

2.2 Sample Management



JAguc

Sample Management

Sample Name: complete

Sample Date: 16.09.2009

Description:

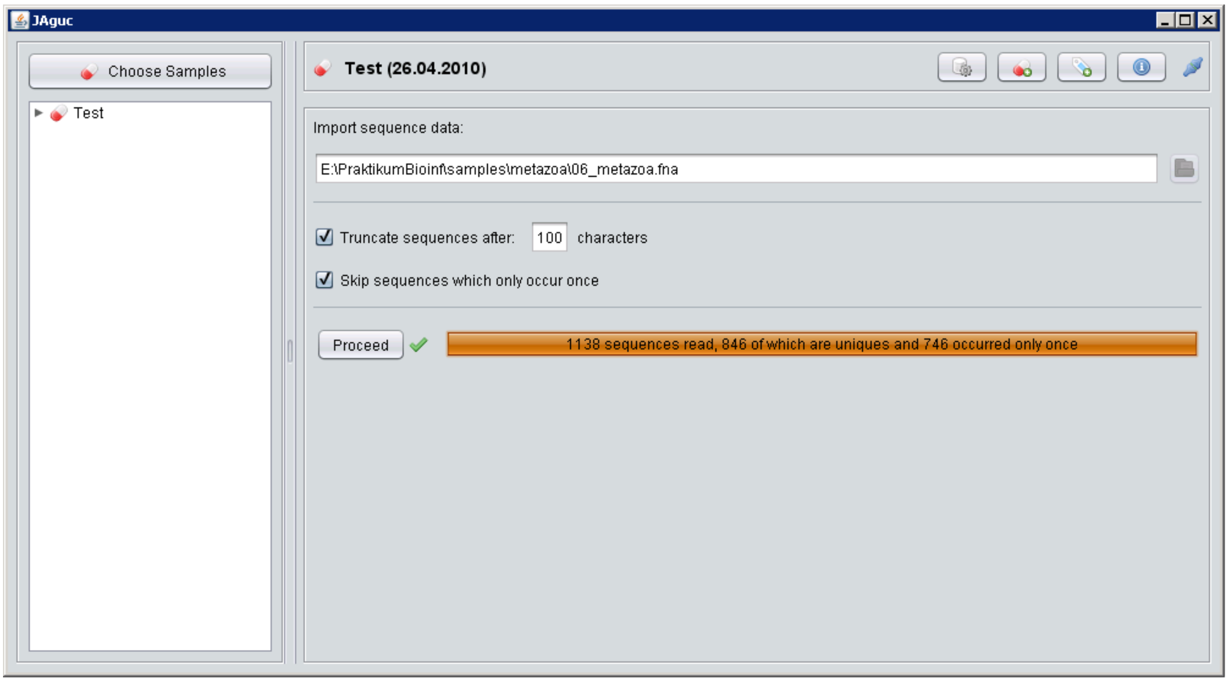
Figure 2: Sample Management

When the configuration is complete, a new sample should be created (if this has not happened before). The corresponding panel can be reached via the *Manage samples...*

button (top right, middle button). For each sample, a name, date, and a description can be defined. In a later step, it will be required to specify a file containing the input sequences. Click save to store the information. Previously created samples can be edited by double-clicking its name in the list.

2.3 Choosing Samples

When the sample has been created, click *Choose Samples* on the top left, and choose the according sample. If no sequences have been imported for this sample yet, you will be asked to do so.



