

# I. Introduction to Strainer

## A. What is Strainer

Strainer is a viewer for metagenomic data (shotgun, fosmid and GS20). It provides a useful graphical display of assembled sequence reads and a simple set of tools for exploring and manipulating such data. As a Java application, Strainer is platform agnostic. Strainer uses BioJava to read FASTA and GENBANK files as well as BLAST output and ACE formatted assembly files.

## B. Running Strainer

### 1. Requirements

Strainer requires Java 5 or later. This is sometimes still referred to as Java 1.5. The JRE5.0 can be downloaded for most systems from <http://java.sun.com/>. Apple computers running Mac OS X 10.4 (tiger) or later will already have Java 5 installed. Apple has not released Java 5 for older versions of the Macintosh operating system.

### 2. Downloading

Strainer can be obtained from Bioinformatics.org:

<http://bioinformatics.org/strainer/>

### 3. Running

Most operating systems will be able to execute the downloaded .jar file directly. Just place the file somewhere you can find it and double-click.

### 4. Command line execution:

On most unix/linux systems (including Mac OS X) the commands

```
java -jar strainer.jar
```

or

```
java -cp strainer.jar amd.strainer.display.Main
```

will launch the application. For large datasets it may be necessary to increase the memory available to Strainer using the `-mx` flag (see Java documentation or Appendices).

## 5. Obtaining the source

<http://bioinformatics.org/strainer/>

### C. Concepts

#### 1. Read

Also called an end-read or sequencing-read, this is one of thousands of short segments of DNA sequence generated by a sequencing project. In Sanger shotgun sequencing, these will usually be ~700 base pairs (bps) long.

#### 2. Reference sequence

This is the master sequence against which all the reads are aligned. Usually this is either the output of an assembler such as Jazz or Phrap, or it is from a closely related organism. It provides a framework upon which to build strains from the individual reads.

#### 3. Clone/MatePair

Two reads sequenced from opposite ends of a single cloned piece of DNA are called mate pairs. In Strainer, the term Clone refers to a pair of such reads and each read is called the MatePair of the other. In other words, each read will be associated with a single mate-pair (which in turn has that read as its mate-pair) and together they make up a clone.

Some sequencing methods, such as 454-sequencing and related methods, do not produce paired reads.

#### 4. Difference

Due to both sequencing errors and natural diversity, reads will often not align perfectly to the reference sequence. In these cases, the bases at which a read differs from the reference sequence are stored as a list of Differences. These are displayed as colored tick marks along the length of the read. These will sometimes be called “diffs” for brevity.

#### 5. Strain

Strainer helps the user build up groups of similar sequences by associating reads with similar patterns of Differences. When multiple reads are grouped together, they become a “Strain”.

## 6. Recombination

In some populations, organisms exchange DNA. If the population is sampled well enough, these events will be apparent. One end of read or clone will be grouped with one Strain and the other end will be grouped with a different Strain. Such sequences can be flagged as recombinants. They will be highlighted in red. If a recombinant clone is divided between two strains, a red line will connect the strains indicating that DNA has moved laterally between them in the past.

## II. Loading data

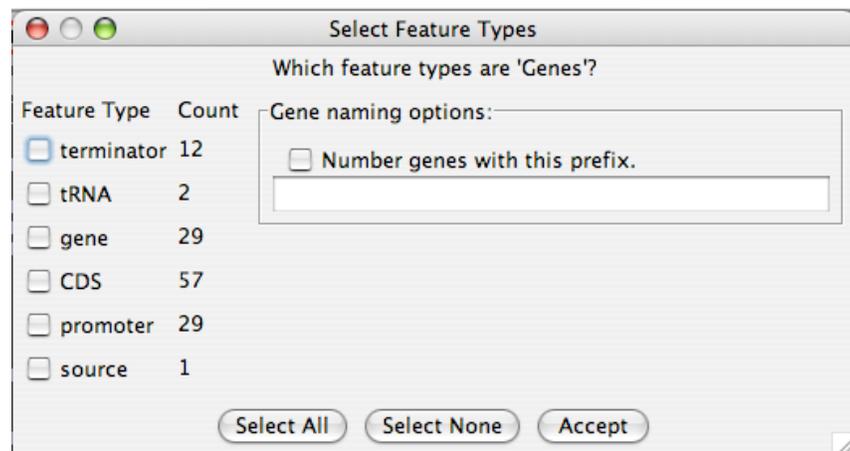
Data may be loaded from either ACE formatted assembly files as those produced by phred or consed or from a BLAST alignment. If mate-pair associations are to be recognized, read sequences should be named so that they differ from their mate-pair only by the first character after the “.” in their names. (e.g. XYG123.x1 and XYG123.y1 are from opposite ends of the same clone.)

