Genome Assembly Tutorial

PATRIC Workshop

Chicago - June 15, 2015

The PATRIC Assembly Service is a web-based environment that allows users to submit datasets of sequence reads to be processed, assembled, and analyzed. This tutorial will introduce you to the current capabilities of the service. You will learn how to upload a set of read files, assemble them, inspect the results, and select the best assembly for your downstream analysis.

Downloading example reads

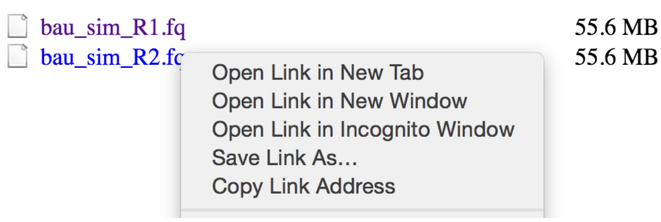
The example data sets used in the PATRIC tutorials are available on the FTP site:

ftp://ftp.patricbrc.org/workshop

1. For this tutorial, we will be using two simulated read files from a recently sequenced *Acinetobacter baumanni* genome (*A. baumanni* strain 100160). Please visit the FTP site in your web browser and download the following two files:

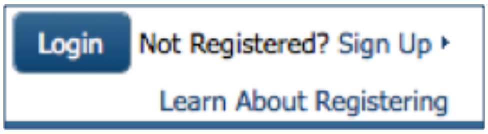
- bau_sim_R1.fastq
- bau_sim_R2.fastq

You may need to right click on the each file and select "save link as" to save it.

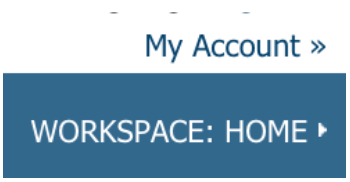


Uploading reads to PATRIC workspace

2. Login to the PATRIC website (patricbrc.org) so that you can use your workspace to store your private data and launch PATRIC analysis services.

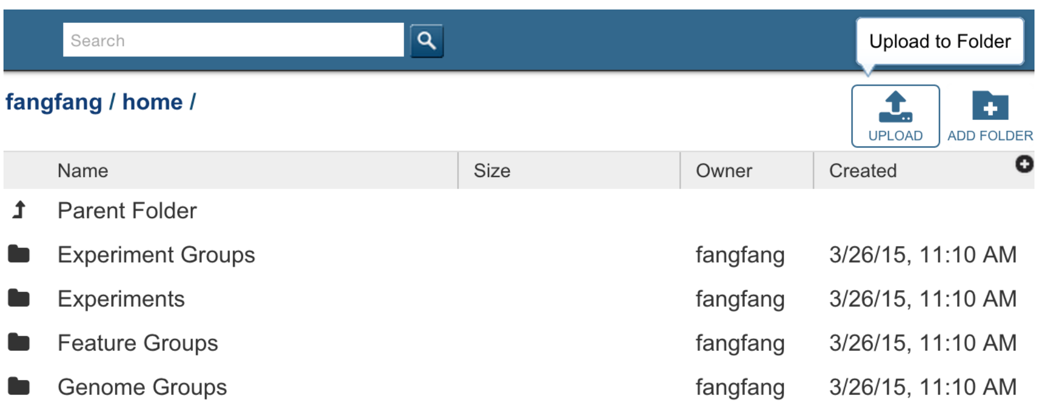


3. Once you are logged in, click on the **WORKSPACE** link at the upper right corner.

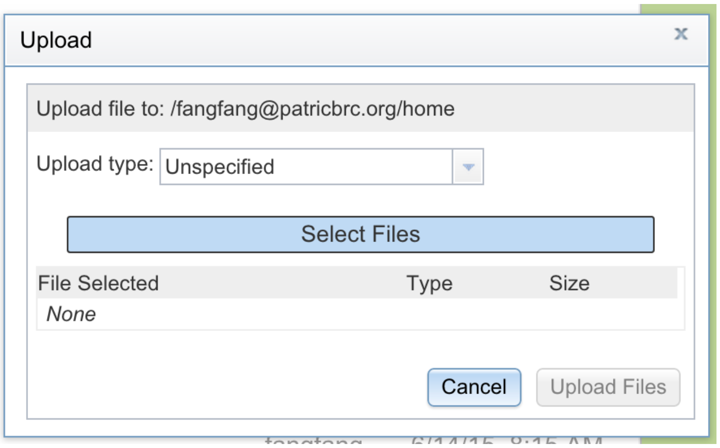


This will take you to the root level of your workspace (/username/home) which is populated with some folders such as "Genome Groups". Think of these as the conventional folders such as Pictures or Downloads on your computer. You can add more folders or remove them.

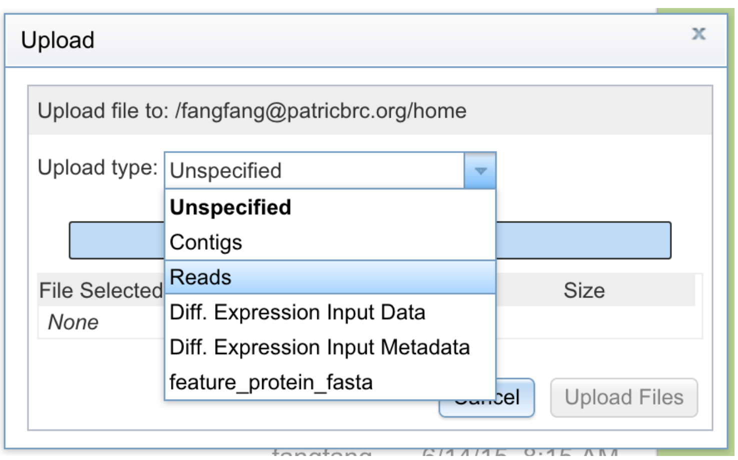
4. Click on the UPLOAD icon to send some local files to your PATRIC workspace.



