
viral-ngs Documentation

Release v1.1.0

Broad Institute Viral Genomics

2015-09-02

1	Contents	1
1.1	Description of the methods	1
1.2	Installation	2
1.3	Command line tools	3
1.4	Using the Snakemake pipelines	41

Contents

1.1 Description of the methods

1.1.1 Taxonomic read filtration

Human, contaminant, and duplicate read removal

The assembly pipeline begins by depleting paired-end reads from each sample of human and other contaminants using [BMTAGGER](#) and [BLASTN](#), and removing PCR duplicates using M-Vicuna (a custom version of [Vicuna](#)).

Taxonomic selection

Reads are then filtered to a genus-level database using [LASTAL](#), quality-trimmed with [Trimmomatic](#), and further deduplicated with [PRINSEQ](#).

1.1.2 Viral genome analysis

Viral genome assembly

The filtered and trimmed reads were subsampled to at most 100,000 pairs. *de novo* assembly is performed using [Trinity](#). Reference-assisted assembly improvements follow (contig scaffolding, orienting, etc.) with [VFAT](#) (which relies on [MUSCLE](#)).

Each sample's reads are aligned to its *de novo* assembly using [Novoalign](#) and any remaining duplicates were removed using [Picard](#) MarkDuplicates. Variant positions in each assembly were identified using [GATK](#) IndelRealigner and UnifiedGenotyper on the read alignments. The assembly was refined to represent the major allele at each variant site, and any positions supported by fewer than three reads were changed to N.

This align-call-refine cycle is iterated twice, to minimize reference bias in the assembly.

Intrahost variant identification

Intrahost variants (iSNVs) were called from each sample's read alignments using [V-Phaser2](#) and subjected to an initial set of filters: variant calls with fewer than five forward or reverse reads or more than a 10-fold strand bias were eliminated. iSNVs were also removed if there was more than a five-fold difference between the strand bias of the variant call and the strand bias of the reference call. Variant calls that passed these filters were additionally subjected to a 0.5% frequency filter. The final list of iSNVs contains only variant calls that passed all filters in two separate

library preparations. These files infer 100% allele frequencies for all samples at an iSNV position where there was no intra-host variation within the sample, but a clear consensus call during assembly. Annotations are computed with `snpEff`.

1.1.3 Taxonomic read identification

Nothing here at the moment. That comes later, but we will later integrate it when it's ready.

1.1.4 Cloud compute implementation

This assembly pipeline is also available via the DNAnexus cloud platform. RNA paired-end reads from either HiSeq or MiSeq instruments can be securely uploaded in FASTQ or BAM format and processed through the pipeline using graphical and command-line interfaces. Instructions for the cloud analysis pipeline are available at <https://github.com/dnanexus/viral-ngs/wiki>

1.2 Installation

1.2.1 System dependencies

This is known to install cleanly on most modern Linux systems with Python, Java, and some basic development libraries. On Ubuntu 14.04 LTS, the following APT packages should be installed on top of the vanilla setup:

```
python3 python3-pip python3-nose
python-software-properties
```

Java ≥ 1.7 is required by GATK and Picard.

1.2.2 Python dependencies

The **command line tools** require Python ≥ 2.7 or ≥ 3.4 . Required packages (pysam and Biopython) are listed in requirements.txt and can be installed the usual pip way:

```
pip install -r requirements.txt
```

Additionally, in order to use the **pipeline infrastructure**, Python 3.4 is required (Python 2 is not supported) and you must install snakemake as well:

```
pip install -r requirements-pipes.txt
```

However, most of the real functionality is encapsulated in the command line tools, which can be used without any of the pipeline infrastructure.

You should either `sudo pip install` or use a virtualenv (recommended).

1.2.3 Tool dependencies

A lot of effort has gone into writing auto download/compile wrappers for most of the bioinformatic tools we rely on here. They will auto-download and install the first time they are needed by any command. If you want to pre-install all of the external tools, simply type this:

```
python -m unittest test.test_tools.TestToolsInstallation -v
```

However, there are two tools in particular that cannot be auto-installed due to licensing restrictions. You will need to download and install these tools on your own (paying for it if your use case requires it) and set environment variables pointing to their installed location.

- GATK - <http://www.broadinstitute.org/gatk/>
- Novoalign - <http://www.novocraft.com/products/novoalign/>

The environment variables you will need to set are `GATK_PATH` and `NOVOALIGN_PATH`. These should be set to the full directory path that contains these tools (the jar file for GATK and the executable binaries for Novoalign).

In order to run GATK, you will need to have an appropriate version of the Java JDK installed. As of this writing, Java 1.7 is required for GATK 3.3.0.

Alternatively, if you are using the Snakemake pipelines, you can create a dictionary called “env_vars” in the config.json file for Snakemake, and the pipelines will automatically set all environment variables prior to running any scripts.

The version of MOSAIK we use seems to fail compile on GCC-4.9 but compiles fine on GCC-4.4. We have not tried intermediate versions of GCC, nor the latest versions of MOSAIK.

1.3 Command line tools

1.3.1 `taxon_filter.py` - tools for taxonomic removal or filtration of reads

This script contains a number of utilities for filtering NGS reads based on membership or non-membership in a species / genus / taxonomic grouping.

```
usage: taxon_filter.py subcommand
```

Sub-commands:

deplete_human Undocumented

Run the entire depletion pipeline: `bmtagger`, `mvicuna`, `blastn`. Optionally, use `lastal` to select a specific taxon of interest.

```
usage: taxon_filter.py deplete_human [-h] [--taxfiltBam TAXFILTBAM]
                                     --bmtaggerDbs BMTAGGERDBS
                                     [BMTAGGERDBS ...] --blastDbs BLASTDBS
                                     [BLASTDBS ...] [--lastDb LASTDB]
                                     [--JVMmemory JVMMEMORY]
                                     [--loglevel {DEBUG,INFO,WARNING,ERROR,CRITICAL}]
                                     [--version] [--tmpDir TMPDIR]
                                     [--tmpDirKeep]
                                     inBam revertBam bmtaggerBam rmdupBam
                                     blastnBam
```

Positional arguments:

inBam	Input BAM file.
revertBam	Output BAM: read markup reverted with Picard.
bmtaggerBam	Output BAM: depleted of human reads with BMTagger.
rmdupBam	Output BAM: <code>bmtaggerBam</code> run through M-Vicuna duplicate removal.

blastnBam Output BAM: rmdupBam run through another depletion of human reads with BLASTN.

Options:

--taxfiltBam Output BAM: blastnBam run through taxonomic selection via LASTAL.

--bmtaggerDbs Reference databases (one or more) to deplete from input. For each db, requires prior creation of db.bitmask by bmtool, and db.srprism.idx, db.srprism.map, etc. by srprism mkindex.

--blastDbs One or more reference databases for blast to deplete from input.

--lastDb One reference database for last (required if --taxfiltBam is specified).

--JVMmemory=4g JVM virtual memory size for Picard FilterSamReads (default: %(default)s)

--loglevel=DEBUG Verboseness of output. [default: %(default)s]
Possible choices: DEBUG, INFO, WARNING, ERROR, CRITICAL, EXCEPTION

--version, -V show program's version number and exit

--tmpDir=/tmp Base directory for temp files. [default: %(default)s]

--tmpDirKeep=False Keep the tmpDir if an exception occurs while running. Default is to delete all temp files at the end, even if there's a failure.

trim_trimmomatic Undocumented

Trim read sequences with Trimmomatic.

```
usage: taxon_filter.py trim_trimmomatic [-h]
                                     [--loglevel {DEBUG,INFO,WARNING,ERROR,CRITICAL}]
                                     [--version] [--tmpDir TMPDIR]
                                     [--tmpDirKeep]
                                     inFastq1 inFastq2 pairedOutFastq1
                                     pairedOutFastq2 clipFasta
```

Positional arguments:

inFastq1 Input reads 1

inFastq2 Input reads 2

pairedOutFastq1 Paired output 1

pairedOutFastq2 Paired output 2

clipFasta Fasta file with adapters, PCR sequences, etc. to clip off

Options:

--loglevel=DEBUG Verboseness of output. [default: %(default)s]
Possible choices: DEBUG, INFO, WARNING, ERROR, CRITICAL, EXCEPTION

--version, -V show program's version number and exit

--tmpDir=/tmp Base directory for temp files. [default: %(default)s]

--tmpDirKeep=False Keep the tmpDir if an exception occurs while running. Default is to delete all temp files at the end, even if there's a failure.

filter_lastal_bam Undocumented

Restrict input reads to those that align to the given reference database using LASTAL.

```
usage: taxon_filter.py filter_lastal_bam [-h] [--JVMmemory JVMMEMORY]
                                         [--loglevel {DEBUG,INFO,WARNING,ERROR,CRITICAL}]
                                         [--version] [--tmpDir TMPDIR]
                                         [--tmpDirKeep]
                                         inBam db outBam
```

Positional arguments:

inBam	Input reads
db	Database of taxa we keep
outBam	Output reads, filtered to refDb

Options:

--JVMmemory=4g	JVM virtual memory size (default: %(default)s)
--loglevel=DEBUG	Verboseness of output. [default: %(default)s] Possible choices: DEBUG, INFO, WARNING, ERROR, CRITICAL, EXCEPTION
--version, -V	show program's version number and exit
--tmpDir=/tmp	Base directory for temp files. [default: %(default)s]
--tmpDirKeep=False	Keep the tmpDir if an exception occurs while running. Default is to delete all temp files at the end, even if there's a failure.

filter_lastal Undocumented

Restrict input reads to those that align to the given reference database using LASTAL. Also, remove duplicates with prinseq.

```
usage: taxon_filter.py filter_lastal [-h]
                                     [--loglevel {DEBUG,INFO,WARNING,ERROR,CRITICAL}]
                                     [--version] [--tmpDir TMPDIR]
                                     [--tmpDirKeep]
                                     inFastq refDb outFastq
```

Positional arguments:

inFastq	Input fastq file
refDb	Reference database to retain from input
outFastq	Output fastq file

Options:

--loglevel=DEBUG	Verboseness of output. [default: %(default)s] Possible choices: DEBUG, INFO, WARNING, ERROR, CRITICAL, EXCEPTION
--version, -V	show program's version number and exit
--tmpDir=/tmp	Base directory for temp files. [default: %(default)s]

--tmpDirKeep=False Keep the tmpDir if an exception occurs while running. Default is to delete all temp files at the end, even if there's a failure.

partition_bmtagger Undocumented

Use bmtagger to partition input reads into ones that match at least one of several databases and ones that don't match any of the databases.

```
usage: taxon_filter.py partition_bmtagger [-h] [--outMatch OUTMATCH OUTMATCH]
                                          [--outNoMatch OUTNOMATCH OUTNOMATCH]
                                          [--loglevel {DEBUG,INFO,WARNING,ERROR,CRITICAL}]
                                          [--version] [--tmpDir TMPDIR]
                                          [--tmpDirKeep]
                                          inFastq1 inFastq2 refDbs
                                          [refDbs ...]
```

Positional arguments:

inFastq1	Input fastq file; 1st end of paired-end reads.
inFastq2	Input fastq file; 2nd end of paired-end reads. Must have same names as inFastq1
refDbs	Reference databases (one or more) to deplete from input. For each db, requires prior creation of db.bitmask by bmttool, and db.srprism.idx, db.srprism.map, etc. by srprism mkindex.

