ATLAS Documentation

Release 2.0

Joe Brown

Feb 25, 2020

Setup

CHAPTER 1

Quick Start

Three commands to start analysing your metagenome data:

```
conda install -c bioconda -c conda-forge metagenome-atlas
atlas init --db-dir databases path/to/fastq/files
atlas run
```

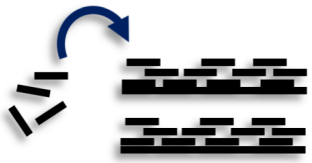
All databases and dependencies are installed on the fly in the directory `--db-dir`. You want to run this three commands on the example_data on the GitHub repo. If you have more time, then we recommend you to configure atlas according to your needs:

- check the samples.tsv
- edit the *config.yaml*
- run atlas on a cluster system

1 Quality Control

- PCR duplicates removal
- Quality trimming
- Host removal
- Common contaminant removal

➢ **QC reads**



2 Assembly

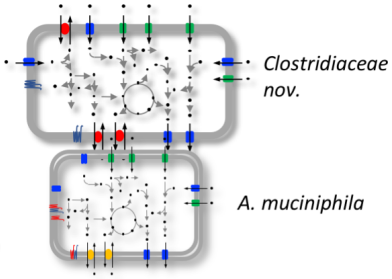
- Error correction
- Paired-end merging
- Assembly (metaSpades/megahit)
- Post-filtering

➢ **High-quality Scaffolds**

3 Genomic Binning

- Binning (metabat, maxbin2)
- Quality Assessment (checkM)
- Bin refining (DAS Tool)
- Dereplication (dRep)
- Quantification
- Robust taxonomic classification (CAT)

➢ **Genomes**
➢ **Abundances**



4 Annotation

- Gene prediction (prodigal)
- Cluster redundant genes (linclust/ cd-hit)
- Annotation (eggNOG)

➢ **Comparable gene catalog**

CHAPTER 2

Documentation

2.1 Install

2.1.1 A. Use conda

You need to install [anaconda](http://anaconda.org/) or miniconda. We recommend you to create a conda environment, then install metagenome-atlas:

```
conda create -y -n atlasenv
source activate atlasenv
conda install -y -c bioconda -c conda-forge metagenome-atlas
```

2.1.2 B. Install the development version from GitHub

Atlas is still under active development, therefore you may want to install the up to date atlas from GitHub.

get code from GitHub:

```
git clone https://github.com/metagenome-atlas/atlas.git
cd atlas
```

Create a conda environment with all primary dependencies. All further dependencies are installed on the fly:

```
conda env create -f atlasenv.yml
source activate atlasenv
```

Install atlas:

```
pip install --editable .
```

Now you should be able to run atlas:

```
atlas init --db-dir databases path/to/fastq/files
atlas run
```

2.1.3 C. Use docker container

We recommend to use the conda package as it allows deployment on clusters. However, if you want to directly start using atlas on a small metagenome you can use the docker container:

```
docker pull metagenomeatlas/atlas
```

Go to a directory on your filesystem where you have the fastq files in a subfolder, e.g. in `reads` Your present working directory will be mounted on `/WD` in the docker container.

The docker container contains all the dependencies and some of the databases in `/databases` . The databases for functional and taxonomic annotation are downloaded while running. To not loose the databases after exiting the docker we recommend to mount them also on your disk.

Create:

```
mkdir -p AtlasDB/GTDB-TK AtlasDB/EggNOGV2
```

Then run the docker:

```
docker run -i -u $(id -u):$(id -g) -v $(pwd):/WD -v $(pwd)/AtlasDB/EggNOGV2/:/
↪databases/EggNOGV2 -v $(pwd)/AtlasDB/GTDB-TK/:/databases/GTDB-TK -t metagenomeatlas/
↪atlas:latest /bin/bash
```

Inside the docker you can run atlas as folows:

```
atlas init -db-dir /databases /WD/reads
```

This should create a sample.tsv and a config.yaml, whcih you can edit on your system. Important don't forget to align the memory of your computer with the memory defined in the config file.

after that run:

```
atlas run all
```

2.2 Usage

Now let's apply atlas on your data.