Import sequence data:

E:\PraktikumBioinf\samples\metazoa\06_metazoa.fna

☑ Truncate sequences after: 100 characters

☑ Skip sequences which only occur once

1138 sequences read, 846 of which are uniques and 746 occurred only once

Figure 3: Importing a FASTA file

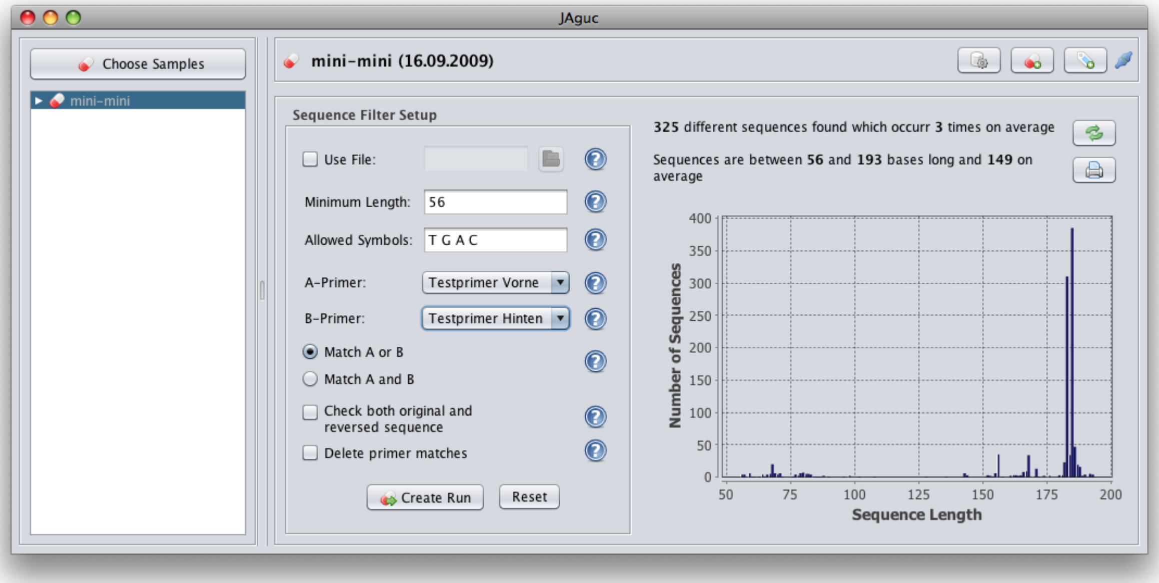
This panel allows the specification of two parameters:

Truncate sequences after x characters: This means that only the first x characters are imported into JAguc, which can for example be helpful if only the first x characters are believed to be reliable for a given sequencing method.

Skip sequences which only occur once: This allows to omit sequences that occur only once and thus probably contain sequencing errors.

Click *Import* to complete this step. The list on the left now contains the sample, and clicking on the triangle will show all the unique tags that occurred in the input that has been loaded. This, of course, is a rather long list and will not be particularly helpful. After the sequences have been clustered, these sequences will be assigned to their clusters.

2.4 Filtering



JAguc

mini-mini (16.09.2009)

Sequence Filter Setup

325 different sequences found which occur 3 times on average

Sequences are between 56 and 193 bases long and 149 on average

□ Use File:

Minimum Length: 56

Allowed Symbols: T G A C

A-Primer: Testprimer Vorne

B-Primer: Testprimer Hinten

● Match A or B

○ Match A and B

□ Check both original and reversed sequence

□ Delete primer matches

Figure 4: Filtering options

The final step before starting to align and cluster is creating a new *analysis run*. This can be parametrized in several ways, the first of which involves filtering options. The diagram on the right shows how many sequences there are for each sequence length, which might be helpful for choosing a reasonable value for the minimum sequence length. As with any other JAguc diagram, holding & dragging allows for zooming, double clicking provides a black & white printable version, and right clicking allows further customizations.

The following options are available in this panel:

Use File: If this option is checked, a previously created *similarity file* can be loaded. Similarity files basically store the results of the aligning process. Thus, when using the option, aligning will be skipped, instead JAguc will proceed with the (much faster) clustering right away. A similarity file can be reused whenever a certain input (FASTA) file is intended to be analyzed again. Filtering settings etc. are all stored within the file and will be loaded accordingly. It is hence not necessary to use the same database as when initially computing the alignments.

Minimum Length : This field allows to specify a minimum sequence length. All sequences shorter than this will be ignored in further processing.

Allowed Symbols : This field allows to specify allowed symbols. Sequences containing any symbol other than those specified will be ignored in further processing.