Options:

--outMatch	Filenames for fastq output of matching reads.
--outNoMatch	Filenames for fastq output of unmatched reads.
--loglevel=DEBUG	Verboseness of output. [default: %(default)s] Possible choices: DEBUG, INFO, WARNING, ERROR, CRITICAL, EXCEPTION
--version, -V	show program's version number and exit
--tmpDir=/tmp	Base directory for temp files. [default: %(default)s]
--tmpDirKeep=False	Keep the tmpDir if an exception occurs while running. Default is to delete all temp files at the end, even if there's a failure.

deplete_bam_bmtagger Undocumented

Use bmtagger to deplete input reads against several databases.

```
usage: taxon_filter.py deplete_bam_bmtagger [-h] [--JVMmemory JVMMEMORY]
                                             [--loglevel {DEBUG,INFO,WARNING,ERROR,CRITICAL}]
                                             [--version] [--tmpDir TMPDIR]
                                             [--tmpDirKeep]
                                             inBam refDbs [refDbs ...] outBam
```

Positional arguments:

inBam	Input BAM file.
refDbs	Reference databases (one or more) to deplete from input. For each db, requires prior creation of db.bitmask by bmttool, and db.srprism.idx, db.srprism.map, etc. by srprism mkindex.
outBam	Output BAM file.

Options:

- JVMmemory=4g** JVM virtual memory size (default: %(default)s)
- loglevel=DEBUG** Verboseness of output. [default: %(default)s]
Possible choices: DEBUG, INFO, WARNING, ERROR, CRITICAL, EXCEPTION
- version, -V** show program's version number and exit
- tmpDir=/tmp** Base directory for temp files. [default: %(default)s]
- tmpDirKeep=False** Keep the tmpDir if an exception occurs while running. Default is to delete all temp files at the end, even if there's a failure.

deplete_blastn Undocumented

Use blastn to remove reads that match at least one of the databases.

```
usage: taxon_filter.py deplete_blastn [-h]
                                     [--loglevel {DEBUG, INFO, WARNING, ERROR, CRITICAL, EXCEPTION}]
                                     [--version] [--tmpDir TMPDIR]
                                     [--tmpDirKeep]
                                     inFastq outFastq refDbs [refDbs ...]
```

Positional arguments:

- inFastq** Input fastq file.
- outFastq** Output fastq file with matching reads removed.
- refDbs** One or more reference databases for blast.

Options:

- loglevel=DEBUG** Verboseness of output. [default: %(default)s]
Possible choices: DEBUG, INFO, WARNING, ERROR, CRITICAL, EXCEPTION
- version, -V** show program's version number and exit
- tmpDir=/tmp** Base directory for temp files. [default: %(default)s]
- tmpDirKeep=False** Keep the tmpDir if an exception occurs while running. Default is to delete all temp files at the end, even if there's a failure.

deplete_blastn_paired Undocumented

Use blastn to remove reads that match at least one of the databases.

```
usage: taxon_filter.py deplete_blastn_paired [-h]
                                              [--loglevel {DEBUG, INFO, WARNING, ERROR, CRITICAL, EXCEPTION}]
                                              [--version] [--tmpDir TMPDIR]
                                              [--tmpDirKeep]
                                              infq1 infq2 outfq1 outfq2 refDbs
                                              [refDbs ...]
```

Positional arguments:

- infq1** Input fastq file.
- infq2** Input fastq file.
- outfq1** Output fastq file with matching reads removed.

outfq2	Output fastq file with matching reads removed.
refDbs	One or more reference databases for blast.

Options:

--loglevel=DEBUG	Verboseness of output. [default: %(default)s] Possible choices: DEBUG, INFO, WARNING, ERROR, CRITICAL, EXCEPTION
--version, -V	show program's version number and exit
--tmpDir=/tmp	Base directory for temp files. [default: %(default)s]
--tmpDirKeep=False	Keep the tmpDir if an exception occurs while running. Default is to delete all temp files at the end, even if there's a failure.

deplete_blastn_bam Undocumented

Use blastn to remove reads that match at least one of the specified databases.

```
usage: taxon_filter.py deplete_blastn_bam [-h] [--chunkSize CHUNKSIZE]
                                         [--JVMmemory JVMMEMORY]
                                         [--loglevel {DEBUG,INFO,WARNING,ERROR,CRITICAL,EXCEPTION}]
                                         [--version] [--tmpDir TMPDIR]
                                         [--tmpDirKeep]
                                         inBam refDbs [refDbs ...] outBam
```

Positional arguments:

inBam	Input BAM file.
refDbs	One or more reference databases for blast.
outBam	Output BAM file with matching reads removed.

Options:

--chunkSize=1000000	FASTA chunk size (default: %(default)s)
--JVMmemory=4g	JVM virtual memory size (default: %(default)s)
--loglevel=DEBUG	Verboseness of output. [default: %(default)s] Possible choices: DEBUG, INFO, WARNING, ERROR, CRITICAL, EXCEPTION
--version, -V	show program's version number and exit
--tmpDir=/tmp	Base directory for temp files. [default: %(default)s]
--tmpDirKeep=False	Keep the tmpDir if an exception occurs while running. Default is to delete all temp files at the end, even if there's a failure.

lastal_build_db Undocumented

```
usage: taxon_filter.py lastal_build_db [-h]
                                         [--outputFilePrefix OUTPUTFILEPREFIX]
                                         [--loglevel {DEBUG,INFO,WARNING,ERROR,CRITICAL,EXCEPTION}]
                                         [--version] [--tmpDir TMPDIR]
                                         [--tmpDirKeep]
                                         inputFasta outputDirectory
```

Positional arguments:

inputFasta	Location of the input FASTA file
-------------------	----------------------------------

outputDirectory	Location for the output files (default is cwd: %(default)s)
Options:	
--outputFilePrefix	Prefix for the output file name (default: inputFasta name, sans ".fasta" extension)
--loglevel=DEBUG	Verboseness of output. [default: %(default)s] Possible choices: DEBUG, INFO, WARNING, ERROR, CRITICAL, EXCEPTION
--version, -V	show program's version number and exit
--tmpDir=/tmp	Base directory for temp files. [default: %(default)s]
--tmpDirKeep=False	Keep the tmpDir if an exception occurs while running. Default is to delete all temp files at the end, even if there's a failure.

1.3.2 assembly.py - *de novo* assembly

This script contains a number of utilities for viral sequence assembly from NGS reads. Primarily used for Lassa and Ebola virus analysis in the Sabeti Lab / Broad Institute Viral Genomics.

usage: assembly.py subcommand

Sub-commands:

trim_rmdup_subsamp Undocumented

Take reads through Trimmomatic, Prinseq, and subsampling. This should probably move over to read_utils or taxon_filter.

```
usage: assembly.py trim_rmdup_subsamp [-h] [--n_reads N_READS]
                                     [--loglevel {DEBUG, INFO, WARNING, ERROR, CRITICAL, EXCEPTION}]
                                     [--version] [--tmpDir TMPDIR]
                                     [--tmpDirKeep]
                                     inBam clipDb outBam
```

Positional arguments:

inBam	Input reads, unaligned BAM format.
clipDb	Trimmomatic clip DB.
outBam	Output reads, unaligned BAM format (currently, read groups and other header information are destroyed in this process).

Options:

--n_reads=100000	Subsample reads to no more than this many pairs. (default %(default)s)
--loglevel=DEBUG	Verboseness of output. [default: %(default)s] Possible choices: DEBUG, INFO, WARNING, ERROR, CRITICAL, EXCEPTION
--version, -V	show program's version number and exit
--tmpDir=/tmp	Base directory for temp files. [default: %(default)s]
--tmpDirKeep=False	Keep the tmpDir if an exception occurs while running. Default is to delete all temp files at the end, even if there's a failure.

assemble_trinity Undocumented

This step runs the Trinity assembler. First trim reads with trimmomatic, rmdup with prinseq, and random subsample to no more than 100k reads.

```
usage: assembly.py assemble_trinity [-h] [--n_reads N_READS]
                                     [--outReads OUTREADS]
                                     [--JVMmemory JVMMEMORY]
                                     [--threads THREADS]
                                     [--loglevel {DEBUG,INFO,WARNING,ERROR,CRITICAL,
                                     [--version] [--tmpDir TMPDIR]
                                     [--tmpDirKeep]
                                     inBam clipDb outFasta
```

Positional arguments:

inBam	Input unaligned reads, BAM format.
clipDb	Trimmomatic clip DB.
outFasta	Output assembly.

Options:

--n_reads=100000	Subsample reads to no more than this many pairs. (default %(default)s)
--outReads	Save the trimmomatic/prinseq/subsamp reads to a BAM file
--JVMmemory=4g	JVM virtual memory size (default: %(default)s)
--threads=1	Number of threads (default: %(default)s)
--loglevel=DEBUG	Verboseness of output. [default: %(default)s] Possible choices: DEBUG, INFO, WARNING, ERROR, CRITICAL, EXCEPTION
--version, -V	show program's version number and exit
--tmpDir=/tmp	Base directory for temp files. [default: %(default)s]
--tmpDirKeep=False	Keep the tmpDir if an exception occurs while running. Default is to delete all temp files at the end, even if there's a failure.

order_and_orient Undocumented

This step cleans up the Trinity assembly with a known reference genome. Uses VFAT (switch to Bellini later): Take the Trinity contigs, align them to the known reference genome, switch it to the same strand as the reference, and produce chromosome-level assemblies (with runs of N's in between the Trinity contigs).

```
usage: assembly.py order_and_orient [-h] [--inReads INREADS]
                                     [--loglevel {DEBUG,INFO,WARNING,ERROR,CRITICAL,
                                     [--version] [--tmpDir TMPDIR]
                                     [--tmpDirKeep]
                                     inFasta inReference outFasta
```

Positional arguments:

inFasta	Input de novo assembly/contigs, FASTA format.
inReference	Reference genome for ordering, orienting, and merging contigs, FASTA format.

outFasta Output assembly, FASTA format, with the same number of chromosomes as inReference, and in the same order.

Options:

--inReads Input reads in unaligned BAM format. These can be used to improve the merge process.