### A. Separate Reference and Alignment data

#### 1. The reference sequence file

This is a file containing the name and sequence of your reference sequence. It may also contain a list of genes and their locations. It must be in FASTA or GenBank format.

When loading a GenBank formatted file, strainer will ask the user to indicate which annotations are to be treated as genes. This permits some flexibility in styles of annotation names.



The feature types on the left are a list of features found in the GenBank file. Check the types that should be displayed and then click “Accept.” To replace the feature names with custom names

numbered from left to right, check the box under “Gene naming options” and type a prefix into the text box.

## **2. The Strainer XML file**

This XML file is generated when you save your work from within Strainer. It contains read alignment information (including differences), strain groupings, and recombinant designations.

### **BLAST output**

If you do not have access to the assembler data, you can use the BLAST alignment tool to recreate the assembly. Use `blastn` to search for the short sequences in a database containing your reference. Make sure the output is in the `m0` format. (use `-m 0` when running `blastall`).

## **3. The LOAD dialog**

When using a separate reference sequence and read alignment files, select “Load” from the data menu or click the load icon (an open folder) on the left of the toolbar. The dialog has two text boxes, one for each file. You may either enter the location of the file directly or click the “browse” button to see a filesystem dialog.

## ***B. Assembled data***

Assembly information can be read directly from an ace formatted file. This file can be generated by the PHRAP assembler or by CONSED (allowing for refinement of the assembly prior to straining). Select the “import ace” option from the data menu. Select a file, then enter the contig number you wish to load.

Once the full contig assembly is loaded you can save the read alignments to a strainer XML file. If you do not have the contig sequence in its own file, use the “Export Reference” menu option to get the contig sequence as a FASTA file.

## ***C. Adding quality data***

Information on base quality can be imported into a strainer data set during the data loading process. The format of the quality file must be consistent with the output of `phred`. Each read should have an entry in this file. Each entry is a sequence of integers, each indicating the confidence of the corresponding nucleotide in the read sequence. The quality threshold is the minimum score that indicates a trusted base call. Anything below that will be ignored.

N.B. The quality data file generated by phred contains information on all reads in the sample and will take a long time to load. When using quality data for many scaffolds or contigs, it is best to use one of the bulk loading tools (Appendix A), so the file only needs to be read once.

#### ***D. Saving data***

A set of read alignments, strain groupings, and recombinant states can be saved to a strainer XML file. Select “Save Strains” from the data menu and choose the location to which data should be written.

The Strainer XML file does not store information about the reference sequence. This allows annotations to be updated independently of Strainer. However, it requires the user to keep track of two files at a time. Additionally, contigs loaded from an Ace file require the composite sequence to be exported separately. This can be done either in Strainer or CONSED.

#### ***E. Bulk import***

Strainer has a collection of bulk import methods to assist in the formatting of data. These are covered in Appendix A.

## III. Exploring data

### A. Graphical layout



#### 1. The reference bar

A black bar at the top of the display window represents the entire reference sequence. When the view is zoomed in on a particular region, a red box over the reference bar indicates the location of the data shown below. Clicking on the bar or dragging the red box will re-position the view.

#### 2. Reads

Reads are displayed in the main window as horizontal bars pointed at one end. The pointed end indicates the direction of sequencing and (for Sanger sequencing) the expected relative location of the read's mate-pair.