A-Primer : This allows to specify a prefix that has to be matched by all sequences of interest. For details on primers, refer to the next section.

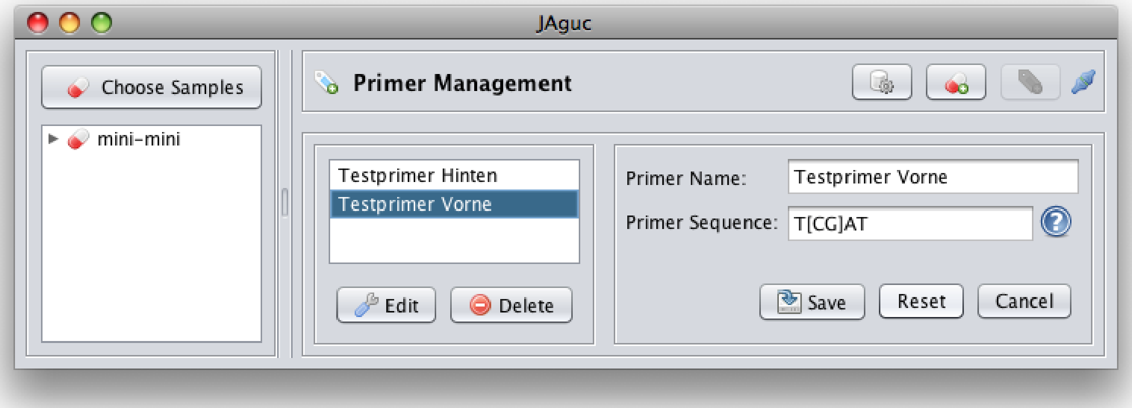
B-Primer : This allows to specify a suffix that otherwise behaves the same as an A-Primer.

Match A or B / Match A and B : Allows to define behavior when both A-Primer and B-Primer have been specified.

Check both original and reversed sequence : Both the original sequence and its mirror will be matched against the primer(s). Either match will be considered in further processing.

To customize primers, refer to the next section. Otherwise, click *Create Run* and proceed to Section 2.6.