--loglevel=DEBUG Verboseness of output. [default: %(default)s]
Possible choices: DEBUG, INFO, WARNING, ERROR, CRITICAL, EXCEPTION

--version, -V show program's version number and exit

--tmpDir=/tmp Base directory for temp files. [default: %(default)s]

--tmpDirKeep=False Keep the tmpDir if an exception occurs while running. Default is to delete all temp files at the end, even if there's a failure.

impute_from_reference Undocumented

This takes a de novo assembly, aligns against a reference genome, and imputes all missing positions (plus some of the chromosome ends) with the reference genome. This provides an assembly with the proper structure (but potentially wrong sequences in areas) from which we can perform further read-based refinement. Two steps: filter_short_seqs: We then toss out all assemblies that come out to < 15kb or < 95% unambiguous and fail otherwise. modify_contig: Finally, we trim off anything at the end that exceeds the length of the known reference assembly. We also replace all Ns and everything within 55bp of the chromosome ends with the reference sequence. This is clearly incorrect consensus sequence, but it allows downstream steps to map reads in parts of the genome that would otherwise be Ns, and we will correct all of the inferred positions with two steps of read-based refinement (below), and revert positions back to Ns where read support is lacking. FASTA indexing: output assembly is indexed for Picard, Samtools, Novoalign.

```
usage: assembly.py impute_from_reference [-h] [--newName NEWNAME]
                                         [--minLengthFraction MINLENGTHFRACTION]
                                         [--minUnambig MINUNAMBIG]
                                         [--replaceLength REPLACELENGTH]
                                         [--loglevel {DEBUG,INFO,WARNING,ERROR,CRITICAL}]
                                         [--version] [--tmpDir TMPDIR]
                                         [--tmpDirKeep]
                                         inFasta inReference outFasta
```

Positional arguments:

inFasta Input assembly/contigs, FASTA format, already ordered, oriented and merged with inReference.

inReference Reference genome to impute with, FASTA format.

outFasta Output assembly, FASTA format.

Options:

--newName rename output chromosome (default: do not rename)

--minLengthFraction=0.9 minimum length for contig, as fraction of reference (default: %(default)s)

--minUnambig=0.0 minimum percentage unambiguous bases for contig (default: %(default)s)

--replaceLength=0	length of ends to be replaced with reference (default: <code>%(default)s</code>)
--loglevel=DEBUG	Verboseness of output. [default: <code>%(default)s</code>] Possible choices: DEBUG, INFO, WARNING, ERROR, CRITICAL, EXCEPTION
--version, -V	show program's version number and exit
--tmpDir=/tmp	Base directory for temp files. [default: <code>%(default)s</code>]
--tmpDirKeep=False	Keep the tmpDir if an exception occurs while running. Default is to delete all temp files at the end, even if there's a failure.

refine_assembly Undocumented

This a refinement step where we take a crude assembly, align all reads back to it, and modify the assembly to the majority allele at each position based on read pileups. This step considers both SNPs as well as indels called by GATK and will correct the consensus based on GATK calls. Reads are aligned with Novoalign, then PCR duplicates are removed with Picard (in order to debias the allele counts in the pileups), and realigned with GATK's IndelRealigner (in order to call indels). Output FASTA file is indexed for Picard, Samtools, and Novoalign.

```
usage: assembly.py refine_assembly [-h] [--outBam OUTBAM] [--outVcf OUTVCF]
                                   [--min_coverage MIN_COVERAGE]
                                   [--novo_params NOVO_PARAMS]
                                   [--chr_names [CHR_NAMES [CHR_NAMES ...]]]
                                   [--keep_all_reads] [--JVMmemory JVMMEMORY]
                                   [--threads THREADS]
                                   [--loglevel {DEBUG,INFO,WARNING,ERROR,CRITICAL,EXCEPTION}]
                                   [--version] [--tmpDir TMPDIR]
                                   [--tmpDirKeep]
                                   inFasta inBam outFasta
```

Positional arguments:

inFasta	Input assembly, FASTA format, pre-indexed for Picard, Samtools, and Novoalign.
inBam	Input reads, unaligned BAM format.
outFasta	Output refined assembly, FASTA format, indexed for Picard, Samtools, and Novoalign.

Options:

--outBam	Reads aligned to inFasta. Unaligned and duplicate reads have been removed. GATK indel realigned.
--outVcf	GATK genotype calls for genome in inFasta coordinate space.
--min_coverage=3	Minimum read coverage required to call a position unambiguous.
--novo_params=-r Random -l 40 -g 40 -x 20 -t 100	Alignment parameters for Novoalign.
--chr_names=[]	Rename all output chromosomes (default: retain original chromosome names)
--keep_all_reads=False	Retain all reads in BAM file? Default is to remove unaligned and duplicate reads.

--JVMmemory=2g JVM virtual memory size (default: %(default)s)
--threads=1 Number of threads (default: %(default)s)
--loglevel=DEBUG Verboseness of output. [default: %(default)s]
Possible choices: DEBUG, INFO, WARNING, ERROR, CRITICAL, EXCEPTION
--version, -V show program's version number and exit
--tmpDir=/tmp Base directory for temp files. [default: %(default)s]
--tmpDirKeep=False Keep the tmpDir if an exception occurs while running. Default is to delete all temp files at the end, even if there's a failure.

filter_short_seqs Undocumented

Check sequences in inFile, retaining only those that are at least minLength

```
usage: assembly.py filter_short_seqs [-h] [-f FORMAT] [-of OUTPUT_FORMAT]
                                     [--loglevel {DEBUG, INFO, WARNING, ERROR, CRITICAL, EXCEPTION}]
                                     [--version]
                                     inFile minLength minUnambig outFile
```

Positional arguments:

inFile input sequence file
minLength minimum length for contig
minUnambig minimum percentage unambiguous bases for contig
outFile output file

Options:

-f=fasta, --format=fasta Format for input sequence (default: %(default)s)
-of=fasta, --output-format=fasta Format for output sequence (default: %(default)s)
--loglevel=DEBUG Verboseness of output. [default: %(default)s]
Possible choices: DEBUG, INFO, WARNING, ERROR, CRITICAL, EXCEPTION
--version, -V show program's version number and exit

modify_contig Undocumented

Modifies an input contig. Depending on the options selected, can replace N calls with reference calls, replace ambiguous calls with reference calls, trim to the length of the reference, replace contig ends with reference calls, and trim leading and trailing Ns. Author: rsealfon.

```
usage: assembly.py modify_contig [-h] [-n NAME] [-cn] [-t] [-r5] [-r3]
                                [-l REPLACE_LENGTH] [-f FORMAT] [-r] [-rn]
                                [-ca] [--tmpDir TMPDIR] [--tmpDirKeep]
                                [--loglevel {DEBUG, INFO, WARNING, ERROR, CRITICAL, EXCEPTION}]
                                [--version]
                                input output ref
```

Positional arguments:

input input alignment of reference and contig (should contain exactly 2 sequences)
output Destination file for modified contigs

ref reference sequence name (exact match required)

Options:

-n, --name fasta header output name (default: existing header)

-cn=False, --call-reference-ns=False should the reference sequence be called if there is an N in the contig and a more specific base in the reference (default: %(default)s)

-t=False, --trim-ends=False should ends of contig.fasta be trimmed to length of reference (default: %(default)s)

-r5=False, --replace-5ends=False should the 5'-end of contig.fasta be replaced by reference (default: %(default)s)

-r3=False, --replace-3ends=False should the 3'-end of contig.fasta be replaced by reference (default: %(default)s)

-l=10, --replace-length=10 length of ends to be replaced (if replace-ends is yes) (default: %(default)s)

-f=fasta, --format=fasta Format for input alignment (default: %(default)s)

-r=False, --replace-end-gaps=False Replace gaps at the beginning and end of the sequence with reference sequence (default: %(default)s)

-rn=False, --remove-end-ns=False Remove leading and trailing N's in the contig (default: %(default)s)

-ca=False, --call-reference-ambiguous=False should the reference sequence be called if the contig seq is ambiguous and the reference sequence is more informative & consistent with the ambiguous base (ie Y->C) (default: %(default)s)

--tmpDir=/tmp Base directory for temp files. [default: %(default)s]

--tmpDirKeep=False Keep the tmpDir if an exception occurs while running. Default is to delete all temp files at the end, even if there's a failure.

--loglevel=DEBUG Verboseness of output. [default: %(default)s]
Possible choices: DEBUG, INFO, WARNING, ERROR, CRITICAL, EXCEPTION

--version, -V show program's version number and exit

vcf_to_fasta Undocumented

Take input genotypes (VCF) and construct a consensus sequence (fasta) by using majority-read-count alleles in the VCF. Genotypes in the VCF will be ignored—we will use the allele with majority read support (or an ambiguity base if there is no clear majority). Uncalled positions will be emitted as N's. Author: dpark.

```
usage: assembly.py vcf_to_fasta [-h] [--trim_ends] [--min_coverage MIN_DP]
                                [--major_cutoff MAJOR_CUTOFF]
                                [--min_dp_ratio MIN_DP_RATIO]
                                [--name [NAME [NAME ...]]]
                                [--loglevel {DEBUG, INFO, WARNING, ERROR, CRITICAL, EXCE
                                [--version]
                                inVcf outFasta
```

Positional arguments:

inVcf Input VCF file
outFasta Output FASTA file

Options:

--trim_ends=False If specified, we will strip off continuous runs of N's from the beginning and end of the sequences before writing to output. Interior N's will not be changed.

--min_coverage=3 Specify minimum read coverage (with full agreement) to make a call. [default: %(default)s]

--major_cutoff=0.5 If the major allele is present at a frequency higher than this cutoff, we will call an unambiguous base at that position. If it is equal to or below this cutoff, we will call an ambiguous base representing all possible alleles at that position. [default: %(default)s]

--min_dp_ratio=0.0 The input VCF file often reports two read depth values (DP)—one for the position as a whole, and one for the sample in question. We can optionally reject calls in which the sample read count is below a specified fraction of the total read count. This filter will not apply to any sites unless both DP values are reported. [default: %(default)s]

--name=[] output sequence names (default: reference names in VCF file)

--loglevel=DEBUG Verboseness of output. [default: %(default)s]
Possible choices: DEBUG, INFO, WARNING, ERROR, CRITICAL, EXCEPTION

--version, -V show program's version number and exit

trim_fasta Undocumented

Take input sequences (fasta) and trim any continuous sections of N's from the ends of them. Write trimmed sequences to an output fasta file.

```
usage: assembly.py trim_fasta [-h]
                             [--loglevel {DEBUG,INFO,WARNING,ERROR,CRITICAL,EXCEPTION}]
                             [--version]
                             inFasta outFasta
```

Positional arguments:

inFasta Input fasta file
outFasta Output (trimmed) fasta file

Options:

--loglevel=DEBUG Verboseness of output. [default: %(default)s]
Possible choices: DEBUG, INFO, WARNING, ERROR, CRITICAL, EXCEPTION

--version, -V show program's version number and exit

deambig_fasta Undocumented

Take input sequences (fasta) and replace any ambiguity bases with a random unambiguous base from among the possibilities described by the ambiguity code. Write output to fasta file.

```
usage: assembly.py deambig_fasta [-h]
                                [--loglevel {DEBUG,INFO,WARNING,ERROR,CRITICAL,EXCEPTION}]
                                [--version]
                                inFasta outFasta
```

Positional arguments:

inFasta	Input fasta file
outFasta	Output fasta file

Options:

--loglevel=DEBUG	Verboseness of output. [default: %(default)s] Possible choices: DEBUG, INFO, WARNING, ERROR, CRITICAL, EXCEPTION
--version, -V	show program's version number and exit

dpdiff Undocumented

Take input VCF files (all with only one sample each) and report on the discrepancies between the two DP fields (one in INFO and one in the sample's genotype column).

```
usage: assembly.py dpdiff [-h]
                          [--loglevel {DEBUG,INFO,WARNING,ERROR,CRITICAL,EXCEPTION}]
                          [--version]
                          inVcfs [inVcfs ...] outFile
```

Positional arguments:

inVcfs	Input VCF file
outFile	Output flat file

Options:

--loglevel=DEBUG	Verboseness of output. [default: %(default)s] Possible choices: DEBUG, INFO, WARNING, ERROR, CRITICAL, EXCEPTION
--version, -V	show program's version number and exit

1.3.3 interhost.py - species and population-level genetic variation

This script contains a number of utilities for SNP calling, multi-alignment, phylogenetics, etc.

```
usage: interhost.py subcommand
```

Sub-commands:**snpEff** Undocumented

Annotate variants in VCF file with translation consequences using snpEff.

```
usage: interhost.py snpEff [-h] [--tmpDir TMPDIR] [--tmpDirKeep]
                           [--loglevel {DEBUG,INFO,WARNING,ERROR,CRITICAL,EXCEPTION}]
                           [--version]
                           inVcf genomes [genomes ...] outVcf emailAddress
```

Positional arguments:

inVcf	Input VCF file
--------------	----------------

genomes	genome name
outVcf	Output VCF file
emailAddress	Your email address. To access the Genbank CoreNucleotide database, NCBI requires you to specify your email address with each request. In case of excessive usage of the E-utilities, NCBI will attempt to contact a user at the email address provided before blocking access.