#### 3. Genes and other annotations

Annotations read from a GenBank file will appear just below the reference bar and above the reads. Genes are drawn as dark grey bars, rectangular on one end and pointed on the other. The direction of the point indicates the orientation (strand) of the gene.

#### 4. Differences

When the zoom level is such that at least one pixel width is devoted to each nucleotide position, each read is marked with vertical colored ticks to indicate where it differs from the reference sequence.

## **B. Changing the view**

The following options are also accessible from the “view” menu.

### **1. Zoom**



- a) The zoom buttons on the tool bar will change the scope of the displayed region.
- b) The user can also zoom in by right clicking (ctrl-click on a Macintosh) and dragging across the display window. The view will zoom in to show the region spanned by the dragged mouse.

### **2. Panning**



- a) The left and right arrows in the toolbar will shift the display to the left or right by half the current width of view.



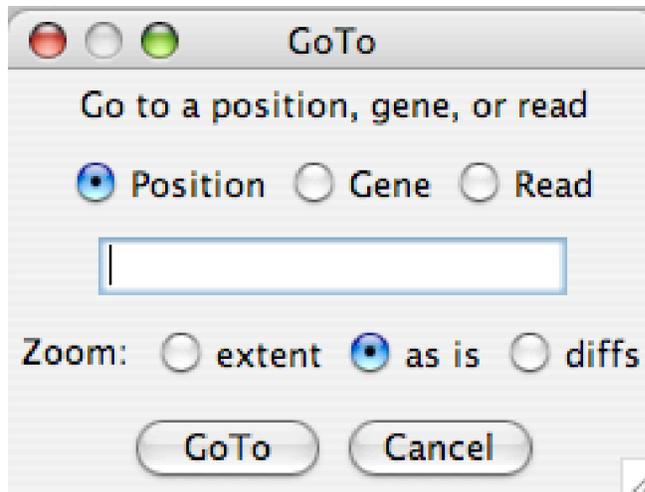
- b) The red rectangle over the black reference sequence bar indicates which portion of the reference sequence is currently displayed. The user may click anywhere on the black bar to re-center the display at the base represented by the point clicked. The user may also click on and drag the red rectangle to move the display.

### **3. Row Size**

The “bigger rows” and “smaller rows” options in the “view” menu control the zoom in the vertical dimension. The smaller row setting allows more sequences to be displayed at the expense of visibility.

### **4. GoTo**

The “GoTo” option allows the user to jump to any position in the data.



The user can jump to any nucleotide position on the reference sequence, any gene in the annotations, or a specific read. The zoom level at the new location can be left “as is”, set to the “extent” of the gene or read, or set to the minimum level to see colored differences.

### **C. View options**

The “Options” item in the “View” menu allows the user to adjust the way data is displayed. The dialog box has three panes. The first two are similar and set up display options for reads and strains (i.e. groups of reads) respectively. The third has miscellaneous options. The options for reads and strains follow.

#### **1. Sorting and stacking**

Read and strains are “stacked” onto the display in the order defined here. The user may choose from of: length of the sequence, the number of reads in a strain (default for strains) or identity to the reference(default for reads). The first strains are placed in the top of the display in their proper alignment and subsequent strains are placed lower if necessary to make the alignments correct. The same process is used to place reads within strains. When strict stacking is selected, no read or strain can be placed directly above any part of an earlier placed read or strain.

#### **2. Colors**

The coloring of reads and strains can be set by the user. There are 3 options for reads. There are 4 (the same 3 plus one more) for strains.

##### **a) Constant**

A single color can be selected for all reads or for all strains.

### **b) Random (Strains only)**

Strain colors can be randomized for easier visual separation.

### **c) Shaded**

Each read or strain can be made a solid color based its similarity to the reference. The user selects two colors, one represents completely identical sequences and the other, sequences below a user-defined similarity threshold. Each read or strain is colored on a sliding scale between these.

### **d) Two-toned**

In this mode, each sequence is colored based on a similarity threshold using two colors. At each point along the length of the sequence, the “high” color is used if the underlying sequence is above the threshold and the low color, if not.