2.2.1 atlas init

```
Usage: atlas init [OPTIONS] PATH_TO_FASTQ

  Write the file CONFIG and complete the sample names and paths for all
  FASTQ files in PATH.

  PATH is traversed recursively and adds any file with '.fastq' or '.fq' in
  the file name with the file name minus extension as the sample ID.
```

(continues on next page)

(continued from previous page)

```
Options:
  -d, --db-dir PATH               location to store databases (need ~50GB)
                                  [default: /Users/silas/Documents/GitHub/atla
                                  s/databases]
  -w, --working-dir PATH          location to run atlas
  --assembler [megahit|spades]    assembler  [default: spades]
  --data-type [metagenome|metatranscriptome]
                                  sample data type  [default: metagenome]
  --interleaved-fastq             fastq files are paired-end in one files
                                  (interleaved)
  --threads INTEGER               number of threads to use per multi-threaded
                                  job
  --skip-qc                       Skip QC, if reads are already pre-processed
  -h, --help                      Show this message and exit.
```

This command creates a `samples.tsv` and a `config.yaml` in the working directory.

Have a look at them with a normal text editor and check if the samples names are inferred correctly. Samples should be alphanumeric names and cam be dash delimited. Underscores should be fine too. See the `example sample table`

The `BinGroup` parameter is used during the genomic binning. In short: all samples in which you expect the same strain to be found should belong to the same group, e.g. all metagenome samples from mice in the same cage. If you want to use *long reads* for a hybrid assembly, you can also specify them in the sample table.

You should also check the `config.yaml` file, especially:

- You may want to add ad *host genomes* to be removed.
- You may want to change the resources configuration, depending on the system you run atlas on.

Details about the parameters can be found in the section *Configure Atlas*

2.2.2 atlas run

```
Usage: atlas run [OPTIONS]
                 [[qc|assembly|binning|genomes|genecatalog|None|all]]
                 [SNAKEMAKE_ARGS]...

  Runs the ATLAS pipline

  By default all steps are executed but a sub-workflow can be specified.
  Needs a config-file and expects to find a sample table in the working-
  directory. Both can be generated with 'atlas init'

  Most snakemake arguments can be appended to the command for more info see
  'snakemake --help'

  For more details, see: https://metagenome-atlas.readthedocs.io

Options:
  -w, --working-dir PATH  location to run atlas.
  -c, --config-file PATH  config-file generated with 'atlas init'
  -j, --jobs INTEGER      use at most this many jobs in parallel (see cluster
                          submission for mor details).  [default: 8]
  --profile TEXT          snakemake profile e.g. for cluster execution.
```

(continues on next page)

(continued from previous page)

```
  -n, --dryrun            Test execution.  [default: False]
  -h, --help              Show this message and exit.
```

`atlas run` need to know the working directory with a `samples.tsv` inside it.

Take note of the `--dryrun` parameter, see the section snakemake for other handy snakemake arguments.

We recommend to use atlas on a *Cluster execution* system, which can be set up in a view more commands.

2.3 Cluster execution

Thank the underlying, Snakemake system, atlas can be executed on virtually all clusters and cloud systems. Instead of running all steps of the pipeline in one cluster job atlas can automatically submit each step to your cluster system, specifying the necessary threads, memory, and runtime, based on the values in the config file. Atlas periodically checks the status of each cluster job and can re-run failed jobs or continue with other jobs.

If you have a common cluster system (Slurm, LSF, PBS ...) we have an easy set up (see below). Otherwise, if you have an other cluster system, fill in an GitHub issue (feature request) so we can help you bringing the magic of atlas to your cluster system. For more information about cluster- and cloud submission, have a look at the snakemake doc.