Options:

--tmpDir=/tmp	Base directory for temp files. [default: %(default)s]
--tmpDirKeep=False	Keep the tmpDir if an exception occurs while running. Default is to delete all temp files at the end, even if there's a failure.
--loglevel=DEBUG	Verboseness of output. [default: %(default)s] Possible choices: DEBUG, INFO, WARNING, ERROR, CRITICAL, EXCEPTION
--version, -V	show program's version number and exit

align_mafft Undocumented

Run the mafft alignment on the input FASTA file.

```
usage: interhost.py align_mafft [-h] [--localpair | --globalpair]
                                [--preservecase] [--reorder]
                                [--gapOpeningPenalty GAOPENINGPENALTY]
                                [--ep EP] [--verbose] [--outputAsClustal]
                                [--maxiters MAXITERS] [--threads THREADS]
                                [--loglevel {DEBUG,INFO,WARNING,ERROR,CRITICAL,EXCEPTION}]
                                [--version] [--tmpDir TMPDIR] [--tmpDirKeep]
                                inFastas [inFastas ...] outFile
```

Positional arguments:

inFastas	Input FASTA files.
outFile	Output file containing alignment result (default format: FASTA)

Options:

--localpair	All pairwise alignments are computed with the Smith-Waterman algorithm.
--globalpair	All pairwise alignments are computed with the Needleman-Wunsch algorithm.
--preservecase	Preserve base or aa case, as well as symbols.
--reorder	Output is ordered aligned rather than in the order of the input (default: %(default)s).
--gapOpeningPenalty=1.53	Gap opening penalty (default: %(default)s).
--ep	Offset (works like gap extension penalty).
--verbose=False	Full output (default: %(default)s).
--outputAsClustal	Write output file in Clustal format rather than FASTA

--maxiters=0	Maximum number of refinement iterations (default: <code>%(default)s</code>). Note: if “--localpair” or “--globalpair” is specified this defaults to 1000.
--threads=-1	Number of processing threads (default: <code>%(default)s</code> , where -1 indicates use of all available cores).
--loglevel=DEBUG	Verboseness of output. [default: <code>%(default)s</code>] Possible choices: DEBUG, INFO, WARNING, ERROR, CRITICAL, EXCEPTION
--version, -V	show program’s version number and exit
--tmpDir=/tmp	Base directory for temp files. [default: <code>%(default)s</code>]
--tmpDirKeep=False	Keep the tmpDir if an exception occurs while running. Default is to delete all temp files at the end, even if there’s a failure.

multichr_mafft Undocumented

Run the mafft alignment on a series of chromosomes provided in sample-partitioned FASTA files. Output as FASTA. (i.e. file1.fasta would contain chr1, chr2, chr3; file2.fasta would also contain chr1, chr2, chr3)

```
usage: interhost.py multichr_mafft [-h] [--localpair | --globalpair]
                                   [--preservecase] [--reorder]
                                   [--gapOpeningPenalty GAOPENINGPENALTY]
                                   [--ep EP] [--verbose] [--outputAsClustal]
                                   [--maxiters MAXITERS] [--threads THREADS]
                                   [--outFilePrefix OUTFILEPREFIX]
                                   [--sampleRelationFile SAMPLERELATIONFILE]
                                   [--sampleNameListFile SAMPLENAMELISTFILE]
                                   [--loglevel {DEBUG,INFO,WARNING,ERROR,CRITICAL,EXCEPTION}]
                                   [--version] [--tmpDir TMPDIR]
                                   [--tmpDirKeep]
                                   inFastas [inFastas ...] outDirectory
```

Positional arguments:

inFastas	Input FASTA files.
outDirectory	Location for the output files (default is cwd: <code>%(default)s</code>)

Options:

--localpair	All pairwise alignments are computed with the Smith-Waterman algorithm.
--globalpair	All pairwise alignments are computed with the Needleman-Wunsch algorithm.
--preservecase	Preserve base or aa case, as well as symbols.
--reorder	Output is ordered aligned rather than in the order of the input (default: <code>%(default)s</code>).
--gapOpeningPenalty=1.53	Gap opening penalty (default: <code>%(default)s</code>).
--ep	Offset (works like gap extension penalty).
--verbose=False	Full output (default: <code>%(default)s</code>).
--outputAsClustal	Write output file in Clustal format rather than FASTA

--maxiters=0	Maximum number of refinement iterations (default: <code>%(default)s</code>). Note: if “ <code>--localpair</code> ” or “ <code>--globalpair</code> ” is specified this defaults to 1000.
--threads=-1	Number of processing threads (default: <code>%(default)s</code> , where -1 indicates use of all available cores).
--outFilePrefix=aligned	Prefix for the output file name (default: <code>%(default)s</code>)
--sampleRelationFile	If the parameter <code>sampleRelationFile</code> is specified (as a file path), a JSON file will be written mapping sample name to sequence position in the output.
--sampleNameListFile	If the parameter <code>sampleRelationFile</code> is specified (as a file path), a file will be written mapping sample names in the order of their sequence positions in the output.
--loglevel=DEBUG	Verboseness of output. [default: <code>%(default)s</code>] Possible choices: DEBUG, INFO, WARNING, ERROR, CRITICAL, EXCEPTION
--version, -V	show program’s version number and exit
--tmpDir=/tmp	Base directory for temp files. [default: <code>%(default)s</code>]
--tmpDirKeep=False	Keep the <code>tmpDir</code> if an exception occurs while running. Default is to delete all temp files at the end, even if there’s a failure.

1.3.4 intrahost.py - within-host genetic variation (iSNVs)

This script contains a number of utilities for intrahost variant calling and annotation for viral genomes.

usage: `intrahost.py subcommand`

Sub-commands:

vphaser_one_sample Undocumented

Input: a single BAM file, representing reads from one sample, mapped to its own consensus assembly. It may contain multiple read groups and libraries. Output: a tab-separated file with no header containing filtered V Phaser-2 output variants with additional column for sequence/chrom name, and library counts and p-values appended to the counts for each allele.

```
usage: intrahost.py vphaser_one_sample [-h]
                                     [--vphaserNumThreads VPHASERNUMTHREADS]
                                     [--minReadsEach MINREADSEACH]
                                     [--maxBias MAXBIAS]
                                     [--removeDoublyMappedReads]
                                     [--loglevel {DEBUG,INFO,WARNING,ERROR,CRITICAL}]
                                     [--version]
                                     inBam inConsFasta outTab
```

Positional arguments:

inBam	Input Bam file.
inConsFasta	Consensus assembly fasta.
outTab	Tab-separated headerless output file.

Options:

--vphaserNumThreads Number of threads in call to V-Phaser 2.

--minReadsEach=5 Minimum number of reads on each strand (default: %(default)s).

--maxBias=10 Maximum allowable ratio of number of reads on the two strands (default: %(default)s). Ignored if minReadsEach = 0.

--removeDoublyMappedReads=False When calling V-Phaser, keep reads mapping to more than one contig.

--loglevel=DEBUG Verboseness of output. [default: %(default)s]
Possible choices: DEBUG, INFO, WARNING, ERROR, CRITICAL, EXCEPTION

--version, -V show program's version number and exit

vphaser Undocumented

Run V-Phaser 2 on the input file without any additional filtering. Combine the non-header lines of the CHROM.var.raw.txt files it produces, adding CHROM as the first field on each line.

```
usage: intrahost.py vphaser [-h] [--numThreads NUMTHREADS]
                             [--loglevel {DEBUG, INFO, WARNING, ERROR, CRITICAL, EXCEPTION}]
                             [--version]
                             inBam outTab
```

Positional arguments:

inBam Input Bam file.

outTab Tab-separated headerless output file.

Options:

--numThreads Number of threads in call to V-Phaser 2.

--loglevel=DEBUG Verboseness of output. [default: %(default)s]
Possible choices: DEBUG, INFO, WARNING, ERROR, CRITICAL, EXCEPTION

--version, -V show program's version number and exit

tabfile_rename Undocumented

Take input tab file and copy to an output file while changing the values in a specific column based on a mapping file. The first line will pass through untouched (it is assumed to be a header).

```
usage: intrahost.py tabfile_rename [-h] [--col_idx COL]
                                     [--loglevel {DEBUG, INFO, WARNING, ERROR, CRITICAL, EXCEPTION}]
                                     [--version]
                                     inFile mapFile outFile
```

Positional arguments:

inFile Input flat file

mapFile Map file. Two-column headerless file that maps input values to output values. This script will error if there are values in inFile that do not exist in mapFile.

outFile Output flat file

Options:

--col_idx=0	Which column number to replace (0-based index). [default: %(default)s]
--loglevel=DEBUG	Verboseness of output. [default: %(default)s] Possible choices: DEBUG, INFO, WARNING, ERROR, CRITICAL, EXCEPTION
--version, -V	show program's version number and exit

merge_to_vcf Undocumented

Combine and convert vPhaser2 parsed filtered output text files into VCF format. Assumption: consensus assemblies used in creating alignments do not extend beyond ends of reference. the number of alignment files equals the number of chromosomes / segments

```
usage: intrahost.py merge_to_vcf [-h] --samples SAMPLES [SAMPLES ...] --isnvs
                                ISNVS [ISNVS ...] --alignments ALIGNMENTS
                                [ALIGNMENTS ...] [--strip_chr_version]
                                [--naive_filter] [--parse_accession]
                                [--loglevel {DEBUG,INFO,WARNING,ERROR,CRITICAL,EXC
                                [--version]
                                refFasta outVcf
```