2.5 Primer Management



Primer Name: Testprimer Vorne

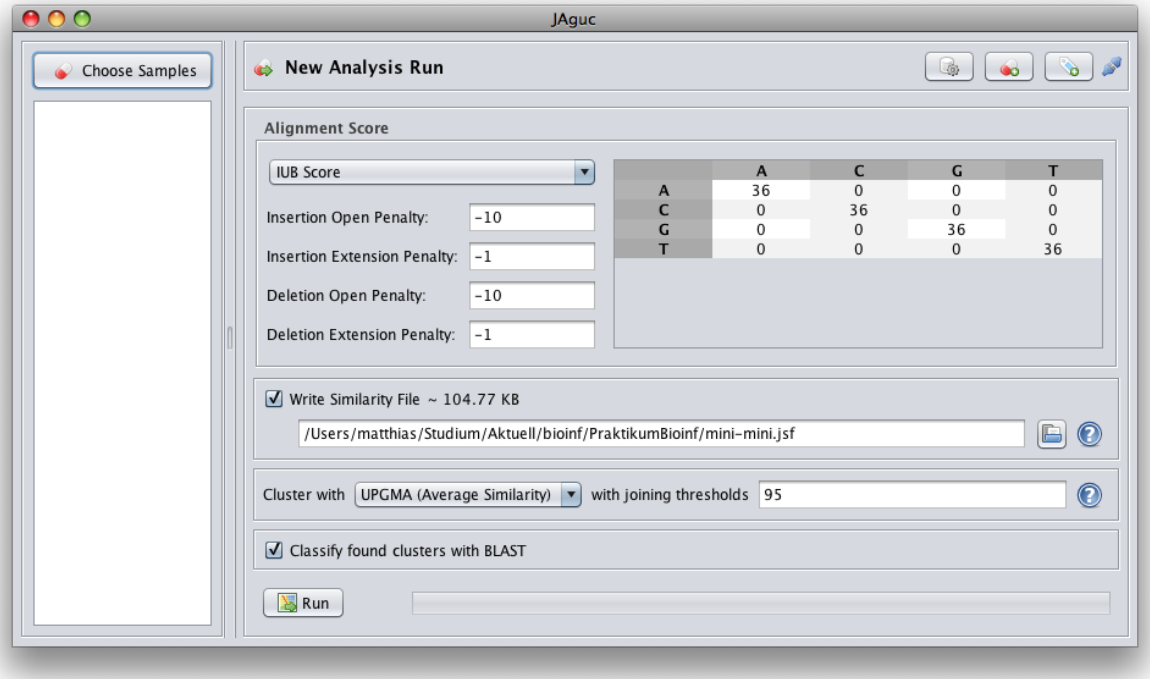
Primer Sequence: T[CG]AT

Figure 5: Primer Management Panel

This panel allows to manage primers, in a similar manner as the sample management. For each primer, a name and the primer itself are specified. For the primers, any kind of regular expressions are allowed.

In particular, write “T[CG]AT” to get a primer that allows words starting with a ‘T’ continuing with a ‘C’ *or* a ‘G’ and then with an ‘A’ and another ‘T’.

For further information on allowed expressions, refer to the Java reference on regular expressions, for example `http://java.sun.com/docs/books/tutorial/essential/regex/`.



Alignment Score

IUB Score

Insertion Open Penalty: -10

Insertion Extension Penalty: -1

Deletion Open Penalty: -10

Deletion Extension Penalty: -1

	A	C	G	T
A	36	0	0	0
C	0	36	0	0
G	0	0	36	0
T	0	0	0	36

☑ Write Similarity File ~ 104.77 KB

/Users/matthias/Studium/Aktuell/bioinf/PraktikumBioinf/mini-mini.jsf

Cluster with UPGMA (Average Similarity) with joining thresholds 95

☑ Classify found clusters with BLAST

Figure 6: Creating a New Analysis Run

2.6 New Analysis Run

This is the second and last step for parametrizing the analysis run. It includes four parameters:

Alignment Score: This allows to customize scores for the aligning itself. Generally speaking, the aligner tries to maximize values. Thus, negative values can be regarded as penalties, positive values as scores. You have the choice between the *IUB Score* and a generic score which can be customized in the following ways:

Insertion Open Penalty: This is the penalty that is applied for each sequence of consecutive insertions once.

Insertion Extension Penalty: This penalty will be applied for every individual insertion.

Deletion Open Penalty: This penalty is applied for each sequence of consecutive deletions once.

Deletion Extension Penalty: This penalty will be applied for every individual deletion.

The Matrix: Here, scores for substitutions are defined. The upper symbols represent the upper sequence when aligning, the symbols on the right represent the lower sequence. Currently, only symmetrical alignment scores are allowed.

Write Similarity File: This allows to create a similarity file simultaneously to the aligning process. See Section 2.4 for information on how similarity files can be used.

Clustering JAguc uses *UPGMA* to cluster[3]. It is possible to specify how UPGMA should calculate similarities after having joined two clusters: Either, the average of the two clusters to all other clusters is used, or the minimum or maximum of the two. Furthermore, a percentage defining the similarity threshold for when to join two clusters can be specified.

Classify found clusters with BLAST: This controls whether or not, after clustering, the representative sequences will be classified via BLAST. This can also be done manually afterwards.

Now JAguc is ready to align & cluster, you can tell it to do so using the *Run* button.