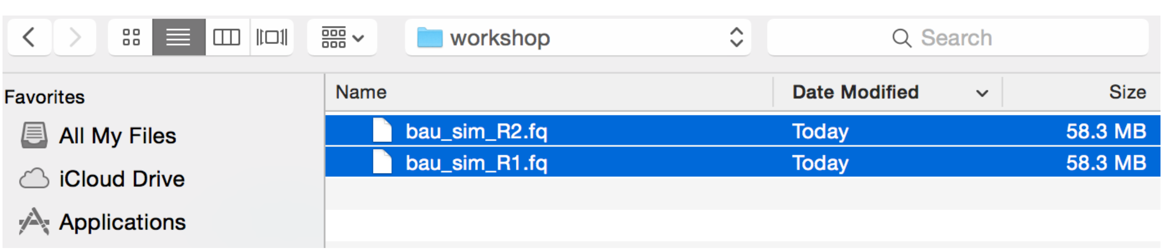
This will open an upload dialog for you to select the files on your computer and specify the types for them.



5. Select the **Reads** type from the “Upload type” drop down menu.

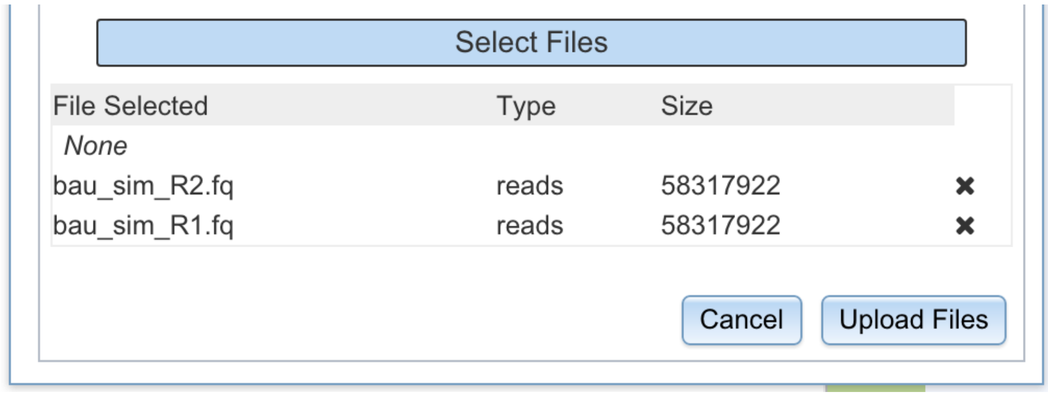


6. Click on the “Select Files” button to select the *A. Baumannii* reads you have downloaded in Step 1; this will open another dialog. Navigate to the folder where you saved the FASTQ files, click on the file(s) and confirm your selection.

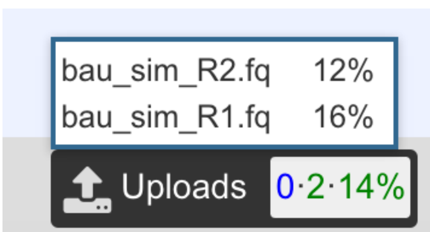


You may be able to use Shift+Click to select multiple files to upload depending on your computer and browser. Or you can upload the files one at a time.

7. Click on "Upload Files" to start uploading the reads.



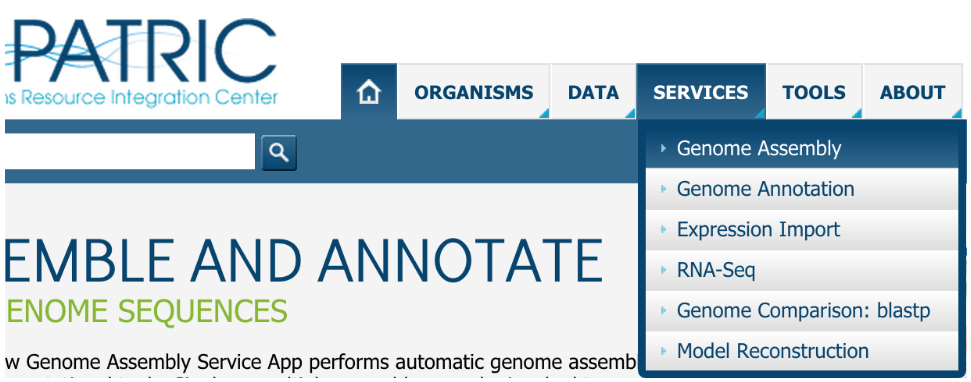
You should be able to see the upload progress in the dark status sections near the bottom right corner of the web page.