Positional arguments:

refFasta	The target reference genome. outVcf will use these chromosome names, coordinate spaces, and reference alleles
outVcf	Output VCF file containing all variants

Options:

--samples	A list of sample names
--isnvs	A list of file names from the output of vphaser_one_sample These must be in the SAME ORDER as samples.
--alignments	a list of fasta files containing multialignment of input assemblies, with one file per chromosome/segment. Each alignment file will contain a line for each sample, as well as the reference genome to which they were aligned.
--strip_chr_version=False	If set, strip any trailing version numbers from the chromosome names. If the chromosome name ends with a period followed by integers, this is interpreted as a version number to be removed. This is because Genbank accession numbers are often used by SnpEff databases downstream, but without the corresponding version number. Default is false (leave chromosome names untouched).
--naive_filter=False	If set, keep only the alleles that have at least two independent libraries of support and allele freq > 0.005. Default is false (do not filter at this stage).
--parse_accession=False	If set, parse only the accession for the chromosome name. Helpful if snpEff has to create its own database
--loglevel=DEBUG	Verboseness of output. [default: %(default)s] Possible choices: DEBUG, INFO, WARNING, ERROR, CRITICAL, EXCEPTION

--version, -V show program's version number and exit

Fws Undocumented

Compute the Fws statistic on iSNV data. See Manske, 2012 (Nature)

```
usage: intrahost.py Fws [-h]
                        [--loglevel {DEBUG,INFO,WARNING,ERROR,CRITICAL,EXCEPTION}]
                        [--version]
                        inVcf outVcf
```

Positional arguments:

inVcf Input VCF file
outVcf Output VCF file

Options:

--loglevel=DEBUG Verboseness of output. [default: %(default)s]
Possible choices: DEBUG, INFO, WARNING, ERROR, CRITICAL, EXCEPTION
--version, -V show program's version number and exit

iSNV_table Undocumented

Convert VCF iSNV data to tabular text

```
usage: intrahost.py iSNV_table [-h]
                               [--loglevel {DEBUG,INFO,WARNING,ERROR,CRITICAL,EXCEPTION}]
                               [--version]
                               inVcf outFile
```

Positional arguments:

inVcf Input VCF file
outFile Output text file

Options:

--loglevel=DEBUG Verboseness of output. [default: %(default)s]
Possible choices: DEBUG, INFO, WARNING, ERROR, CRITICAL, EXCEPTION
--version, -V show program's version number and exit

iSNP_per_patient Undocumented

Aggregate tabular iSNP data per patient x position (all time points averaged)

```
usage: intrahost.py iSNP_per_patient [-h]
                                       [--loglevel {DEBUG,INFO,WARNING,ERROR,CRITICAL,EXCEPTION}]
                                       [--version]
                                       inFile outFile
```

Positional arguments:

inFile Input text file
outFile Output text file

Options:

--loglevel=DEBUG Verboseness of output. [default: %(default)s]
Possible choices: DEBUG, INFO, WARNING, ERROR, CRITICAL, EXCEPTION

--version, -V show program's version number and exit

1.3.5 read_utils.py - utilities that manipulate bam and fastq files

Utilities for working with sequence reads, such as converting formats and fixing mate pairs.

usage: read_utils.py subcommand

Sub-commands:

purge_unmated Undocumented

Use mergeShuffledFastqSeqs to purge unmated reads, and put corresponding reads in the same order. Corresponding sequences must have sequence identifiers of the form SEQID/1 and SEQID/2.

usage: read_utils.py purge_unmated [-h] [--regex REGEX]
[--loglevel {DEBUG, INFO, WARNING, ERROR, CRITICAL, EXCEPTION}]
[--version] [--tmpDir TMPDIR]
[--tmpDirKeep]
inFastq1 inFastq2 outFastq1 outFastq2

Positional arguments:

inFastq1 Input fastq file; 1st end of paired-end reads.

inFastq2 Input fastq file; 2nd end of paired-end reads.

outFastq1 Output fastq file; 1st end of paired-end reads.

outFastq2 Output fastq file; 2nd end of paired-end reads.

Options:

--regex=^@(\S+)/[12]\$ Perl regular expression to parse paired read IDs (default: %(default)s)

--loglevel=DEBUG Verboseness of output. [default: %(default)s]
Possible choices: DEBUG, INFO, WARNING, ERROR, CRITICAL, EXCEPTION

--version, -V show program's version number and exit

--tmpDir=/tmp Base directory for temp files. [default: %(default)s]

--tmpDirKeep=False Keep the tmpDir if an exception occurs while running. Default is to delete all temp files at the end, even if there's a failure.

fastq_to_fasta Undocumented

Convert from fastq format to fasta format. Warning: output reads might be split onto multiple lines.

usage: read_utils.py fastq_to_fasta [-h]
[--loglevel {DEBUG, INFO, WARNING, ERROR, CRITICAL, EXCEPTION}]
[--version] [--tmpDir TMPDIR]
[--tmpDirKeep]
inFastq outFasta

Positional arguments:

inFastq Input fastq file.
outFasta Output fasta file.

Options:

--loglevel=DEBUG Verboseness of output. [default: %(default)s]
Possible choices: DEBUG, INFO, WARNING, ERROR, CRITICAL, EXCEPTION
--version, -V show program's version number and exit
--tmpDir=/tmp Base directory for temp files. [default: %(default)s]
--tmpDirKeep=False Keep the tmpDir if an exception occurs while running. Default is to delete all temp files at the end, even if there's a failure.

index_fasta_samtools Undocumented

Index a reference genome for Samtools.

```
usage: read_utils.py index_fasta_samtools [-h]
                                     [--loglevel {DEBUG, INFO, WARNING, ERROR, CRITICAL, EXCEPTION}]
                                     [--version]
                                     inFasta
```

Positional arguments:

inFasta Reference genome, FASTA format.

Options:

--loglevel=DEBUG Verboseness of output. [default: %(default)s]
Possible choices: DEBUG, INFO, WARNING, ERROR, CRITICAL, EXCEPTION
--version, -V show program's version number and exit

index_fasta_picard Undocumented

Create an index file for a reference genome suitable for Picard/GATK.

```
usage: read_utils.py index_fasta_picard [-h] [--JVMmemory JVMMEMORY]
                                     [--picardOptions [PICARDOPTIONS [PICARDOPTIONNAME=value ...]]]
                                     [--loglevel {DEBUG, INFO, WARNING, ERROR, CRITICAL, EXCEPTION}]
                                     [--version] [--tmpDir TMPDIR]
                                     [--tmpDirKeep]
                                     inFasta
```

Positional arguments:

inFasta Input reference genome, FASTA format.

Options:

--JVMmemory=512m JVM virtual memory size (default: %(default)s)
--picardOptions=[] Optional arguments to Picard's CreateSequenceDictionary, OPTIONNAME=value ...
--loglevel=DEBUG Verboseness of output. [default: %(default)s]
Possible choices: DEBUG, INFO, WARNING, ERROR, CRITICAL, EXCEPTION

--version, -V show program's version number and exit

--tmpDir=/tmp Base directory for temp files. [default: %(default)s]

--tmpDirKeep=False Keep the tmpDir if an exception occurs while running. Default is to delete all temp files at the end, even if there's a failure.

mkdup_picard Undocumented

Mark or remove duplicate reads from BAM file.

```
usage: read_utils.py mkdup_picard [-h] [--outMetrics OUTMETRICS] [--remove]
                                   [--JVMmemory JVMMEMORY]
                                   [--picardOptions [PICARDOPTIONS [PICARDOPTIONS ...]]]
                                   [--loglevel {DEBUG,INFO,WARNING,ERROR,CRITICAL,EXCEPTION}]
                                   [--version] [--tmpDir TMPDIR] [--tmpDirKeep]
                                   inBams [inBams ...] outBam
```

Positional arguments:

inBams Input reads, BAM format.

outBam Output reads, BAM format.

Options:

--outMetrics Output metrics file. Default is to dump to a temp file.

--remove=False Instead of marking duplicates, remove them entirely (default: %(default)s)

--JVMmemory=2g JVM virtual memory size (default: %(default)s)

--picardOptions=[] Optional arguments to Picard's MarkDuplicates, OPTION-NAME=value ...

--loglevel=DEBUG Verboseness of output. [default: %(default)s]

Possible choices: DEBUG, INFO, WARNING, ERROR, CRITICAL, EXCEPTION

--version, -V show program's version number and exit

--tmpDir=/tmp Base directory for temp files. [default: %(default)s]

--tmpDirKeep=False Keep the tmpDir if an exception occurs while running. Default is to delete all temp files at the end, even if there's a failure.

revert_bam_picard Undocumented

Revert BAM to raw reads

```
usage: read_utils.py revert_bam_picard [-h] [--JVMmemory JVMMEMORY]
                                         [--picardOptions [PICARDOPTIONS [PICARDOPTIONS ...]]]
                                         [--loglevel {DEBUG,INFO,WARNING,ERROR,CRITICAL,EXCEPTION}]
                                         [--version] [--tmpDir TMPDIR]
                                         [--tmpDirKeep]
                                         inBam outBam
```

Positional arguments:

inBam Input reads, BAM format.

outBam Output reads, BAM format.

Options:

--JVMmemory=2g JVM virtual memory size (default: %(default)s)
--picardOptions=[] Optional arguments to Picard's RevertSam, OPTION-NAME=value ...
--loglevel=DEBUG Verboseness of output. [default: %(default)s]
Possible choices: DEBUG, INFO, WARNING, ERROR, CRITICAL, EXCEPTION
--version, -V show program's version number and exit
--tmpDir=/tmp Base directory for temp files. [default: %(default)s]
--tmpDirKeep=False Keep the tmpDir if an exception occurs while running. Default is to delete all temp files at the end, even if there's a failure.

picard Undocumented

Generic Picard runner.

```
usage: read_utils.py picard [-h] [--JVMmemory JVMMEMORY]
                             [--picardOptions [PICARDOPTIONS [PICARDOPTIONS ...]]]
                             [--loglevel {DEBUG, INFO, WARNING, ERROR, CRITICAL, EXCEPTION}]
                             [--version] [--tmpDir TMPDIR] [--tmpDirKeep]
                             command
```

Positional arguments:

command	picard command
----------------	----------------

Options:

--JVMmemory=2g JVM virtual memory size (default: %(default)s)
--picardOptions=[] Optional arguments to Picard, OPTIONNAME=value ...
--loglevel=DEBUG Verboseness of output. [default: %(default)s]
Possible choices: DEBUG, INFO, WARNING, ERROR, CRITICAL, EXCEPTION
--version, -V show program's version number and exit
--tmpDir=/tmp Base directory for temp files. [default: %(default)s]
--tmpDirKeep=False Keep the tmpDir if an exception occurs while running. Default is to delete all temp files at the end, even if there's a failure.

sort_bam Undocumented

Sort BAM file

```
usage: read_utils.py sort_bam [-h] [--index] [--md5] [--JVMmemory JVMMEMORY]
                               [--picardOptions [PICARDOPTIONS [PICARDOPTIONS ...]]]
                               [--loglevel {DEBUG, INFO, WARNING, ERROR, CRITICAL, EXCEPTION}]
                               [--version] [--tmpDir TMPDIR] [--tmpDirKeep]
                               inBam outBam {unsorted, queryname, coordinate}
```

Positional arguments:

inBam	Input bam file.
outBam	Output bam file, sorted.

sortOrder How to sort the reads. [default: %(default)s]
Possible choices: unsorted, queryname, coordinate

Options:

--index=False Index outBam (default: %(default)s)
--md5=False MD5 checksum outBam (default: %(default)s)
--JVMmemory=2g JVM virtual memory size (default: %(default)s)
--picardOptions=[] Optional arguments to Picard's SortSam, OPTIONNAME=value ...
--loglevel=DEBUG Verboseness of output. [default: %(default)s]
Possible choices: DEBUG, INFO, WARNING, ERROR, CRITICAL, EXCEPTION
--version, -V show program's version number and exit
--tmpDir=/tmp Base directory for temp files. [default: %(default)s]
--tmpDirKeep=False Keep the tmpDir if an exception occurs while running. Default is to delete all temp files at the end, even if there's a failure.

merge_bams Undocumented

Merge multiple BAMs into one

```
usage: read_utils.py merge_bams [-h] [--JVMmemory JVMMEMORY]
                                [--picardOptions [PICARDOPTIONS [PICARDOPTIONS ...]]]
                                [--loglevel {DEBUG,INFO,WARNING,ERROR,CRITICAL,EXCEPTION}]
                                [--version] [--tmpDir TMPDIR] [--tmpDirKeep]
                                inBams [inBams ...] outBam
```

Positional arguments:

inBams Input bam files.
outBam Output bam file.