2.3.1 Set up of cluster execution

You need cookiecutter to be installed, which comes with atlas

Then run:

```
cookiecutter --output-dir ~/.config/snakemake https://github.com/metagenome-atlas/
↪clusterprofile.git
```

This opens a interactive shell dialog and ask you for the name of the profile and your cluster system. We recommend you to keep the default name `cluster`. The profile supports `slurm`,‘‘lsf‘‘ and `pbs`.

The resources (threads, memory and time) are defined in the atlas config file (hours and GB).

If you need to specify **queues or accounts** you can do this for all rules or for specific rules in the `~/.config/snakemake/cluster/cluster_config.yaml`. In addition, using this file you can overwrite the resources defined in the config file.

Example for `cluster_config.yaml` with queues defined:

```
__default__:
# default parameter for all rules
  queue: normal
  nodes: 1


# The following rules in atlas need need more time/memory.
# If you need to submit them to different queues you can configure this as outlined.

run_megahit:
  queue: bigmem
run_spades:
  queue: bigmem

This rules can take longer
```

(continues on next page)

(continued from previous page)

```
run_checkm_lineage_wf:
  queue: long
```

Now, you can run atlas on a cluster with:

```
atlas run <options> --profile cluster
```

As the whole pipeline can take several days, I usually run this command in a screen on the head node, even when system administrators don't like that. On the head node atlas doesn't uses much threads/memory. It only schedules the jobs and does some steps which combine tables. Obviously you can also submit the atlas command as a long lasting job.

If a job fails, you will find the "external jobid" in the error message. You can investigate the job via this ID.

2.3.2 Useful command line options

The atlas argument `--jobs` now becomes the number of jobs simultaneously submitted to the cluster system. You can set this as high as 99 if you don't have a problem with your colleges of over-using the cluster system.

In the case of an failed job, `--keep-going` (default false) allows atlas to continue with independent steps.

2.4 Cloud execution

Atlas, like any other snakemake pipeline can thanks also easily be submitted to cloud systems. I let look at the snakemake doc. Keep in mind any snakemake comand line argument can just be appended to the atlas command.

2.5 Local execution

The number of threads used **for each step** can be configured in the config file:

```
threads: 8
assembly_threads: 8
```

For local execution the `--jobs` command line arguments defines the number of threads used in total and by default is set to the number of processor of your system. If you have less core available than specified in the config file. The jobs are downscaled. If you have more Atlas tries to start multiple jobs, to optimally use the cores on you machine. If you don't want that Atlas uses all the available cores then you can specify this with the `--jobs` command line argument.

2.6 Test atlas

If you want to test atlas on a small example data here is a two sample, three genome minimal metagenome dataset, to test atlas. Even when atlas will run faster on the test data, it will anyway download all the databases and requirements, for the a complete run, which can take a certain amount of time and especially disk space (>100Gb).

The database dir of the test run should be the same as for the later atlas executions.

The example data can be downloaded as following:

```
git clone https://github.com/metagenome-atlas/example_data.git
```

We initialize a atlas working directory `testrun` using the test reads. The reads are paired end stored in one file therefore we set the `--interleaved-fastq` flag. The test samples don't require a lot of threads (set to 2), they do require However some memory (~60GB).:

```
atlas init --db-dir databases --working-dir testrun --interleaved-fastq --threads=2␣
↪example_data/reads/test
```

After the set up you can run:

```
atlas run --working-dir testrun
```

2.6.1 Test atlas using docker container

Testing atlas using the docker container works similarly to the above example.

Create a working directory:

```
mkdir DockerTest; cd DockerTest
```

get the example data:

```
git clone https://github.com/metagenome-atlas/example_data.git
```

Set up the docker as explained here and run it:

```
mkdir -p AtlasDB/GTDB-TK AtlasDB/EggNOGV2
docker run -i -u $(id -u):$(id -g) -v $(pwd):/WD -v $(pwd)/AtlasDB/EggNOGV2/:/
↪databases/EggNOGV2 -v $(pwd)/AtlasDB/GTDB-TK/:/databases/GTDB-TK -t metagenomeatlas/
↪atlas:latest /bin/bash
```

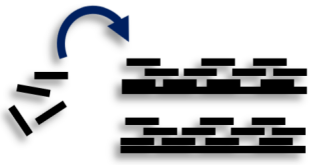
Inside the docker you have access to the `example_data` and you can initialize and run atlas test example as above.