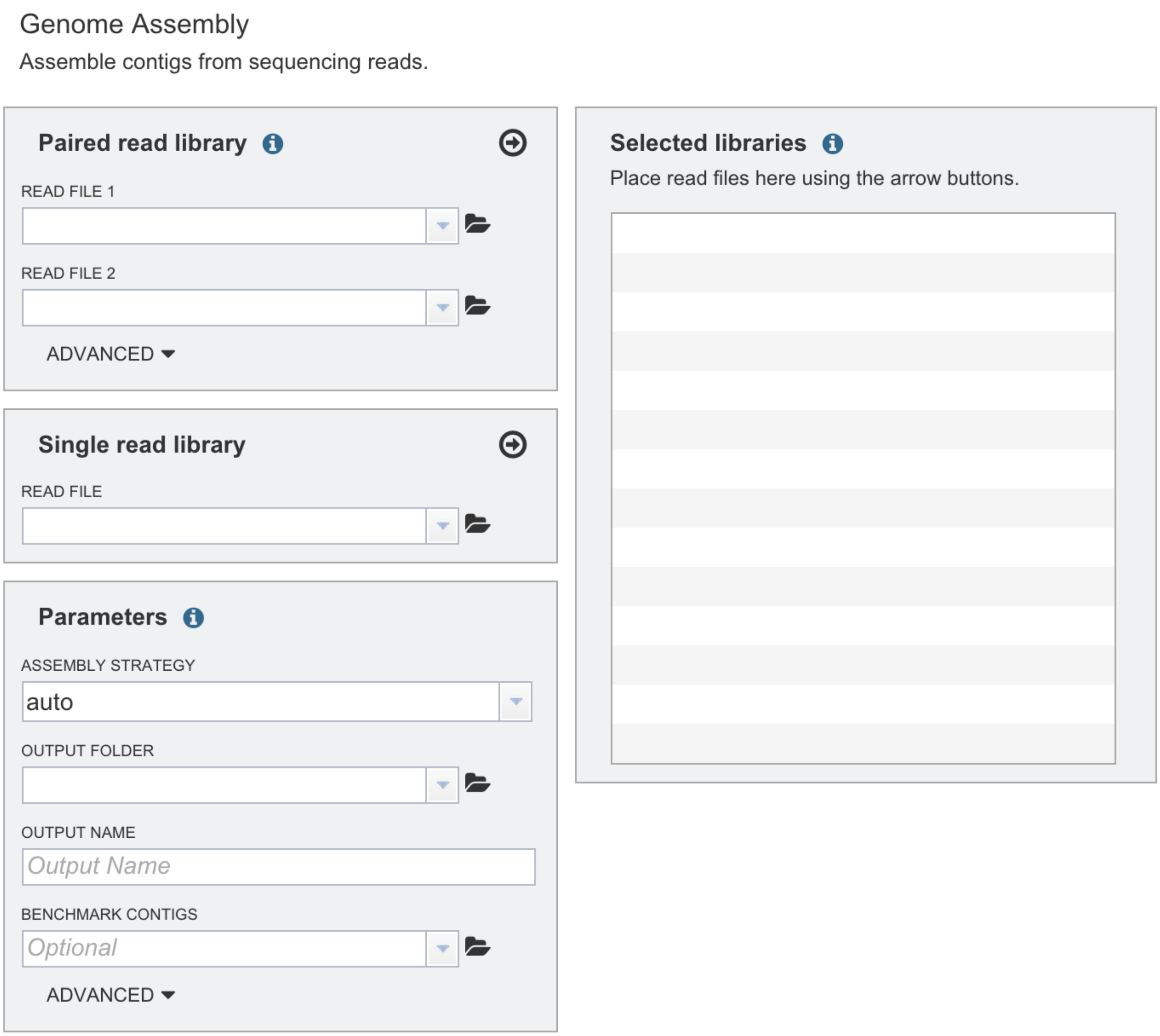
Note: You may not want to switch to or open other tabs in this browser window, for we have noticed that, on some platforms, multi-tabbing may cause the upload progress to stall.

Launching an assembly job

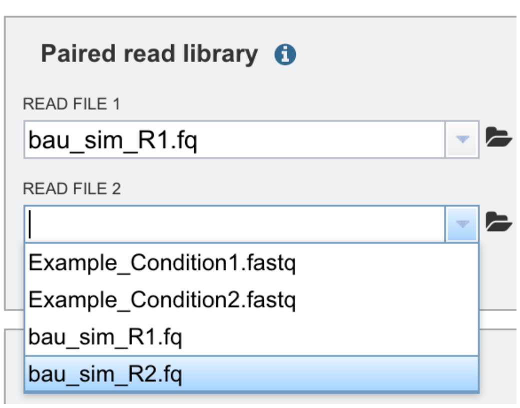
8. Wait for the upload to complete. Then click on the **Genome Assembly** service from the **SERVICES** drop down menu.



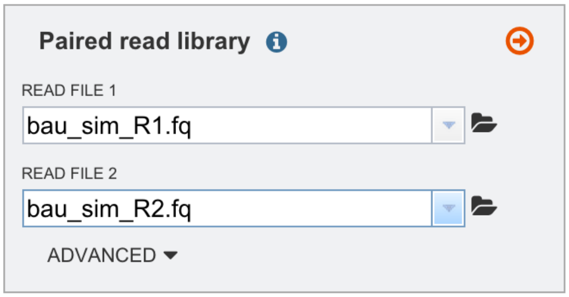
This will open up a page where you can specify the parameters for your assembly job.



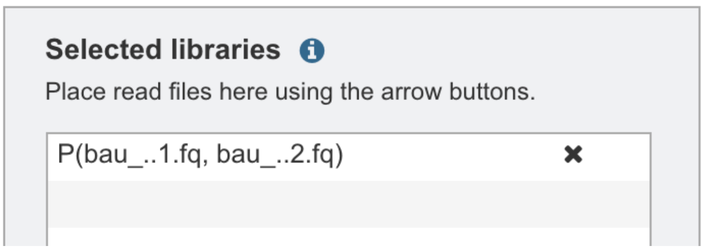
9. Select the *A. baumannii* read files in the **Paired read library** box.



10. Click the circled right arrow button to commit the read pair.

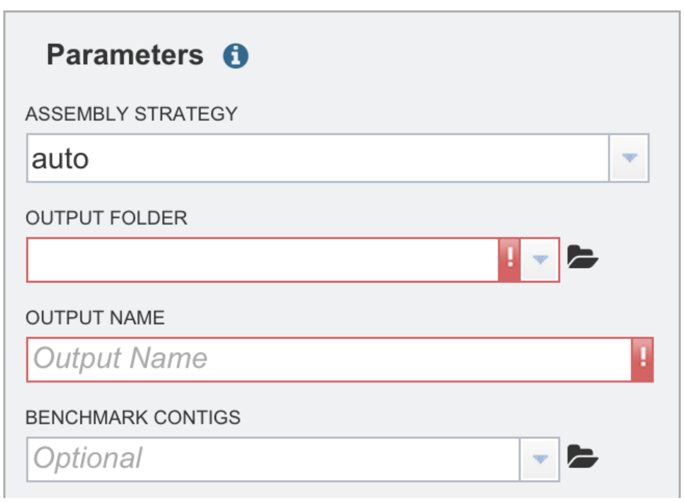


This paired end library should now appear in the **Selected libraries** box on the right. If you add some read library by mistake, you can remove it from the list using the cross button.