Options:

--JVMmemory=2g JVM virtual memory size (default: %(default)s)
--picardOptions=[] Optional arguments to Picard's MergeSamFiles, OPTION-NAME=value ...
--loglevel=DEBUG Verboseness of output. [default: %(default)s]
Possible choices: DEBUG, INFO, WARNING, ERROR, CRITICAL, EXCEPTION
--version, -V show program's version number and exit
--tmpDir=/tmp Base directory for temp files. [default: %(default)s]
--tmpDirKeep=False Keep the tmpDir if an exception occurs while running. Default is to delete all temp files at the end, even if there's a failure.

filter_bam Undocumented

Filter BAM file by read name

```
usage: read_utils.py filter_bam [-h] [--exclude] [--JVMmemory JVMMEMORY]
                                [--picardOptions [PICARDOPTIONS [PICARDOPTIONS ...]]]
                                [--loglevel {DEBUG,INFO,WARNING,ERROR,CRITICAL,EXCEPTION}]
                                [--version] [--tmpDir TMPDIR] [--tmpDirKeep]
                                inBam readList outBam
```

Positional arguments:

inBam	Input bam file.
readList	Input file of read IDs.
outBam	Output bam file.

Options:

--exclude=False	If specified, readList is a list of reads to remove from input. Default behavior is to treat readList as an inclusion list (all unnamed reads are removed).
--JVMmemory=4g	JVM virtual memory size (default: %(default)s)
--picardOptions=[]	Optional arguments to Picard's FilterSamReads, OPTION-NAME=value ...
--loglevel=DEBUG	Verboseness of output. [default: %(default)s] Possible choices: DEBUG, INFO, WARNING, ERROR, CRITICAL, EXCEPTION
--version, -V	show program's version number and exit
--tmpDir=/tmp	Base directory for temp files. [default: %(default)s]
--tmpDirKeep=False	Keep the tmpDir if an exception occurs while running. Default is to delete all temp files at the end, even if there's a failure.

bam_to_fastq Undocumented

Convert a bam file to a pair of fastq paired-end read files and optional text header.

```
usage: read_utils.py bam_to_fastq [-h] [--outHeader OUTHEADER]
                                   [--JVMmemory JVMMEMORY]
                                   [--picardOptions [PICARDOPTIONS [PICARDOPTIONS ...]]]
                                   [--loglevel {DEBUG,INFO,WARNING,ERROR,CRITICAL,EXCEPTION}]
                                   [--version] [--tmpDir TMPDIR] [--tmpDirKeep]
                                   inBam outFastq1 outFastq2
```

Positional arguments:

inBam	Input bam file.
outFastq1	Output fastq file; 1st end of paired-end reads.
outFastq2	Output fastq file; 2nd end of paired-end reads.

Options:

--outHeader	Optional text file name that will receive bam header.
--JVMmemory=2g	JVM virtual memory size (default: %(default)s)
--picardOptions=[]	Optional arguments to Picard's SamToFastq, OPTION-NAME=value ...

--loglevel=DEBUG Verboseness of output. [default: %(default)s]
Possible choices: DEBUG, INFO, WARNING, ERROR, CRITICAL, EXCEPTION

--version, -V show program's version number and exit

--tmpDir=/tmp Base directory for temp files. [default: %(default)s]

--tmpDirKeep=False Keep the tmpDir if an exception occurs while running. Default is to delete all temp files at the end, even if there's a failure.

fastq_to_bam Undocumented

Convert a pair of fastq paired-end read files and optional text header to a single bam file.

```
usage: read_utils.py fastq_to_bam [-h]
                                (--sampleName SAMPLENAME | --header HEADER)
                                [--JVMmemory JVMMEMORY]
                                [--picardOptions [PICARDOPTIONS [PICARDOPTIONS ...]]]
                                [--loglevel {DEBUG,INFO,WARNING,ERROR,CRITICAL,EXCEPTION}]
                                [--version] [--tmpDir TMPDIR] [--tmpDirKeep]
                                inFastq1 inFastq2 outBam
```

Positional arguments:

inFastq1 Input fastq file; 1st end of paired-end reads.

inFastq2 Input fastq file; 2nd end of paired-end reads.

outBam Output bam file.

Options:

--sampleName Sample name to insert into the read group header.

--header Optional text file containing header.

--JVMmemory=2g JVM virtual memory size (default: %(default)s)

--picardOptions=[] Optional arguments to Picard's FastqToSam, OPTION-NAME=value ... Note that header-related options will be overwritten by HEADER if present.

--loglevel=DEBUG Verboseness of output. [default: %(default)s]
Possible choices: DEBUG, INFO, WARNING, ERROR, CRITICAL, EXCEPTION

--version, -V show program's version number and exit

--tmpDir=/tmp Base directory for temp files. [default: %(default)s]

--tmpDirKeep=False Keep the tmpDir if an exception occurs while running. Default is to delete all temp files at the end, even if there's a failure.

split_reads Undocumented

Split fasta or fastq file into chunks of maxReads reads or into numChunks chunks named outPrefix01, outPrefix02, etc. If both maxReads and numChunks are None, use defaultMaxReads. The number of characters in file names after outPrefix is indexLen; if not specified, use defaultIndexLen.

```
usage: read_utils.py split_reads [-h]
                                [--maxReads MAXREADS | --numChunks NUMCHUNKS]
                                [--indexLen INDEXLEN]
                                [--format {fastq,fasta}]
```

```
[--outSuffix OUTSUFFIX]
inFileName outPrefix
```

Positional arguments:

inFileName	Input fastq or fasta file.
outPrefix	Output files will be named \${outPrefix}01\${outSuffix}, \${outPrefix}02\${outSuffix}...

Options:

--maxReads	Maximum number of reads per chunk (default 1000 if neither maxReads nor numChunks is specified).
--numChunks	Number of output files, if maxReads is not specified.
--indexLen=2	Number of characters to append to outputPrefix for each output file (default %(default)s). Number of files must not exceed 10^INDEXLEN.
--format=fastq	Input fastq or fasta file (default: %(default)s). Possible choices: fastq, fasta
--outSuffix=	Output filename suffix (e.g. .fastq or .fastq.gz). A suffix ending in .gz will cause the output file to be gzip compressed. Default is no suffix.

split_bam Undocumented

Split BAM file equally into several output BAM files.

```
usage: read_utils.py split_bam [-h]
                                [--loglevel {DEBUG,INFO,WARNING,ERROR,CRITICAL,EXCEPTION}]
                                [--version] [--tmpDir TMPDIR] [--tmpDirKeep]
                                inBam outBams [outBams ...]
```

Positional arguments:

inBam	Input BAM file.
outBams	Output BAM files

Options:

--loglevel=DEBUG	Verboseness of output. [default: %(default)s] Possible choices: DEBUG, INFO, WARNING, ERROR, CRITICAL, EXCEPTION
--version, -V	show program's version number and exit
--tmpDir=/tmp	Base directory for temp files. [default: %(default)s]
--tmpDirKeep=False	Keep the tmpDir if an exception occurs while running. Default is to delete all temp files at the end, even if there's a failure.

rmdup_mvicuna_bam Undocumented

Remove duplicate reads from BAM file using M-Vicuna. The primary advantage to this approach over Picard's MarkDuplicates tool is that Picard requires that input reads are aligned to a reference, and M-Vicuna can operate on unaligned reads.

```
usage: read_utils.py rmdup_mvicuna_bam [-h] [--JVMmemory JVMMEMORY]
                                         [--loglevel {DEBUG,INFO,WARNING,ERROR,CRITICAL,EXCEPTION}]
```

```

[--version] [--tmpDir TMPDIR]
[--tmpDirKeep]
inBam outBam

```

Positional arguments:

inBam Input reads, BAM format.

outBam Output reads, BAM format.

Options:

--JVMmemory=4g JVM virtual memory size (default: %(default)s)

--loglevel=DEBUG Verboseness of output. [default: %(default)s]

 Possible choices: DEBUG, INFO, WARNING, ERROR, CRITICAL, EXCEPTION

--version, -V show program's version number and exit

--tmpDir=/tmp Base directory for temp files. [default: %(default)s]

--tmpDirKeep=False Keep the tmpDir if an exception occurs while running. Default is to delete all temp files at the end, even if there's a failure.

dup_remove_mvicuna Undocumented

Run mvicuna's duplicate removal operation on paired-end reads.

```

usage: read_utils.py dup_remove_mvicuna [-h]
                                         [--unpairedOutFastq UNPAIREDOUTFASTQ]
                                         [--loglevel {DEBUG,INFO,WARNING,ERROR,CRITICAL,EXCEPTION}]
                                         [--version] [--tmpDir TMPDIR]
                                         [--tmpDirKeep]
                                         inFastq1 inFastq2 pairedOutFastq1
                                         pairedOutFastq2

```

Positional arguments:

inFastq1 Input fastq file; 1st end of paired-end reads.

inFastq2 Input fastq file; 2nd end of paired-end reads.

pairedOutFastq1 Output fastq file; 1st end of paired-end reads.

pairedOutFastq2 Output fastq file; 2nd end of paired-end reads.