2.7 Analyzing the Results

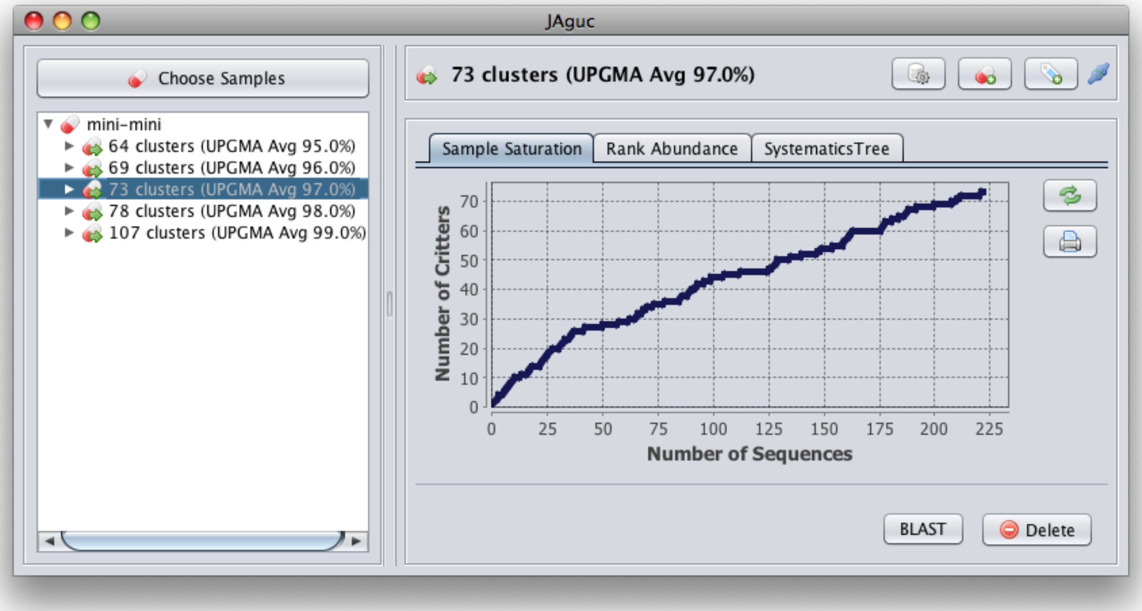

JAguc
Choose Samples
mini-mini
64 clusters (UPGMA Avg 95.0%)
69 clusters (UPGMA Avg 96.0%)
73 clusters (UPGMA Avg 97.0%)
78 clusters (UPGMA Avg 98.0%)
107 clusters (UPGMA Avg 99.0%)
73 clusters (UPGMA Avg 97.0%)
Sample Saturation
Rank Abundance
SystematicsTree
Number of Critters
0
10
20
30
40
50
60
70
0
25
50
75
100
125
150
175
200
225
Number of Sequences
BLAST
Delete

Figure 7: The Results After Several Analysis Runs

Figure 7 shows JAguc after several analysis runs have been performed. On the left, a tree is displayed, which gives an overview over the different runs. Hovering a run

[3] see Durbin et al. *Biological sequence analysis*

with the mouse will reveal further details. Clicking on the triangles displays a run's clusters, as well as a cluster's unique tags. The representative sequence which is used for BLAST is highlighted in green. The BLAST button is useful for classifying clusters if this has not been done before, or reclassifying them (for example if the BLAST database has changed).

On the right, three different graphical views can be activated using the tabs.

Sample Saturation: Shows how many species can be recognized when only considering a certain amount of input sequences. This allows to judge whether or not the sample contains a significant proportion of the habitat's overall population. In the latter case, additional samples of the same habitat will be likely to reveal new species that were not represented in the first sample. This is the case if the curve is growing in a linear manner. Otherwise, if the curve becomes fairly flat when approaching the maximum number of sequences, it can be assumed that most of the habitat's population is represented in this sample. Figure 8 shows an example of this. Since the sample saturation is created using a randomized algorithm, a *Refresh* button is available for recalculating the curve.