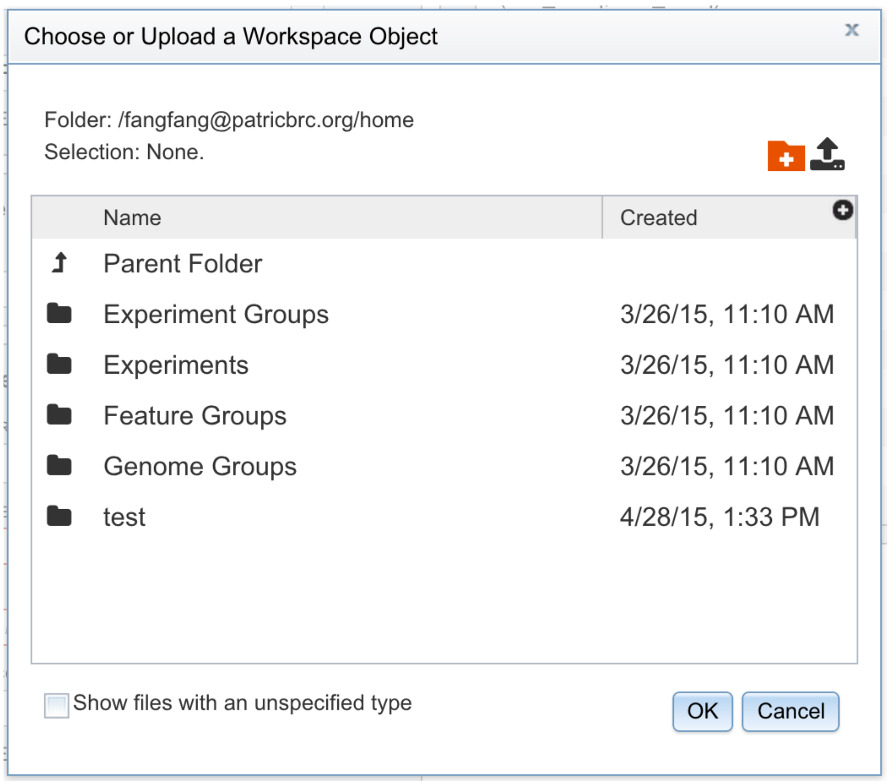


In this tutorial, we will run an assembly job with only one read library. But you can add multiple paired read libraries and single end libraries or mix them.

We will launch an assembly job with the default assembly strategy (auto), but there are two required parameters we need to set: **OUTPUT FOLDER** and **OUTPUT NAME**.

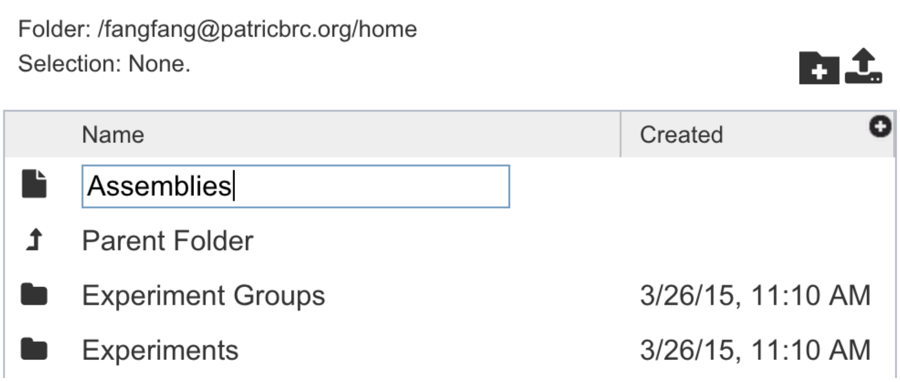


11. Click on the folder icon to the right of the **OUTPUT FOLDER** menu. This will open up a dialog for selecting a folder in your PATRIC workspace.