Options:

--unpairedOutFastq File name of output unpaired reads

--loglevel=DEBUG Verboseness of output. [default: %(default)s]

 Possible choices: DEBUG, INFO, WARNING, ERROR, CRITICAL, EXCEPTION

--version, -V show program's version number and exit

--tmpDir=/tmp Base directory for temp files. [default: %(default)s]

--tmpDirKeep=False Keep the tmpDir if an exception occurs while running. Default is to delete all temp files at the end, even if there's a failure.

rmdup_prinseq_fastq Undocumented

Run prinseq-lite's duplicate removal operation on paired-end reads. Also removes reads with more than one N.

```
usage: read_utils.py rmdup_prinseq_fastq [-h]
                                         [--loglevel {DEBUG,INFO,WARNING,ERROR,CRITICAL,EXCEPTION}]
                                         [--version] [--tmpDir TMPDIR]
                                         [--tmpDirKeep]
                                         inFastq1 inFastq2 outFastq1 outFastq2
```

Positional arguments:

inFastq1	Input fastq file; 1st end of paired-end reads.
inFastq2	Input fastq file; 2nd end of paired-end reads.
outFastq1	Output fastq file; 1st end of paired-end reads.
outFastq2	Output fastq file; 2nd end of paired-end reads.

Options:

--loglevel=DEBUG	Verboseness of output. [default: %(default)s] Possible choices: DEBUG, INFO, WARNING, ERROR, CRITICAL, EXCEPTION
--version, -V	show program's version number and exit
--tmpDir=/tmp	Base directory for temp files. [default: %(default)s]
--tmpDirKeep=False	Keep the tmpDir if an exception occurs while running. Default is to delete all temp files at the end, even if there's a failure.

filter_bam_mapped_only Undocumented

Samtools to reduce a BAM file to only reads that are aligned (-F 4) with a non-zero mapping quality (-q 1) and are not marked as a PCR/optical duplicate (-F 1024).

```
usage: read_utils.py filter_bam_mapped_only [-h]
                                             [--loglevel {DEBUG,INFO,WARNING,ERROR,CRITICAL,EXCEPTION}]
                                             [--version] [--tmpDir TMPDIR]
                                             [--tmpDirKeep]
                                             inBam outBam
```

Positional arguments:

inBam	Input aligned reads, BAM format.
outBam	Output sorted indexed reads, filtered to aligned-only, BAM format.

Options:

--loglevel=DEBUG	Verboseness of output. [default: %(default)s] Possible choices: DEBUG, INFO, WARNING, ERROR, CRITICAL, EXCEPTION
--version, -V	show program's version number and exit
--tmpDir=/tmp	Base directory for temp files. [default: %(default)s]
--tmpDirKeep=False	Keep the tmpDir if an exception occurs while running. Default is to delete all temp files at the end, even if there's a failure.

novoalign Undocumented

Align reads with Novoalign. Sort and index BAM output.

```
usage: read_utils.py novoalign [-h] [--options OPTIONS] [--min_qual MIN_QUAL]
                                [--JVMmemory JVMMEMORY]
                                [--loglevel {DEBUG,INFO,WARNING,ERROR,CRITICAL,EXCEPTION}]
                                [--version] [--tmpDir TMPDIR] [--tmpDirKeep]
                                inBam refFasta outBam
```

Positional arguments:

inBam	Input reads, BAM format.
refFasta	Reference genome, FASTA format, pre-indexed by Novoindex.
outBam	Output reads, BAM format (aligned).

Options:

--options=-r Random	Novoalign options (default: %(default)s)
--min_qual=0	Filter outBam to minimum mapping quality (default: %(default)s)
--JVMmemory=2g	JVM virtual memory size (default: %(default)s)
--loglevel=DEBUG	Verboseness of output. [default: %(default)s] Possible choices: DEBUG, INFO, WARNING, ERROR, CRITICAL, EXCEPTION
--version, -V	show program's version number and exit
--tmpDir=/tmp	Base directory for temp files. [default: %(default)s]
--tmpDirKeep=False	Keep the tmpDir if an exception occurs while running. Default is to delete all temp files at the end, even if there's a failure.

novoindex Undocumented

Index a FASTA file (reference genome) for use with Novoalign. The input file name must end in ".fasta". This will create a new ".nix" file in the same directory. If it already exists, it will be deleted and regenerated.

```
usage: read_utils.py novoindex [-h]
                                [--loglevel {DEBUG,INFO,WARNING,ERROR,CRITICAL,EXCEPTION}]
                                [--version]
                                refFasta
```

Positional arguments:

refFasta	Reference genome, FASTA format.
-----------------	---------------------------------

Options:

--loglevel=DEBUG	Verboseness of output. [default: %(default)s] Possible choices: DEBUG, INFO, WARNING, ERROR, CRITICAL, EXCEPTION
--version, -V	show program's version number and exit

gatk Ug Undocumented

Call genotypes using the GATK UnifiedGenotyper.

```
usage: read_utils.py gatk Ug [-h] [--options OPTIONS] [--JVMmemory JVMMEMORY]
                             [--loglevel {DEBUG,INFO,WARNING,ERROR,CRITICAL,EXCEPTION}]
                             [--version] [--tmpDir TMPDIR] [--tmpDirKeep]
                             inBam refFasta outVcf
```

Positional arguments:

inBam	Input reads, BAM format.
refFasta	Reference genome, FASTA format, pre-indexed by Picard.
outVcf	Output calls in VCF format. If this filename ends with .gz, GATK will BGZIP compress the output and produce a Tabix index file as well.

Options:

--options=--min_base_quality_score 15 -ploidy 4	UnifiedGenotyper options (default: %(default)s)
--JVMmemory=2g	JVM virtual memory size (default: %(default)s)
--loglevel=DEBUG	Verboseness of output. [default: %(default)s] Possible choices: DEBUG, INFO, WARNING, ERROR, CRITICAL, EXCEPTION
--version, -V	show program's version number and exit
--tmpDir=/tmp	Base directory for temp files. [default: %(default)s]
--tmpDirKeep=False	Keep the tmpDir if an exception occurs while running. Default is to delete all temp files at the end, even if there's a failure.

gatk_realign Undocumented

Local realignment of BAM files with GATK IndelRealigner.

```
usage: read_utils.py gatk_realign [-h] [--JVMmemory JVMMEMORY]
                                   [--loglevel {DEBUG,INFO,WARNING,ERROR,CRITICAL,EXCEPTION}]
                                   [--version] [--tmpDir TMPDIR] [--tmpDirKeep]
                                   [--threads THREADS]
                                   inBam refFasta outBam
```

Positional arguments:

inBam	Input reads, BAM format, aligned to refFasta.
refFasta	Reference genome, FASTA format, pre-indexed by Picard.
outBam	Realigned reads.

Options:

--JVMmemory=2g	JVM virtual memory size (default: %(default)s)
--loglevel=DEBUG	Verboseness of output. [default: %(default)s] Possible choices: DEBUG, INFO, WARNING, ERROR, CRITICAL, EXCEPTION
--version, -V	show program's version number and exit
--tmpDir=/tmp	Base directory for temp files. [default: %(default)s]
--tmpDirKeep=False	Keep the tmpDir if an exception occurs while running. Default is to delete all temp files at the end, even if there's a failure.

--threads=1 Number of threads (default: %(default)s)

align_and_fix Undocumented

Take reads, align to reference with Novoalign, mark duplicates with Picard, realign indels with GATK, and optionally filter final file to mapped/non-dupe reads.

```
usage: read_utils.py align_and_fix [-h] [--outBamAll OUTBAMALL]
                                   [--outBamFiltered OUTBAMFILTERED]
                                   [--novoalign_options NOVOALIGN_OPTIONS]
                                   [--JVMmemory JVMMEMORY] [--threads THREADS]
                                   [--loglevel {DEBUG,INFO,WARNING,ERROR,CRITICAL,ERROR}
                                   [--version] [--tmpDir TMPDIR]
                                   [--tmpDirKeep]
                                   inBam refFasta
```

Positional arguments:

inBam Input unaligned reads, BAM format.

refFasta Reference genome, FASTA format, pre-indexed by Picard and Novoalign.

Options:

--outBamAll Aligned, sorted, and indexed reads. Unmapped reads are retained and duplicate reads are marked, not removed.

--outBamFiltered Aligned, sorted, and indexed reads. Unmapped reads and duplicate reads are removed from this file.

--novoalign_options=-r Random Novoalign options (default: %(default)s)

--JVMmemory=4g JVM virtual memory size (default: %(default)s)

--threads=1 Number of threads (default: %(default)s)

--loglevel=DEBUG Verboseness of output. [default: %(default)s]

Possible choices: DEBUG, INFO, WARNING, ERROR, CRITICAL, EXCEPTION

--version, -V show program's version number and exit

--tmpDir=/tmp Base directory for temp files. [default: %(default)s]

--tmpDirKeep=False Keep the tmpDir if an exception occurs while running. Default is to delete all temp files at the end, even if there's a failure.

1.3.6 reports.py - produce various metrics and reports

Reports

```
usage: reports.py subcommand
```

Sub-commands:

assembly_stats Undocumented

Fetch assembly-level statistics for a given sample

```
usage: reports.py assembly_stats [-h]
                                   [--cov_thresholds COV_THRESHOLDS [COV_THRESHOLDS .
                                   [--assembly_dir ASSEMBLY_DIR]
```

```
[--assembly_tmp ASSEMBLY_TMP]
[--align_dir ALIGN_DIR]
[--reads_dir READS_DIR]
[--raw_reads_dir RAW_READS_DIR]
samples [samples ...] outFile
```

Positional arguments:

samples	Sample names.
outFile	Output report file.

Options:

```
--cov_thresholds=(1, 5, 20, 100)  Genome coverage thresholds to report on. (default:
%(default)s)
--assembly_dir=data/02_assembly  Directory with assembly outputs. (default: %(de-
fault)s)
--assembly_tmp=tmp/02_assembly  Directory with assembly temp files. (default:
%(default)s)
--align_dir=data/02_align_to_self  Directory with reads aligned to own assembly.
(default: %(default)s)
--reads_dir=data/01_per_sample  Directory with unaligned filtered read BAMs. (de-
fault: %(default)s)
--raw_reads_dir=data/00_raw  Directory with unaligned raw read BAMs. (default:
%(default)s)
```

consolidate_fastqc Undocumented

Consolidate multiple FASTQC reports into one.

```
usage: reports.py consolidate_fastqc [-h] inDirs [inDirs ...] outFile
```

Positional arguments:

inDirs	Input FASTQC directories.
outFile	Output report file.

consolidate_spike_count Undocumented

Consolidate multiple spike count reports into one.

```
usage: reports.py consolidate_spike_count [-h] inDir outFile
```

Positional arguments:

inDir	Input spike count directory.
outFile	Output report file.