2.7 The Atlas pipeline

1 Quality Control

- PCR duplicates removal
- Quality trimming
- Host removal
- Common contaminant removal

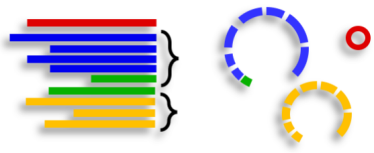
➢ **QC reads**



2 Assembly

- Error correction
- Paired-end merging
- Assembly (metaSpades/megahit)
- Post-filtering

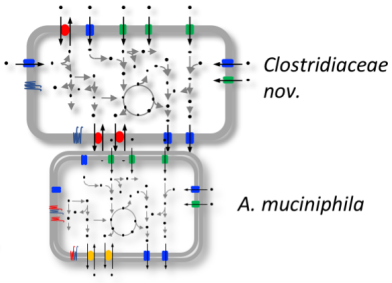
➢ **High-quality Scaffolds**



3 Genomic Binning

- Binning (metabat, maxbin2)
- Quality Assessment (checkM)
- Bin refining (DAS Tool)
- Dereplication (dRep)
- Quantification
- Robust taxonomic classification (CAT)

➢ **Genomes**
➢ **Abundances**



4 Annotation

- Gene prediction (prodigal)
- Cluster redundant genes (linclust/ cd-hit)
- Annotation (eggNOG)

➢ **Comparable gene catalog**

2.8 Expected output

2.8.1 Quality control

```
atlas run qc
#or
atlas run all
```

Runs quality control of single or paired end reads and summarizes the main QC stats in reports/QC_report.html.

Per sample it generates:

- `{sample}/sequence_quality_control/{sample}_QC_{fraction}.fastq.gz`
- Various quality stats in {sample}/sequence_quality_control/read_stats

Fractions:

When the input was paired end, we will put out three the reads in three fractions R1,R2 and se The se are the paired end reads which lost their mate during the filtering. The se are seamlessly integrated in the next steps.

2.8.2 Assembly

```
atlas run assembly
#or
atlas run all
```

Besides the reports/assembly_report.html this rule outputs the following files per sample:

- `{sample}/{sample}_contigs.fasta`
- `{sample}/sequence_alignment/{sample}.bam`
- `{sample}/assembly/contig_stats/final_contig_stats.txt`

2.8.3 Binning

```
atlas run binning
#or
atlas run all
```

When you use different binners (e.g. metabat, maxbin) and a binner-reconciliator (e.g. DAS Tool), then Atlas will produce for each binner and sample:

- `{sample}/binning/{binner}/cluster_attribution.tsv`

which shows the attribution of contigs to bins. For the final_binner it produces the

- `reports/bin_report_{binner}.html`

See an example as a summary of the quality of all bins.

2.8.4 Genomes

```
atlas run genomes
#or
atlas run all
```

As the binning can predict several times the same genome it is recommended to de-replicate these genomes. For now we use DeRep to filter and de-replicate the genomes. The Metagenome assembled genomes are then renamed, but we keep mapping files.

- `genomes/Dereplication`
- `genomes/clustering/contig2genome.tsv`

- genomes/clustering/allbins2genome.tsv

The fasta sequence of the dereplicated and renamed genomes can be found in genomes/genomes and their quality estimation are in genomes/checkm/completeness.tsv. The quantification of the genomes can be found in:

- genomes/counts/median_coverage_genomes.tsv
- genomes/counts/raw_counts_genomes.tsv

See in Atlas example how to analyze these abundances.

The predicted genes and translated protein sequences are in genomes/annotations/genes.