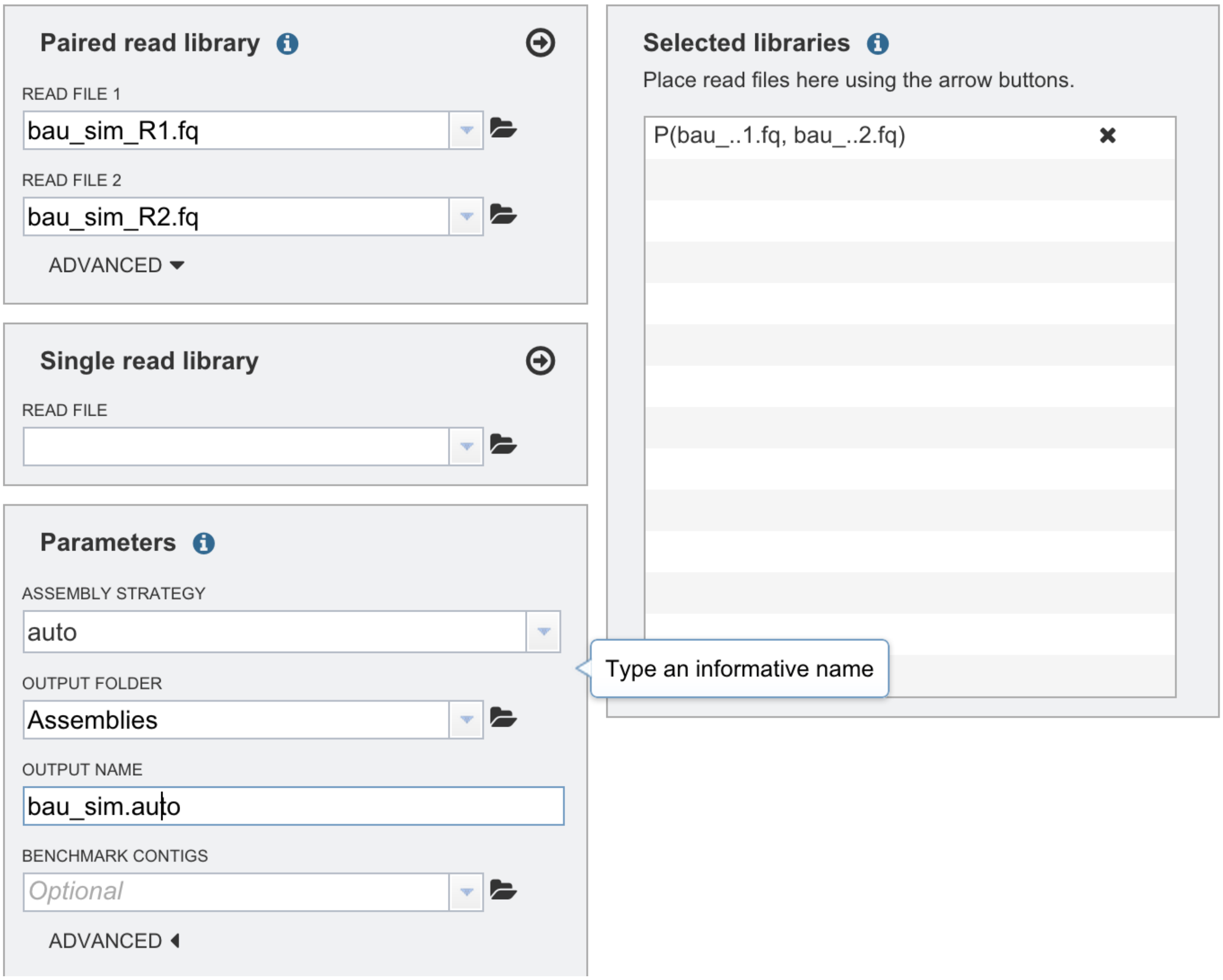


We will show you how to create a new folder from this interface to store your assembly output, although you can select one of the existing folders.

12. Click on the +folder icon and type "Assemblies" in the new text box. Then select this folder, and click on OK to close the dialog.

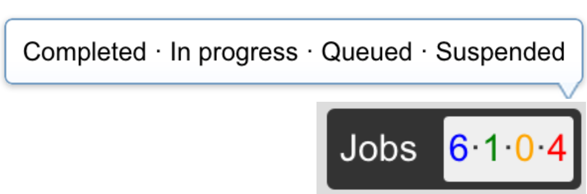


Now that you have filled in all the required parameters, the assembly page should look like this:



And the Assemble button at the bottom should no longer be grayed out.

13. Click on the Assembly button to launch the job. The job will first be queued, then in progress, and finally completed or suspended. You can find out the number of your jobs in each of the four possible states in the status section. Failed jobs are counted as suspended.



14. Click on the **Jobs** status section to see a full list of jobs.

Status	Submit	App	Output Name	Start	Completed
completed	6/14/15, 6:58 PM	GenomeAssembly	bau.auto	6/14/15, 6:58 PM	6/14/15, 7:05 PM
completed	6/14/15, 6:35 PM	GenomeAssembly	bau_demo	6/14/15, 6:35 PM	6/14/15, 6:42 PM
completed	6/14/15, 10:28 AM	DifferentialExpression	baumannii_test	6/14/15, 10:28 AM	6/14/15, 10:30 AM
completed	6/14/15, 8:15 AM	GenomeAssembly	rhodo	6/14/15, 8:15 AM	6/14/15, 8:32 AM
completed	6/14/15, 8:07 AM	GenomeAssembly	test1	6/14/15, 8:07 AM	6/14/15, 8:08 AM
completed	6/13/15, 1:58 PM	DifferentialExpression	test_exp	6/13/15, 1:59 PM	6/13/15, 1:59 PM
completed	6/11/15, 5:25 PM	RNASeq	testg37p	6/11/15, 5:25 PM	6/11/15, 5:25 PM

Our assembly job on the simulated *A. baumannii* reads should take ~7 minutes to complete.

Evaluating assembly results

15. Click on the completed job row to see the job details and result files.

fangfang / home / Assemblies / bau_sim.auto

Genome Assembly Job Result

Start time	6/14/15, 10:00 PM
End time	6/14/15, 10:06 PM
Run time	5m52s
Parameters	{"output_file":"bau_sim.auto","output_path":"/fangfang@patri [{"read2":"/fangfang@patricbrc.org/home/bau_sim_R2.fq","re

Result Files

	Filename	Type	File size
	6_analysis.zip	unspecified	234.8 kB
	report.txt	txt	121.7 kB
	6_1.spades_contigs.fa	reads	4.0 MB
	6_2.velvet_contigs.fa	contigs	4.0 MB
	6_3.idba_contigs.fa	contigs	3.9 MB
	contigs.fa	contigs	3.9 MB

The result files include:

- An assembly report (report.txt)
- An autom selected contig file (contigs.fa)
- All the raw contig files generated by different assemblers (x_y.assembler_contigs.fa)
- A zip file containing assembly statistics and visual comparisons (x_analysis.zip)

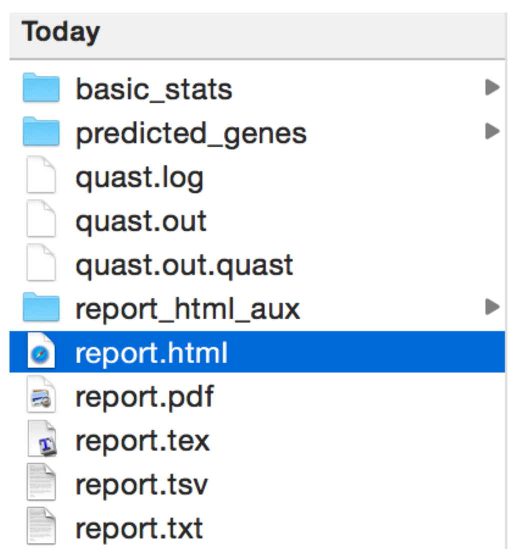
16. Click on the download button for the assembly report file.