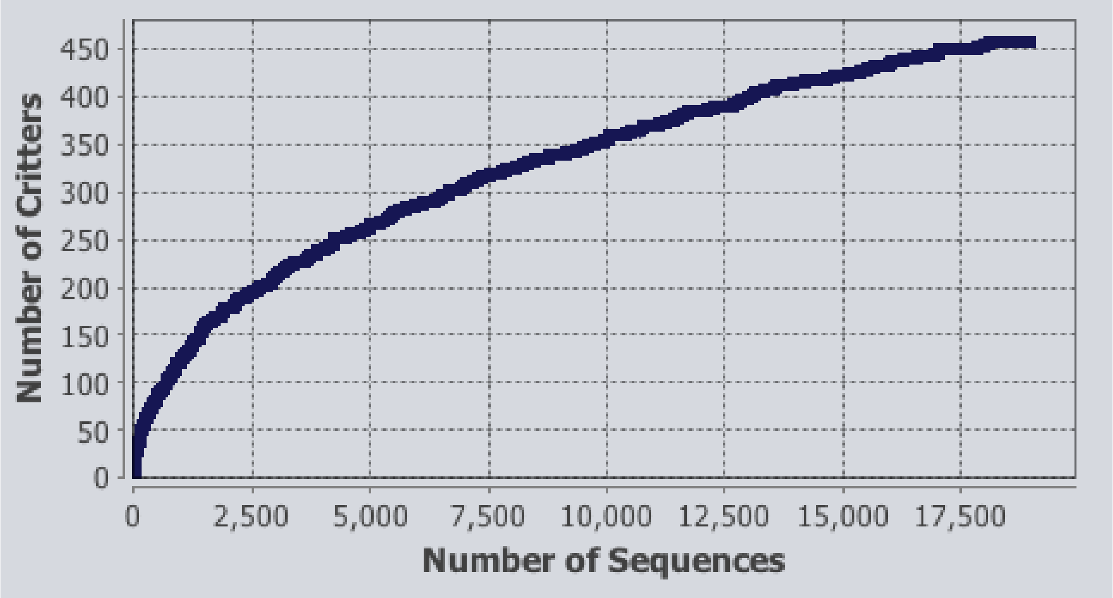


Figure 8: A Flattening Sample Saturation diagram.

Rank Abundance: This diagram gives an overview of the size of the clusters. It displays the clusters in order if decreasing size, using a bar that is longer, the bigger the cluster. Since maximum and minimum cluster size can be pretty far apart, it is probably the best to use a logarithmic scale, which can be toggled using the button on the right.

Systematics Tree: This displays a tree containing the classified clusters in their biological order. Double-clicking unveils sub-clusters, and so on. Right-clicking

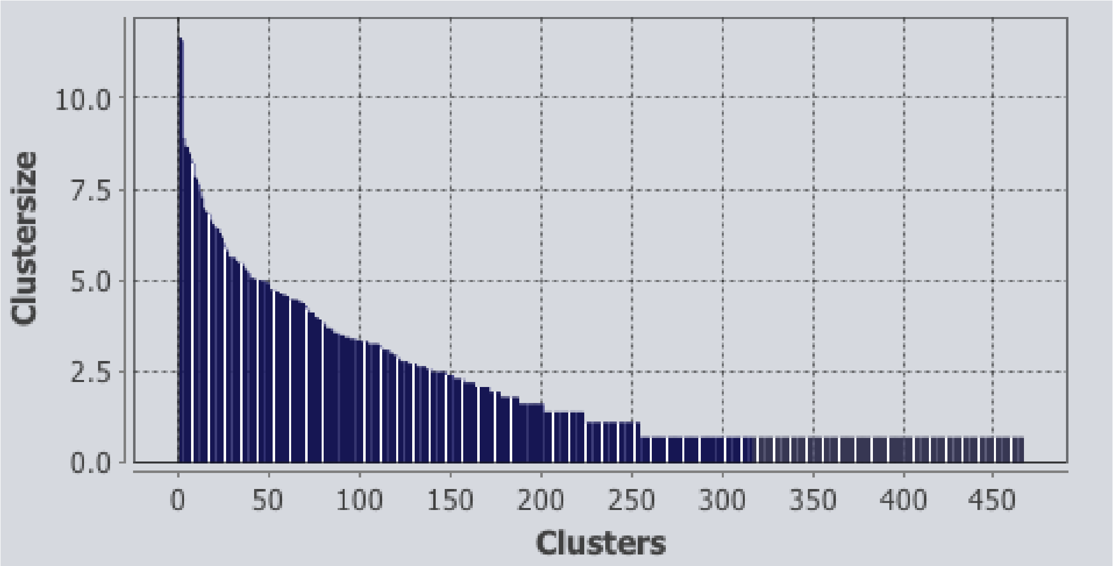


Figure 9: A Rank Abundance Diagram Using Logarithmic Scale.

displays an option for exporting this cluster into a new FASTA file. The buttons on the right provide functionality to toggle between *best fit* and *max fit*, as well as several ways of controlling the view.