## **3. Other options**

### **a) Recombinant colors**

These are the colors used to outline recombinant reads.

## ***D. The data window***

Below the main display window, a text box displays information about the selected objects.

### **1. Gene annotations**

When a gene is selected, two lines are printed in the text box. First is a short line with the gene’s name and position. The second, longer line contains all the annotation information for the gene.

### **2. Sequence information**

When a sequence is selected a short description is printed.

#### **a) Strains**

The identification number for the strain, the number of reads it contains, and its location on the reference, are printed in one line. If strain identities are being calculated on the fly (see Appendix B), a second line is printed indicating the strain’s similarity to the reference.

When two overlapping strains are selected in succession, the similarity between sequences is printed (again, only if strain identities are being calculated on the fly).

### **b) Reads**

The read name and position are printed along with the similarity to the reference.

### **c) Clones**

The name and positions are printed for the pair of reads that make up the clone, followed by the overall similarity to the reference

## ***E. The pop-up information box***

A small yellow box pops up if the mouse is left hovering over the display window. Its contents depend on the location of the mouse. The box always contains the name of the reference sequence and the current position. If the mouse is over a gene, that gene's name is added. If over a strain, the strain ID number and – if strain similarities are being calculated (Appendix B) – the percent identity to the reference are included. The name and percent identity are printed for reads.

If the zoom level is high enough that individual bases are about one pixel wide (colored difference ticks appear), the pop-up box will also indicate the base at the current position.

## **IV. Working with data**

### ***A. Grouping reads***

#### **1. Selecting reads**

The first step in making read groups (defining strains) is selecting reads to put into a group. Reads that have been selected are outlined in blue.

#### **a) Clicking**

Reads can be selected by clicking on them directly. If the read is in a clone, the entire clone is selected. To select just one of the mate pairs in a clone, double click on the desired read. If a strain is clicked upon, all its reads in that strain are selected simultaneously. Clicking a second time will unselect the read or strain.

#### **b) Boxing**

The user can draw a box around reads, and only those reads that are completely within the box will be added to the selection.

### c) GoTo

The GoTo operation will add the destination read to the selection list.

### d) Clear selections



The clear selections action will remove all recently selected reads from the running list.

### e) Select all

All reads in the data set are marked as selected.

## 2. Creating strains



The create strains button will group all selected reads into a single strain.

## 3. Get mate pairs

On occasion, reads get separated from their mate pairs. To reunite them, select the “Get mate pairs” option from the selection menu. This will bring the mate pair of the selected reads into the strain with it. If an entire strain was the last object clicked, all reads will have their mate pairs brought in. If “get all mate pairs” is selected from the auto menu, then for all strains, from smallest to largest, all missing mate pairs are brought in.

## 4. Undo!



The undo and redo buttons allow the user to navigate back and forth through recent changes.

## ***B. Changing the reference sequence***

Often, assemblies of non-clonal sequences produce composite sequences that are chimeras of multiple strain types. In these cases it may be desirable to modify the reference sequence so it reflects the dominant strain type.

Strainer allows the user to alter the reference sequence by right clicking on the strain or read. If the user right-clicks on a strain or read, the reference

sequence is altered so that it matches the selected sequence for the full length of that sequence. If the user clicks on a colored difference within a read, just that position will be changed.

Once the reference sequence has been altered, it is necessary to export the new sequence to a FASTA file in addition to saving the new strain and read data.

In order to keep memory usage manageable for large data sets, strainer only stores quality data for SNPs. Changing the reference sequence will produce new SNPs with no quality data. If complete quality data is needed, select “Data->Import Quality” to re-load quality information once the reference is fully altered to fill in the missing values.

### ***C. Tagging recombinant reads***

Reads that show evidence of recombination can be tagged. This does two things. First, reads are outlined in red to make them stand out. Second, if the two mate pairs in a clone are placed in different strains, strainer will draw diagonal lines to connect them if they are tagged as recombinant.

#### **1. One at a time**

The “toggle recombinant” action will switch the status of the currently selected read from untagged to tagged or vice versa.