17. Open the report file to get a basic idea of how the assemblies compare.

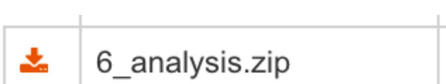
```
report.txt
All statistics are based on contigs of size >= 500 bp, unless otherwise noted (e.g., "#
contigs (>= 0 bp)" and "Total length (>= 0 bp)" include all contigs).

Assembly                    spades_contigs   velvet_contigs   idba_contigs
# contigs (>= 0 bp)         75               96               154
# contigs (>= 1000 bp)      52               33               105
Total length (>= 0 bp)      3885279          3889016          3879722
Total length (>= 1000 bp)   3874992          3879454          3858420
# contigs                   62               35               121
Largest contig              242390           472903           214259
Total length                3881484          3881006          3869184
GC (%)                      42.77            42.76            42.76
N50                         121210           214665           71381
N75                         72575            121455           39299
L50                         11               7                17
L75                         21               13               35
# N's per 100 kbp           0.00             193.58           0.00
# predicted genes (unique)  3712             3763             3741
# predicted genes (>= 0 bp) 3712             3763             3741
# predicted genes (>= 300 bp) 3356           3380             3356
# predicted genes (>= 1500 bp) 466           458              463
# predicted genes (>= 3000 bp) 42            40               41
Arast Pipeline: Job 6
```

The service has sorted the contig sets generated by the assemblies based on a variety of metrics including N50, ambiguous bases (N's), or mapping-based likelihood scores. The left-most assembly is considered to be the best, and the default output (contigs.fa) is a filtered version of this assembly. The numeric prefix for the assembly files is the job number. Please include this number when you submit error reports to us. The report file also includes detailed command line logs that may provide a clue.



18. Download the zip file for a visual comparison of the assemblies.



19. Unzip the analysis file on your computer and double-click on the report.html file.

You can mouse over the rows in the QUAST report to see how the assemblies compare in terms of each metric.

QUAST report

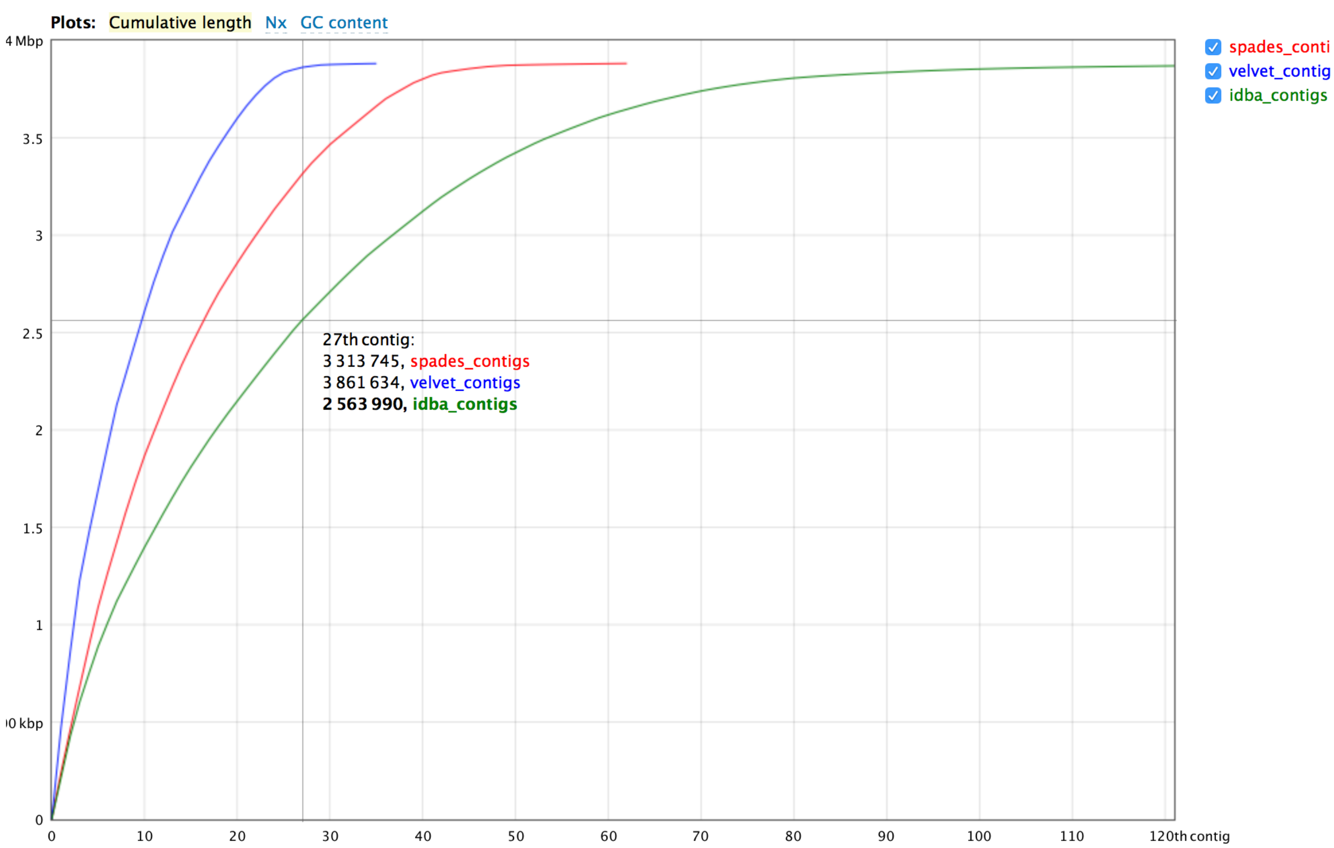
14 June 2015, Sunday, 20:35:10

All statistics are based on contigs of size >= 500 bp, unless otherwise noted (e.g., "# contig

Extended report worst............best

Statistics without reference	spades_contigs	velvet_contigs	idba_contigs
# contigs	62	35	121
Largest contig	242 390	472 903	214 259
Total length	3 881 484	3 881 109	3 869 184
N50	121 210	214 665	71 381

The "Cumulative length" and Nx plots are also helpful for deciding which assembly to use for your downstream analysis. (Nx is the largest contig length, L, such that using contigs of length >= L accounts for at least x% of the bases of the assembly. N50 is a special case of Nx.)

