Detailed descriptions of each module.

The Read QC module is meant to pre-process raw Illumina sequencing reads in preparation for assembly and alignment. The raw reads are trimmed based on adapted content and PHRED scored with the default setting of Trim-galore, ensuring that only high-quality sequences are left. Then reads are then aligned to the human genome with bmtagger, and any human reads are removed from the metagenomic data to remove human contamination. Read pairs where only one read was aligned to the human genome are also removed. Finally, FASTQC is used to generate quality reports of the raw and final read sets in order to assess read quality improvement. The user has control over which of the above features he wishes to use.

The Assembly module allows the user to assemble a set of metagenomic reads with either metaSPAdes or MegaHit (default). While metaSPAdes results in a superior assembly in most samples, MegaHit is much more memory efficient, faster, and scales well with large datasets. In addition to simplifying parameter selection for the user, this module also sorts and formats the MegaHit assembly in a way that makes it easier to inspect. The contigs are sorted by length and their naming is changed to resemble that of SPAdes, including the contig ID, length, and coverage. Finally, short scaffolds are discarded (<1000bp), and an assembly report is generated with QUAST.

The Kraken module takes in any number of fastq or fasta files, classifies the contained sequences with KRAKEN, and reports the taxonomy distribution in a kronagram using KronaTools. If the sequences passed to the module belong to an assembly and follow the contig naming convention of the Assembly module, the taxonomy of each contig is weighted based on its length and coverage [weight=coverage\*length]. The classifications of the sequences are then summarized in a format that KronaTools’ ktImportText function recognizes, and a final kronagram in html format is made with all the samples.

The Binning module is meant to be a convenient wrapper around three metagenomic binning software: MaxBin2, metaBAT2, and CONCOCT. First the metagenomic assembly is indexed with bwa-index, and then paired end reads from any number of samples are aligned to it. The alignments are sorted and compressed with samtools, and library insert size statistics are also gathered at the same time (insert size average and standard deviation). metaBAT2’s jgi\_summarize\_bam\_contig\_depths function is used to generate contig adundance table, and it is then converted into the correct format for each of the three binners to take as input. After MaxBin2, metaBAT2, and CONCOCT finish binning the contigs with default settings (the user can specify which software he wants to bin with), the final bins folders are created with formatted bin fasta files for easy inspection. Optionally, the user can chose to immediately run CheckM on the bins to determine the success of the binning. CheckM’s lineage\_wf function is used to predict essential genes and estimate the completion and contamination of each bin, and a custom script is used to generate an easy to view report on each bin set.

The Bin\_refinement module utilizes a hybrid approach to take in two or three bin sets that were obtained with different software (or the same software with different parameters) and produces a consolidated, improved bin set. First, binning\_refiner is used to cready hybridized bins from every possible combination of sets. If there were three bin sets: A, B, and C, then the following hybrid sets will be produced with binning\_refiner: AB, BC, AC, and ABC. CheckM is then run to evaluate the completion and contamination of the bins in each of the 7 bin sets (3 originals, 4 hybridized). The bins sets are then iteratively compared to each other, and each pair is consolidated into an improved bin set. To do this, the same bin is identified within the two bin sets based on a minimum of 80% overlap in genome length, and the better bin is determined based on which bin has the higher score. The scoring function is S=Completion-5\*Contamination. After all bin sets are incorporated into the consolidated bin collection, a de-replication function removes any duplicate contigs. If a contig is present in more than one bin, it is removed from all but the best bin (based on scoring function). CheckM is then run on the final bin set and a final report file is generated showing the completion, contamination, and other statistics generated by CheckM for each bin. Completion and contamination rank plots are also generated to evaluate the success of the Bin\_refinement module, and compare its output to the quality of the original bins.

The Reassemble\_bins module aims to improve a set of bins by finding reads that align to them and re-assembling them. First, bwa is used to index the entire metagenomic assembly and align fastq reads back to it. Reads mapping back to contings belonging to metagenomic bins are stored in separate fastq files. If only one read mate is aligned, the pair still gets recruited into that bin. For each bin, two sets of reads are stored – reads mapping perfectly (strict), and reads mapping with less than 3 mismatches (permissive). The each set of reads is then reassembled with SPAdes with the --carefull setting, and short contigs (<1000bp) are removed. CheckM is then used to evaluate the completion and contamination of each of the three versions of each bin – the original bin, the strict re-assembled bin, and permissive reassembled bin. The best version is chosen based on the highest score [S=Completion-5\*Contamination], and added to the final bin set. The final bins set it then re-evaluated with CheckM, and summary statistics are generated. Additionally, a completion and contamination rank plots is generated to evaluate the improvements in the bin sets following reassembly.

The Quant\_bins module quickly estimates the abundance of bins across a number of samples. Salmon is used to index the entire metagenomic assembly, and then align reads from each sample back to the assembly. Coverage tables are generated estimating the abundance of each contig in each sample. The average abundance of each bin in each sample is calculated by taking the length-weighted average of the bins’s contig abundances. A final bin abundance table is made, and a clustered heatmap is generated to visualize bin abundance variation across samples.

The Blobology module creates blobplots (a GC vs abundance plot of all the contigs) of a metagenomic assembly, and annotates it with phylogenetic information or bin information. The assembly contigs can be randomly sub-sampled based on user preference.

First, the taxonomy of each contig is estimated with Mega-BLAST by taking the top-most confident alignment against the NCBI\_nt database. Then the assembly is indexed and the reads from any number of samples are aligned against it with bowtie2. Finally Blobology’s gc\_cov\_annotate.pl function is used to generate a blobplot file with the GC, coverage (in all samples), and taxonomy of each contig. If the user specified a set of bins to annotate, the contigs are also annotated with the bins they belong to. Finally, Blobology’s makeblobplot.R function is used to make the blobplots of the contigs across all the provided samples, with taxonomic annotation (at the super-kingdom, phylum, and order level) and with binning annotation (which contigs belong to which bin and the phylum taxonomy of only binned contigs).

The Classify\_bins module is a conservative, but accurate way to assign taxonomy to a set of metagenomic bins. First, the contigs in all bins are combined into one file, and MegaBLAST is used to align the contigs to the NCBI\_nt database. The alignment results are then used by taxator-kt to estimate the most likely taxonomy of each contig. Finally, the overall most likely taxonomy of each metagenomic bin is estimated from individual contig taxonomy predictions. Taxonomy of each contig are added to a phylogenetic tree, adding weight to each branch based on the length of that contig. The tree is then traversed from the root, going down a taxonomic rank only if the weight of the next branch is >50% of the current branch, indicating a minimum confidence in that prediction. Once no further taxonomic rank can be estimated, the final taxonomy of that bin is reported.