#### **2. Multiple reads**

The “toggle all recombinant” action will flip the status of all selected (highlighted in blue) reads. Note that it will not make them all the same, it will switch the status of each separately.

#### **3. Undo!**

The undo and redo buttons also apply to recombinant designations.

### ***D. Exporting data***

#### **1. Read lists**

Strainer can export a list of read names. Simply select the reads you are interested in or click on a strain to get all its reads. Then choose “Get read list” from the selection menu.

#### **2. The Get Sequence dialog**

The “Get Sequence” option in the selection menu lets the user output sequence from selected (or all) sequences.

### **a) Example uses**

To get the consensus sequence for a single strain, click on the strain to select all its reads, and in the get sequence dialog, choose “selection sequence” as the output element and “selection span” as the output scope.

To get the protein sequences for each strain at a given gene, select the gene, then in the get sequence dialog choose “strain groups” as the output elements and “gene span” as the output scope and “amino acids” as the output dictionary.

To compare the sequence of a few reads over a short region, select the reads you are interested and zoom the view to show the region you want. In the get sequence dialog, choose “individual reads” as the output elements, “visible region” as the output scope, “just selected” under “limit to selection,” and “MSA” as the output format.

### **b) When to use “Variant Sequences” instead**

To get the strain sequences at every gene in the reference sequence in one action, use the ManualStrainer algorithm in the Autostrainer dialog.

## **V. Autostrainer**

The “Variant Sequences” option in the “Auto” menu gives the user access to algorithms that automatically generate variant gene sequences. Only one such algorithm (OriginalGeneCrawler) is supplied, but a simple API is provided for adding new algorithms.

Additionally, Strainer can use the algorithm output to automatically group reads into strains. This does not work perfectly, but provides a good starting point for building strains.

### ***A. The variant sequences dialog***

#### **1. Scope**

The user can choose to get the variants for a single gene, all genes, the currently visible segment, or the combined span of all the selected reads.

#### **2. Reads**

The user can limit the reads considered to those selected or just use all the data.

### 3. Output

Variant sequences can be output into a FASTA list. Strainer can also use the algorithm output to group reads into strains. Since a single read is allowed to contribute to more than one variant, generated strains often do not represent the full list of variants.

### 4. Algorithms

#### a) Selection

Use the drop-down box to choose the algorithm to apply.

#### b) Configure

Each algorithm can define the settings it needs. The “Configure” button gives the user access to these. It is in this dialog that the user can specify for gene sequences to be converted to proteins.

Some algorithms (SegmentLinker and Substrainer) break up the problem into pieces and use another algorithm to strain the pieces. To configure the settings of the underlying algorithm, you must temporarily select it as your primary algorithm and use the “configure” button set it up as you desire before going back to the primary algorithm you wanted.

#### c) Add/Delete

Any class implementing the SegmentStrainer interface in `amd.strainer.algs` can be added as an algorithm if it is compiled and placed in the class path used to run Strainer (Appendix D). Click on the add button and enter the full path and name of the class. Select an algorithm and click delete to remove it from the list.

## ***B. The standard algorithms***

### 1. OriginalGeneCrawler

This is the basic variant finding algorithm. It finds every possible chain of reads spanning the gene where the reads overlap by some minimum number of bases and have at better than the specified difference (fraction of bases differing) in that overlap.

### 2. SegmentLinker

OriginalGeneCrawler can be very slow for large regions with many, many reads (roughly, more than 4kb and 20x coverage,). Use segment linker to break large regions up into smaller bits.

### **3. ManualStrainer**

Manual strainer does no processing. It simply outputs the sequences for each user-generated strain overlapping the gene (or other selected region). This is useful for generating a list of strain sequences for every gene in the reference since the “get sequence” dialog does not offer an option to do more than the selected gene.

### **4. Substrainer**

Substrainer runs OriginalGeneCrawler within each user defined strain. Gaps in variants are filled with the strain consensus sequence (not the reference sequence).