Taxonomic adnnotation

```
annotations:
  - gtdb_tree
  - gtdb_taxonomy
  - checkm_tree
  - checkm_taxonomy
```

Different annotations can be turned on and off in the config file under the heading annotations: A taxonomy for the dereplicated genomes is proposed GTDB. The results can be found in genomes/taxonomy. The genomes are placed in a phylogenetic tree separately for bacteria and archaea (if there are any) using the GTDB markers. In addition a tree for bacteria and archaea can be generated based on the checkm markers. All trees are properly rooted using the midpoint. The files can be found in genomes/tree

2.8.5 Gene Catalog

```
atlas run all
# or
atlas run genecatalog
```

The gene catalog takes either genes predicted from the genomes or all genes predicted on the contigs and clusters them according to the configuration. This rule produces the following output file for the whole dataset.

- Genecatalog/gene_catalog.fna
- Genecatalog/gene_catalog.faa
- Genecatalog/annotations/eggNog.tsv.gz

2.9 Configure Atlas

2.9.1 Example config file

```
##################################################################
####                  ________   _                  ______      ####
####          /\      |__   __| | |          /\       / _____|     ####
####         /  \        | |    | |         /  \     | (____       ####
####        / /\ \       | |    | |        / /\ \     \____ \      ####
####       / ____ \      | |    | |_____  / ____ \    _____) |     ####
####      /_/    \_\     |_|    |_______|/_/    \_\  |______/      ####
####                                                              ####
```

(continues on next page)

(continued from previous page)

```
#####################################################################

#  For more details about the config values see:
#  https://metagenome-atlas.rtfd.io


########################
# Execution parameters
########################
# threads and memory (GB) for most jobs especially from BBtools, which are memory␣
↪demanding
threads: 8
mem: 60

# threads and memory for jobs needing high amount of memory. e.g GTDB-tk,checkm or␣
↪assembly
large_mem: 100
large_threads: 8
assembly_threads: 8
assembly_memory: 250

#Runtime only for cluster execution
runtime: #in h
    default: 5
    assembly: 48
    long: 24

# Local directory for temp files, useful for cluster execution without shared file␣
↪system
tmpdir: /tmp
# directory where databases are downloaded with 'atlas download'
database_dir: databases

########################
# Quality control
########################
data_type: metagenome # metagenome or metatranscriptome
interleaved_fastqs: false

# remove (PCR)-duplicated reads using clumpify
deduplicate: true
duplicates_only_optical: false
duplicates_allow_substitutions: 2

# used to trim adapters from reads and read ends
preprocess_adapters: /path/to/databases/adapters.fa
preprocess_minimum_base_quality: 10
preprocess_minimum_passing_read_length: 51
# 0.05 requires at least 5 percent of each nucleotide per sequence
preprocess_minimum_base_frequency: 0.05
preprocess_adapter_min_k: 8
preprocess_allowable_kmer_mismatches: 1
preprocess_reference_kmer_match_length: 27
# error correction where PE reads overlap
error_correction_overlapping_pairs: true
#contamination references can be added such that -- key: /path/to/fasta
contaminant_references:
```

(continues on next page)

(continued from previous page)

```
  PhiX: /path/to/databases/phiX174_virus.fa
#  host:/path/to/host_genome.fasta

# We won't allow large indels
contaminant_max_indel: 20
contaminant_min_ratio: 0.65
contaminant_kmer_length: 13
contaminant_minimum_hits: 1
contaminant_ambiguous: best


########################
# Pre-assembly-processing
########################

error_correction_before_assembly : true

# join R1 and R2 at overlap; unjoined reads are still utilized
merge_pairs_before_assembly : true
merging_k: 62
# extend reads while merging to this many nucleotides
merging_extend2: 40
# Iterations are performed until extend2 x iterations
merging_flags: "ecct iterations=5"

########################
# Assembly
########################
# megahit OR spades
assembler: spades

# Megahit
#-----------
# 2 is for metagenomes, 3 for genomes with 30x coverage
megahit_min_count: 2
megahit_k_min: 21
megahit_k_max: 121
megahit_k_step: 20
megahit_merge_level: 20,0.98
megahit_prune_level: 2
megahit_low_local_ratio: 0.2
# ['default','meta-large','meta-sensitive']
megahit_preset: default

# Spades
#------------
spades_skip_BayesHammer: true
spades_use_scaffolds: false # use contigs
#Comma-separated list of k-mer sizes to be used (all values must be odd, less than␣
↪128 and listed in ascending order).
spades_k: auto
spades_preset: meta    # meta, ,normal, rna  single end libraries doesn't work for␣
↪metaspades
spades_extra: ""
longread_type: none # [none,"pacbio", "nanopore", "sanger", "trusted-contigs",
↪"untrusted-contigs"]
# Preprocessed long reads can be defined in the sample table with 'longreads' , for␣
↪more info see the spades manual
```