2.8 Advanced Parameters

JAguc contains a small number of additional parameters that cannot be configured using the graphical interface, but instead must be changed by opening the settings file (*.config) with a text editor. Most of the parameters available in this file can be configured graphically and have already been introduced, except for the following:

BLASTMatchReward=x: Match reward parameter used by BLAST when aligning sequences. Default: 5.

BLASTDismatchPenalty=x: Dismatch penalty used by BLAST when aligning sequences. Default: -4.

BLASTGapOpenCost=x: Specifies the costs for opening a gap, used by BLAST when aligning sequences. Default: 8.

BLASTGapExtensionCost=x: Specifies the costs for extending a gap, used by BLAST when aligning sequences. Default:6.

BLASTHitsToEvaluate=x: Specifies how many BLAST results should be evaluated. Default: 50.

CritterSpecifiedThreshold=x: For the BLAST best fit or max fit determined for each cluster, this parameter allows to specify how similar (percentage) it should be to the cluster's representative to be considered valid. If invalid, the sequences will be marked unspecified. Default: 80.

3 Some Notes About Performance

In this chapter, we present some hints about how to make JAguc faster. Generally speaking, JAguc's main bottlenecks are the aligning algorithm and the clustering algorithm, though in many cases clustering will be a lot faster.

3.1 Aligning

The aligning algorithm computes pairwise alignments of all unique tags, and makes use of common prefixes of sequences by reusing previously computed alignment matrices where applicable. It's run time depends on the following factors:

CPU speed: Directly impacts alignment speed.

Number of CPUs/cores: The alignment algorithm makes heavy use of multithreading. Doubling the amount of CPUs (or cores) will almost (though not quite) double its speed.

RAM: Not critical for aligning.

Number of unique tags: Can be reduced by increasing minimum sequence length or generalizing primers. Since pairwise alignments are calculated, run time lies in $\mathcal{O}(n^2)$, and can drastically be reduced by decreasing the number of sequences (n denotes the number of unique tags).

Length of sequences: Sequences can be made shorter by cutting off the primers. Shorter sequences have a similar impact as decreasing their number, since time is saved for each of the $\mathcal{O}(n^2)$ pairwise alignments.

Matching prefixes: As mentioned before, our aligning algorithm makes use of common prefixes by reusing parts of the alignment matrix. More and longer prefixes increase the reuse factor and thus reduce runtime.

Alignment score: The alignment score has no impact on its runtime whatsoever.

Similarity file: Writing the similarity file while aligning will not make it considerably slower.

3.2 Clustering

CPU speed: Directly impacts clustering speed.

Number of CPUs/cores: Clustering does not make use of multiple threads.

RAM: For good performance, it is important to have enough RAM. For 30,000 unique tags (after filtering), about 1GB of main memory is needed. Doubling the amount of unique tags will quadruple the necessary main memory. If not enough main memory is available, the clustering algorithm will store parts of its data on the hard disk, which is much slower than the main memory (although some optimizations have been made that try to minimize disk access).

Number of unique tags: Clustering time is close to linear in the number of unique tags. For this reason, fewer sequences will make clustering faster, but the difference is not as drastic as with the aligning.

Length of sequences/Matching prefixes: Have no impact on clustering.

Clustering methods: Average similarity can be a little slower than maximal/minimal similarity, though this should not make too big a difference.

References

[1] R. Durbin, S. Eddy, A. Krogh, and G. Mitchison. *Biological sequence analysis*. eleventh edition, 2006.