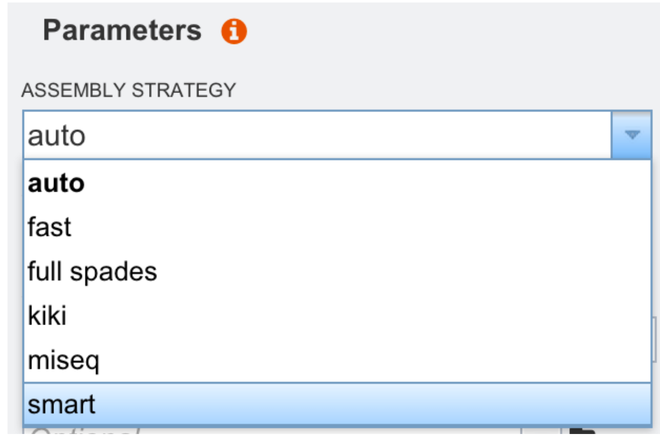


This concludes the tutorial on the basic usage of the assembly service.

Advanced assembly options

Curated Assembly Strategies

The Assembly service offers a list of curated assembly strategies (also called "recipes"). Each strategy is a workflow that invokes multiple tools. For example, the **auto** strategy currently tries three assemblers after correcting errors in the reads. The **smart** strategy is more sophisticated: it optimizes the *k*-mer length parameter, runs three assemblers, computes likelihood scores to evaluate the assemblies based on base quality as well as mapped paired end reads, and finally merges the top two assemblies.

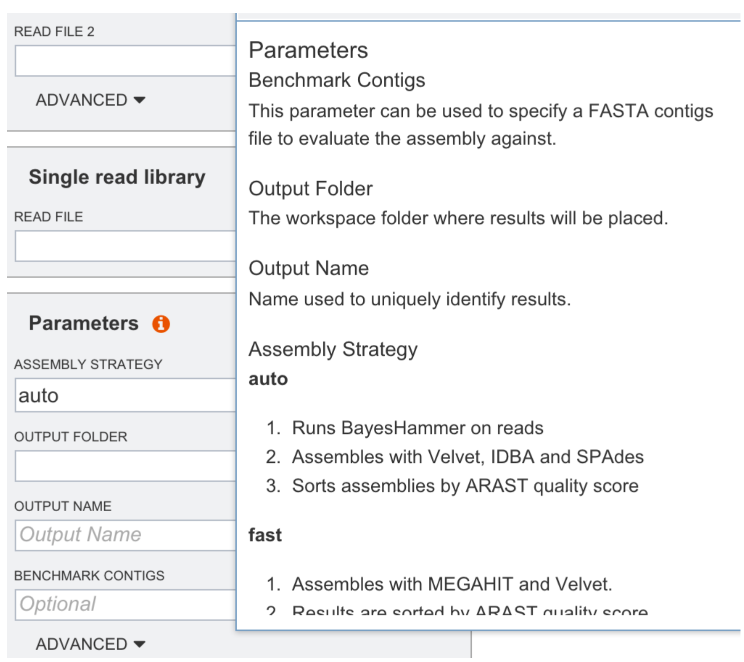


Here are the basic criteria for choosing which assembly strategy to use for your data:

- *auto* — the evolving default strategy recommended for most data
- *full spades* - runs the full SPAdes pipeline, one of the best assemblers for microbial genomes
- *fast* — ~2X faster than *auto*; suited for large genomes or simple microbial communities
- *kiki* — very fast but does not use paired end information; good for metagenome assembly
- *miseq* — good for Illumina MiSeq reads that are 250-350 bp long
- *smart* — the slowest and sometimes the most accurate

None of the recipes supports PacBio or Nanopore reads. However, we are testing two PacBio workflows and plan to support one in the next release. It will also be included in the *auto* and *smart* strategies.

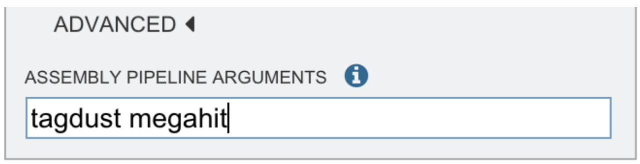
You can read the detailed workflows of the assembly strategies or helper text for other parameters by clicking on the information buttons.



Custom Assembly Pipelines

The service currently supports over 20 assemblers and tools (see Appendix), and its modular design allows for straightforward extension as sequencing technologies and analysis tools evolve. We have built a pipeline engine that enables you to mix and match approaches and evaluate a variety of customized pipelines on your datasets.

Each modules works at one of the three stages of the pipeline: preprocessing, assembly, and post-processing. In general, you can compose a pipeline by concatenating one or more preprocessing modules, one assembler, and optionally one post-processor.



Example 1:

tagdust velvet

This pipeline will simply run tagdust to remove adapter sequences in the reads and then assemble them with velvet. Note: quotes should not be used around the two modules as they have special meaning in pipeline syntax.

Example 2:

a6

You can also invoke an assembler that we have not included in our curated strategies. In this case, a6 is an assembler with its built-in preprocessing and post-processing steps.

Example 3:

"tagdust none" "megahit velvet" sspace

You can use quotes to specify alternative modules you would like to try at each step. This example will launch a cartesian combination of four parallel pipelines: tagdust+megahit+sspace, tagdust+velvet+sspace, megahit+sspace, velvet+sspace.

Note: The pipeline parameter overrides the assembly strategy parameter. Not all modules can be combined successfully.