(continues on next page)

(continued from previous page)

```
# Filtering
#------------
# filter out assembled noise
# this is more important for assemblys from megahit
filter_contigs: true
prefilter_minimum_contig_length: 200
# trim contig tips
contig_trim_bp: 0
# require contigs to have read support
minimum_average_coverage: 1
minimum_percent_covered_bases: 20
minimum_mapped_reads: 0
# after filtering
minimum_contig_length: 300


########################
# Quantification
########################

# Mapping reads to contigs
#--------------------------
contig_min_id: 0.76
contig_map_paired_only: true
contig_max_distance_between_pairs: 1000
maximum_counted_map_sites: 10

########################
# Binning
########################

final_binner: DASTool             # [DASTool or one of the binner, e.g. maxbin]

binner:                           # If DASTool is used as final_binner, use␣
↪predictions of this binners
  - metabat
  - maxbin


metabat:
  sensitivity: sensitive
  min_contig_length: 1500 # metabat needs >1500

maxbin:
  max_iteration: 50
  prob_threshold: 0.9
  min_contig_length: 1000

DASTool:
  search_engine: 'diamond'
  score_threshold: 0.5              #Score threshold until selection algorithm will␣
↪keep selecting bins [0..1].

genome_dereplication:
  ANI: 0.95
  overlap: 0.6
```

(continues on next page)

(continued from previous page)

```
  opt_parameters: ""
  filter:
      noFilter: false
      length: 5000
      completeness: 50
      contamination: 10
  score:
      completeness: 1
      contamination: 5
      N50: 0.5
      length: 0

rename_mags_contigs: true        #Rename contigs of representative MAGs

########################
# Annotations
########################

annotations:
  - gtdb_tree
  - gtdb_taxonomy
  - genes
#  - checkm_taxonomy
#  - checkm_tree

########################
# Gene catalog
########################
genecatalog:
  source: contigs               # [contigs, genomes] Predict genes from all contigs␣
↪or only from the representative genomes
  clustermethod: linclust       # [cd-hit-est or mmseqs or linclust] see mmseqs for␣
↪more details
  minlength_nt: 100
  minid: 0.95                   # min id for gene clustering for the main gene␣
↪catalog used for annotation
  coverage: 0.9
  extra: ""
  SubsetSize: 500000
```

2.9.2 Remove reads from Host

One of the most important steps in the Quality control is to remove host genome. You can add any number of genomes to be removed.

We recommend you to use genomes where repetitive sequences are masked. See here for more details human genome.

2.9.3 Detailed configuration

Quality control of reads

Adapter Trimming

FASTA file paths for adapter sequences to be trimmed from the sequence ends.

We provide the adapter reference FASTA included in *bbmap* for various

```
preprocess_adapters: /database_dir/adapters.fa
```

Quality Trimming

Trim regions with an average quality below this threshold. Higher is more stringent.

```
preprocess_minimum_base_quality: 10
```

Adapter Trimming at Read Tips

Allow shorter kmer matches down to *mink* at the read ends. 0 disables.

```
preprocess_adapter_min_k: 8
```

Allowable Mismatches in Adapter Hits

Maximum number of substitutions between the target adapter kmer and the query sequence kmer. Lower is more stringent.

```
preprocess_allowable_kmer_mismatches: 1
```