## **VI. Appendices**

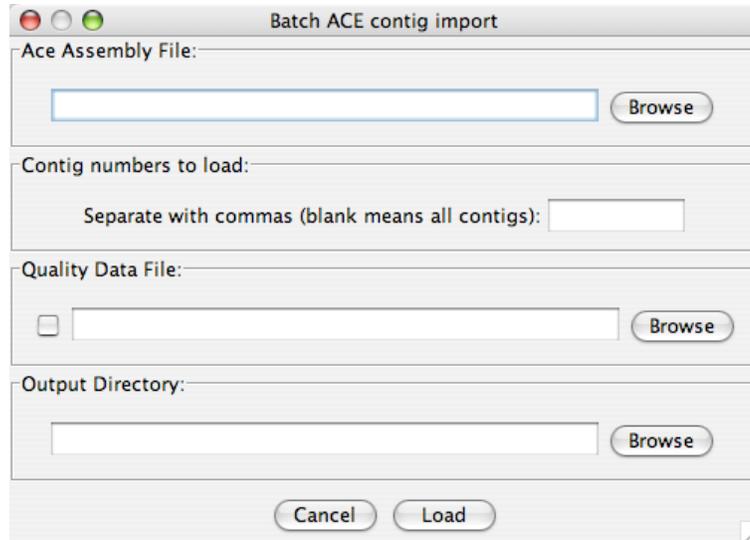
### ***A. Bulk imports***

Some tasks related to using strainer can take a while to complete. Loading quality data is a big culprit. So is parsing external formats such as BLAST and ACE files. These methods can be accessed from the “data” menu in the main program window. They can also be run from the command line without the need of a graphical display. This enables them to be run easily on remote servers that are more capable of handling large data files.

Since quality score files contain data for every read in a sequencing data set, parsing them can take a long time. It is not reasonable to re-scan the file for every new contig or scaffold if only a fraction of the reads in the quality file are aligned to the sequence of interest. One option is to use an external script to pull out the relevant reads first. Strainer offers an alternative approach: load the quality data for multiple reference sequences at once. This can be done with either bulk loading option.

#### **1. Bulk ACE import**

The bulk ACE import converts every contig in an ACE file into two files: a FASTA file containing the contig sequence and a strainer XML file containing the read alignments. This will generate many files (twice as many as contigs in the original ACE file), but will save time and space in the long run as the strainer format is more compact and loads much more quickly.



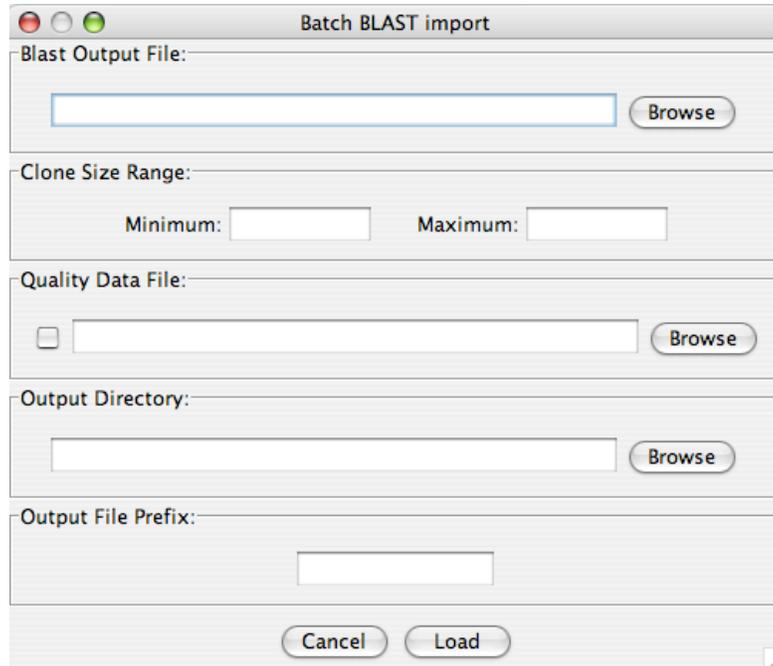
The “Ace assembly file” is the ACE file created by PHRAP or CONSED. You can process all contigs, or just a selected few. You can specify a quality data file to merge into the generated strainer files. Generated file pairs are placed in the specified directory. To run from the command line, enter the following at the prompt:

```
java -cp "strainer.jar"  
amd.strainer.file.AceFileReader -o OUT_DIR -q  
QUAL_FILE ACE_FILE
```

where OUT\_DIR, QUAL\_FILE, and ACE\_FILE are replaced by the actual file names.