Appendix

A. Assembly Strategies

auto

Runs BayesHammer on reads
Assembles with Velvet, IDBA and SPAdes
Sorts assemblies by ARAST quality score

fast

Assembles with MEGAHIT and Velvet.
Results are sorted by ARAST quality score.

full spades

Runs BayesHammer on reads
Assembles with SPAdes.

kiki

Runs the Kiki assembler

miseq

Runs Velvet with hash length 35.
Runs BayesHammer on reads and assembles with SPAdes with k up to 99.
Results are sorted by ARAST quality score.
Works for Illumina MiSeq reads.

smart

Runs BayesHammer on reads, Kmergenie to choose hash-length for Velvet
Assembles with Velvet, IDBA and SPAdes
Sorts assemblies by ALE score
Merges the two best assemblies with GAM-NGS

B. Assembly Modules

```
[Module] a5
  Description: A5 microbial assembly pipeline
  Version: 1.1
  Stages: preprocess,assembler,post-process
  Modules: tagdust,sga,idba,sspace
  Limitations: Updated to 2014 version; may crashes on some datasets (try a modified version in A6)
  References: doi:10.1371/journal.pone.0042304

[Module] a6
  Description: Modified A5 microbial assembly pipeline
  Version: 1.0
  Stages: preprocess,assembler,post-process
  Modules: tagdust,sga,idba,sspace
  Limitations: Does not support reads longer than 255 bp
  References: https://github.com/levinas/a5

[Module] bhammer
  Description: SPAdes component for quality control of sequence data
  Version: 1.0
  Stages: preprocess
  References: doi:10.1089/cmb.2012.0021

[Module] bowtie2
  Description: Bowtie2 aligner that maps reads to contigs
  Version: 1.0
  Stages: post-process
  References: doi:10.1038/nmeth.1923

[Module] bwa
  Description: BWA aligner that maps reads to contigs
  Version: 1.0
  Stages: post-process
  References: 10.1093/bioinformatics/btp324

[Module] fastqc
  Description: FastQC quality control tool for sequence data
  Version: 1.0
  Stages: preprocess
  References: http://www.bioinformatics.babraham.ac.uk/projects/fastqc

[Module] filter_by_length
  Description: Length-based sequencing reads filter and trimmer based on seqtk
  Version: 1.0
  Stages: preprocess
  References: https://github.com/levinas/seqtk
  Customizable parameters: default (available values)
                    end  =  200
                    min  =  250
                   sync  =  True

[Module] idba
  Description: IDBA iterative graph-based assembler for single-cell and standard data
  Version: 1.0
  Stages: assembler
  References: doi:10.1093/bioinformatics/bts174
  Customizable parameters: default (available values)
                  max_k  =  50
               scaffold  =  True

[Module] kiki
  Description: Kiki overlap-based parallel microbial and metagenomic assembler
  Version: 1.0
  Stages: assembler
  References: https://github.com/GeneAssembly/kiki
  Customizable parameters: default (available values)
       contig_threshold  =  800
                      k  =  21
```

```
[Module] megahit
  Description: An ultra-fast single-node solution for large and complex metagenomics assembly via succinct de
Bruijn graph
  Version: 1.0
  Base Version: 0.2.0
  Stages: assembler
  References: doi:10.1093/bioinformatics/btv033
  Customizable parameters: default (available values)
                      foo  =  bar

[Module] quast
  Description: QUAST assembly quality assessment tool (run by default)
  Version: 1.0
  Stages: post-process
  References: doi:10.1093/bioinformatics/btt086
  Customizable parameters: default (available values)
               min_contig  =  500

[Module] ray
  Description: Ray graph-based parallel microbial and metagenomic assembler
  Version: 1.0
  Stages: assembler
  References: doi:10.1186/gb-2012-13-12-r122
  Customizable parameters: default (available values)
                        k  =  31

[Module] reapr
  Description: REAPR assembly error recognizer using paired-end reads
  Version: 1.0
  Stages: post-process
  References: doi:10.1186/gb-2013-14-5-r47
  Customizable parameters: default (available values)
                        a  =  True

[Module] sga_ec
  Description: SGA component for error correction (runs subcommands: 'index' & 'correct')
  Version: 1.0
  Stages: preprocess
  References: doi:10.1101/gr.126953.111

[Module] sga_preprocess
  Description: SGA component for preprocessing reads (runs subcommand 'preprocess')
  Version: 1.0
  Stages: preprocess
  References: doi:10.1101/gr.126953.111
  Customizable parameters: default (available values)
               min_length  =  29
        permute_ambiguous  =  True
           quality_filter  =  20
             quality_trim  =  10

[Module] spades
  Description: SPAdes single-cell and standard assembler based on paired de Bruijn graphs
  Version: 1.2
  Base Version: 3.5.0
  Stages: preprocess,assembler
  Modules: bhammer
  References: doi:10.1089/cmb.2012.0021
  Customizable parameters: default (available values)
                  careful  =  False (bool)
      mismatch_correction  =  True (bool)
           only_assembler  =  True (bool)
              read_length  =  short (short, medium, medium2, long)

[Module] sspace
  Description: SSPACE pre-assembled contig scaffolder
  Version: 1.0
  Stages: post-process
  References: doi:10.1093/bioinformatics/btq683
  Customizable parameters: default (available values)
                        a  =  0.4
                   extend  =  False
```

```
                        k  =  -1
                        m  =  -1
          minimum_overlap  =  15
                        n  =  -1
                        x  =  0

[Module] swap
  Description: SWAP Assembler
  Version: 1.0
  Stages: assembler
  References: http://sourceforge.net/projects/swapassembler
  Customizable parameters: default (available values)
                        k  =  31

[Module] tagdust
  Description: TagDust sequencing artifacts remover
  Version: 1.0
  Stages: preprocess
  References: doi:10.1093/bioinformatics/btp527

[Module] trim_sort
  Description: DynamicTrim and LengthSort from SolexaQA
  Version: 1.0
  Stages: preprocess
  References: doi:10.1186/1471-2105-11-485
  Customizable parameters: default (available values)
                   length  =  25
               probcutoff  =  0.05

[Module] velvet
  Description: Velvet de-bruijn graph based assembler
  Version: 1.0
  Base Version: 1.2.10
  Stages: assembler
  References: doi:10.1101/gr.074492.107
  Customizable parameters: default (available values)
              auto_insert  =  False
              hash_length  =  29
```