Contaminant Kmer Length

Kmer length used for finding contaminants. Contaminant matches shorter than this length will not be found.

```
preprocess_reference_kmer_match_length: 27
```

Read Length Threshold

This is applied after quality and adapter trimming have been applied to the sequence.

```
preprocess_minimum_passing_read_length: 51
```

Sequence Complexity Filter

Require this fraction of each nucleotide per sequence to eliminate low complexity reads.

```
preprocess_minimum_base_frequency: 0.05
```

Contamination Parameters

Contamination reference sequences in the form of nucleotide FASTA files can be provided and filtered from the reads using the following parameters.

If 'rRNA' is defined, it will be added back to metagenomes but not to metatranscriptomes. Additional references can be added arbitrarily, such as::

```
contaminant_references:
    rRNA: /database_dir/silva_rfam_all_rRNAs.fa
    phiX: /database_dir/phiX174_virus.fa
```

Don't look for indels longer than this:

```
contaminant_max_indel: 20
```

Fraction of max alignment score required to keep a site:

```
contaminant_min_ratio: 0.65
```

mapping kmer length; range 8-15; longer is faster but uses more memory; shorter is more sensitive:

```
contaminant_kmer_length: 12
```

Minimum number of seed hits required for candidate sites:

```
contaminant_minimum_hits: 1
```

Set behavior on ambiguously-mapped reads (with multiple top-scoring mapping locations):

- best (use the first best site)
- toss (consider unmapped, retain in reads for assembly)
- random (select one top-scoring site randomly)
- all (retain all top-scoring sites)

```
contaminant_ambiguous: best
```

For human reads we recommend to download the genome [here](), where contaminants and low complexity regions were masked.

Pre-Assambly-processing

Normalization Parameters

To improve assembly time and often assemblies themselves, coverage is normalized across kmers to a target depth and can be set using:

```
# kmer length over which we calculated coverage
normalization_kmer_length: 21
# the normalized target coverage across kmers
normalization_target_depth: 100
# reads must have at least this many kmers over min depth to be retained
normalization_minimum_kmers: 8
```

Error Correction

Optionally perform error correction using `tadpole.sh` from BBTools:

```
perform_error_correction: true
```

Assembly Parameters

Assembler

Currently, the supported assemblers are 'spades' and 'megahit' with the default setting of:

```
assembler: megahit
```

Both assemblers have settings that can be altered in the configuration:

```
# minimum multiplicity for filtering (k_min+1)-mers
megahit_min_count: 2
# minimum kmer size (<= 255), must be odd number
megahit_k_min: 21
# maximum kmer size (<= 255), must be odd number
megahit_k_max: 121
# increment of kmer size of each iteration (<= 28), must be even number
megahit_k_step: 20
# merge complex bubbles of length <= l*kmer_size and similarity >= s
megahit_merge_level: 20,0.98
# strength of low depth pruning (0-3)
megahit_prune_level: 2
# ratio threshold to define low local coverage contigs
megahit_low_local_ratio: 0.2
# minimum length of contigs (after contig trimming)
minimum_contig_length: 200
# comma-separated list of k-mer sizes (must be odd and less than 128)
spades_k: auto
```

Contig Filtering

After assembly, contigs can be filtered based on several metrics:

```
# Discard contigs with lower average coverage.
minimum_average_coverage: 5
# Discard contigs with a lower percent covered bases.
minimum_percent_covered_bases: 40
# Discard contigs with fewer mapped reads.
minimum_mapped_reads: 0
# Trim the first and last X bases of each sequence.
contig_trim_bp: 0
```

Use long reads with spades

Limitation: Hybrid assembly of long and short reads is supported with spades and metaSpades. However metaSpades needs a paired-end short-read library.

The path of the (preprocessed) long reads should be added manually to the the sample table under a new column heading 'longreads'.

In addition the type of the long reads should be defined in the config file: ` longread_type` one of ["pacbio", "nanopore", "sanger", "trusted-contigs", "untrusted-contigs"]