## 2. Bulk BLAST import

The bulk BLAST import allows the user to align reads from a sequencing library against multiple reference sequences at once. This ensures that each read is aligned to only one reference and minimizes the confusion caused by duplicate genes. Strainer will choose the best alignment for each read that places the read within a reasonable distance of its mate-pair (that distance is user specified).



To run from the command line, enter the following at the prompt:

```
java -cp "strainer.jar"  
amd.strainer.file.BatchBlastTask BLAST_OUTPUT  
OUTPUT_DIR MAX MIN QUALITY_FILE
```

The “Blast output file” (or BLAST\_OUTPUT) is the output in m0 from a BLASTN search of a database of reference sequences using all reads as the queries. The “clone size range” (or MAX and MIN) is the upper and lower limit on the clone library size and hence mate-pair separation. The “Output directory” (or OUTPUT\_DIR) is the location to write the strainer XML files. The “Quality file” (or QUALITY\_FILE) is the PHRAP generated list of read quality data.

## **B. Calculation of strain differences**

For very large datasets, calculating the consensus for strains can dramatically slow the program and overwhelm the available memory. To alleviate this problem, the calculation is only done if the display requires knowledge of a strain’s alignment. This is the case if (A) strains are sorted by identity to the reference or (B) strains are colored by identity. In none of these options are selected, strain identities will not be calculated ahead of time and will not be printed in the info box. Strain alignments and consensus sequences will be calculated on demand when using the “get sequence” dialog.

## **C. Increasing memory**

Since only differences are stored (not every read sequence) memory usage is relatively low. However, shotgun sequencing data sets tend to be large and memory usage can quickly add up. It may be necessary to launch

strainer from the command line to instruct the Java environment to increase the maximum heap size. The command

```
Java -jar strainer.jar -Xmx512m
```

will allow strainer to use up to 0.5 GB of RAM. Adjust the amount of RAM to suit your needs.

WARNING: the `-X` options are non-standard and may possibly vary among Java distributions.

## **D. API for new algorithms**

New algorithms can be added to Strainer for generating variant sequences and automatically grouping reads. When run, the algorithm will be given a list of reads to be considered and the region of interest. It should generate a list of strains, each of which is defined by a set of reads. To be added to Strainer, the algorithm must be coded into a Java class that implements the `SegmentStrainer` interface

### **1. Interfaces and Classes in Java**

In Java, source code is compiled into **class** files. These file can be run directly by Java, either as stand-alone programs or as libraries for other programs to use. A library is a collection of functions (called methods in Java) that can be called by other programs.

An **interface** is a list of methods (also called routines or functions). A class may implement an interface only if it includes its own versions of all the functions listed in that interface.

### **2. The `SegmentStrainer` interface**

The following methods must be included in the class file (i.e. java library) that contains the algorithm to be added. These are further described in the javadoc and source code.

#### **a) Initialization methods:**

##### ***(1) void setSegment(`SequenceSegment` pSegment)***

Defines the region of the reference sequence to be processed.

##### ***(2) `SequenceSegment` getSegment()***

Tells other programs what region is being processed

##### ***(3) void setReads(`Iterator<Read>` pReads)***

Optionally limit the set of reads considered

***(4) void setTask(Task pTask)***

The Task sets up a mechanism for the algorithm to send status updates to Strainer so the progress bar can display meaningful information.

**b) The processing method**

***(1) StrainerResult getStrains()***

Executes the algorithm and returns the results.

**c) Methods that identify the algorithm to Strainer**

***(1) String getName()***

Returns the name of the algorithm

***(2) String getDescription()***

Returns a description of the algorithm

***(3) HashMap<String, Object> getOptionsHash()***

Returns a list of settings and default values