1.3.7 broad_utils.py - for data generated at the Broad Institute

Utilities for getting sequences out of the Broad walk-up sequencing pipeline. These utilities are probably not of much use outside the Broad.

```
usage: broad_utils.py subcommand
```

Sub-commands:

get_bustard_dir Undocumented

Find the basecalls directory from a Picard directory

```
usage: broad_utils.py get_bustard_dir [-h]
                                     [--loglevel {DEBUG,INFO,WARNING,ERROR,CRITICAL}]
                                     inDir
```

Positional arguments:

inDir Picard directory

Options:

--loglevel=ERROR Verboseness of output. [default: %(default)s]
Possible choices: DEBUG, INFO, WARNING, ERROR, CRITICAL, EXCEPTION

get_run_date Undocumented

Find the sequencing run date from a Picard directory

```
usage: broad_utils.py get_run_date [-h]
                                   [--loglevel {DEBUG,INFO,WARNING,ERROR,CRITICAL,EXCEPTION}]
                                   inDir
```

Positional arguments:

inDir Picard directory

Options:

--loglevel=ERROR Verboseness of output. [default: %(default)s]
Possible choices: DEBUG, INFO, WARNING, ERROR, CRITICAL, EXCEPTION

get_all_names Undocumented

Get all samples

```
usage: broad_utils.py get_all_names [-h]
                                   [--loglevel {DEBUG,INFO,WARNING,ERROR,CRITICAL,EXCEPTION}]
                                   {samples,libraries,runs} runfile
```

Positional arguments:

type Type of name
Possible choices: samples, libraries, runs
runfile File with seq run information

Options:

--loglevel=ERROR Verboseness of output. [default: %(default)s]
Possible choices: DEBUG, INFO, WARNING, ERROR, CRITICAL, EXCEPTION

make_barcode_file Undocumented

Create input file for extract_barcode

```
usage: broad_utils.py make_barcode_file [-h] inFile outFile
```

Positional arguments:

inFile	Input tab file w/header and 2-5 named columns (last three are optional): sample, barcode_1, barcode_2, library_id_per_sample, run_id_per_library
outFile	Output BARCODE_FILE file for Picard.

extract_barcodes Undocumented

Match every read in a lane against their barcode.

```
usage: broad_utils.py extract_barcodes [-h] [--outMetrics OUTMETRICS]
                                         [--read_structure READ_STRUCTURE]
                                         [--max_mismatches MAX_MISMATCHES]
                                         [--minimum_base_quality MINIMUM_BASE_QUALITY]
                                         [--min_mismatch_delta MIN_MISMATCH_DELTA]
                                         [--max_no_calls MAX_NO_CALLS]
                                         [--minimum_quality MINIMUM_QUALITY]
                                         [--compress_outputs COMPRESS_OUTPUTS]
                                         [--num_processors NUM_PROCESSORS]
                                         [--JVMmemory JVMMEMORY]
                                         [--loglevel {DEBUG,INFO,WARNING,ERROR,CRITICAL}]
                                         [--version] [--tmpDir TMPDIR]
                                         [--tmpDirKeep]
                                         inDir lane barcodeFile outDir
```

Positional arguments:

inDir	Bustard directory.
lane	Lane number.
barcodeFile	Input tab file w/header and four named columns: barcode_name, library_name, barcode_sequence_1, barcode_sequence_2
outDir	Output directory for barcodes.

Options:

--outMetrics	Output metrics file. Default is to dump to a temp file.
--read_structure=101T8B8B101T	Picard ExtractIlluminaBarcodes READ_STRUCTURE (default: %(default)s)
--max_mismatches=0	Picard ExtractIlluminaBarcodes MAX_MISMATCHES (default: %(default)s)
--minimum_base_quality=25	Picard ExtractIlluminaBarcodes MINIMUM_BASE_QUALITY (default: %(default)s)
--min_mismatch_delta	Picard ExtractIlluminaBarcodes MIN_MISMATCH_DELTA (default: %(default)s)
--max_no_calls	Picard ExtractIlluminaBarcodes MAX_NO_CALLS (default: %(default)s)
--minimum_quality	Picard ExtractIlluminaBarcodes MINIMUM_QUALITY (default: %(default)s)
--compress_outputs	Picard ExtractIlluminaBarcodes COMPRESS_OUTPUTS (default: %(default)s)
--num_processors=8	Picard ExtractIlluminaBarcodes NUM_PROCESSORS (default: %(default)s)

--JVMmemory=8g JVM virtual memory size (default: %(default)s)

--loglevel=DEBUG Verboseness of output. [default: %(default)s]
Possible choices: DEBUG, INFO, WARNING, ERROR, CRITICAL, EXCEPTION

--version, -V show program's version number and exit

--tmpDir=/tmp Base directory for temp files. [default: %(default)s]

--tmpDirKeep=False Keep the tmpDir if an exception occurs while running. Default is to delete all temp files at the end, even if there's a failure.

make_params_file Undocumented

Create input file for illumina_basecalls

usage: broad_utils.py make_params_file [-h] inFile bamDir outFile

Positional arguments:

inFile Input tab file w/header and four named columns: barcode_name, library_name, barcode_sequence_1, barcode_sequence_2

bamDir Directory for output bams

outFile Output LIBRARY_PARAMS file for Picard

illumina_basecalls Undocumented

Demultiplex Illumina runs & produce BAM files, one per sample

usage: broad_utils.py illumina_basecalls [-h]
 [--read_structure READ_STRUCTURE]
 [--sequencing_center SEQUENCING_CENTER]
 [--adapters_to_check [ADAPTERS_TO_CHECK [A
 [--platform PLATFORM]
 [--max_reads_in_ram_per_tile MAX_READS_IN
 [--max_records_in_ram MAX_RECORDS_IN_RAM]
 [--num_processors NUM_PROCESSORS]
 [--apply_eamss_filter APPLY_EAMSS_FILTER]
 [--force_gc FORCE_GC]
 [--first_tile FIRST_TILE]
 [--tile_limit TILE_LIMIT]
 [--include_non_pf_reads INCLUDE_NON_PF_REA
 [--run_start_date RUN_START_DATE]
 [--read_group_id READ_GROUP_ID]
 [--JVMmemory JVMMEMORY]
 [--loglevel {DEBUG,INFO,WARNING,ERROR,CRIT
 [--version] [--tmpDir TMPDIR]
 [--tmpDirKeep]
 inBustardDir inBarcodesDir flowcell
 lane paramsFile

Positional arguments:

inBustardDir Bustard directory.

inBarcodesDir Barcodes directory.

flowcell Flowcell ID

lane Lane number.

paramsFile Input tab file w/header and five named columns: BAR-CODE_1, BARCODE_2, OUTPUT, SAMPLE_ALIAS, LIBRARY_NAME

Options:

--read_structure=101T8B8B101T Picard ExtractIlluminaBarcodes
READ_STRUCTURE (default: %(default)s)

--sequencing_center=BI Picard ExtractIlluminaBarcodes SEQUENCING_CENTER
(default: %(default)s)

--adapters_to_check=('PAIRED_END', 'NEXTERA_V1', 'NEXTERA_V2')
Picard ExtractIlluminaBarcodes ADAPTERS_TO_CHECK
(default: %(default)s)

--platform Picard ExtractIlluminaBarcodes PLATFORM (default: %(default)s)

--max_reads_in_ram_per_tile=100000 Picard ExtractIlluminaBarcodes
MAX_READS_IN_RAM_PER_TILE (default: %(default)s)

--max_records_in_ram=100000 Picard ExtractIlluminaBarcodes
MAX_RECORDS_IN_RAM (default: %(default)s)

--num_processors=8 Picard ExtractIlluminaBarcodes NUM_PROCESSORS (default: %(default)s)

--apply_eamss_filter Picard ExtractIlluminaBarcodes APPLY_EAMSS_FILTER
(default: %(default)s)

--force_gc=False Picard ExtractIlluminaBarcodes FORCE_GC (default: %(default)s)

--first_tile Picard ExtractIlluminaBarcodes FIRST_TILE (default: %(default)s)

--tile_limit Picard ExtractIlluminaBarcodes TILE_LIMIT (default: %(default)s)

--include_non_pf_reads Picard ExtractIlluminaBarcodes INCLUDE_NON_PF_READS (default: %(default)s)

--run_start_date Picard ExtractIlluminaBarcodes RUN_START_DATE (default: %(default)s)

--read_group_id Picard ExtractIlluminaBarcodes READ_GROUP_ID (default: %(default)s)

--JVMmemory=54g JVM virtual memory size (default: %(default)s)

--loglevel=DEBUG Verboseness of output. [default: %(default)s]

Possible choices: DEBUG, INFO, WARNING, ERROR, CRITICAL, EXCEPTION

--version, -V show program's version number and exit

--tmpDir=/tmp Base directory for temp files. [default: %(default)s]

--tmpDirKeep=False Keep the tmpDir if an exception occurs while running. Default is to delete all temp files at the end, even if there's a failure.

1.4 Using the Snakemake pipelines

Rather than chaining together viral-ngs pipeline steps as series of tool commands called in isolation, it is possible to execute them as a complete automated pipeline, from processing raw sequencer output to creating files suitable for GenBank submission. This utilizes Snakemake, which is documented at: <https://bitbucket.org/johanneskoester/snakemake/wiki/Home>

Here is an overview of the Snakemake rule graph:



1.4.1 Setting up the Python 3 virtual environment

Note that Python 3.4 is required to use these tools with Snakemake. It is recommended to create a virtual environment within which all of the viral-ngs dependencies can be installed:

```
pyvenv-3.4 venv-viral-ngs
cd venv-viral-ngs
source bin/activate
```

Once the virtual environment has been created and activated, the viral-ngs dependencies can be installed via `pip`:

```
pip install -r requirements.txt
pip install -r requirements-pipes.txt
```

Note: To resume normal use of the system installation of python, call the “deactivate” command in your shell. See the [official venv documentation](#) for more information on Python3 virtual environments.

In addition to the dependencies installed via `pip`, the pipeline needs the standard dependencies described in the main viral-ngs installation section.

Note: If running on the Broad Institute UGER cluster environment, import the following dotkits prior to activating the virtualenv:

```
use .python-3.4.3
use .oracle-java-jdk-1.7.0-51-x86-64
use .bzip2-1.0.6
use .zlib-1.2.6
use .gcc-4.5.3
```

1.4.2 Setting up an analysis directory

Copying and creating project directories and files

The Snakemake pipeline is intended to be run on an input one or more sequencer bam files, each having a filename representing a sample name. The output files are named with the same sample names, and are organized into folders corresponding to the steps of the pipeline in which they were created.

To get started, create an analysis directory somewhere in your compute environment to contain the pipeline input and output files.

Into this directory, copy the following file from the `viral-ngs/pipes` directory:

```
config.json
Snakefile
```

Since the file `config.json` is project-specific, you will need to make changes to it as appropriate for your usage. The config file changes are described in greater detail below.

Next, `cd` to the analysis directory and create symbolic links to the following:

- The viral-ngs virtual environment:
`ln -s /path/to/venv-viral-ngs venv`
- The viral-ngs project, checked out from GitHub or extracted from a version-tagged archive:
`ln -s /path/to/viral-ngs bin`

Within the analysis directory, create the directories and files used by the Snakemake pipeline:

```
data/
  00_raw/
  01_cleaned/
  01_per_sample/
  02_align_to_self/
  02_assembly/
  03_align_to_ref/
  03_interhost/
  04_intrahost/
log/
reports/
tmp/
```

The directory structure created needs to match the locations specified in `config.json`.

Adding input data

- Copy each of the raw sample bam files to the `00_raw/` directory and ensure the file names follow the convention of `{sample}.bam`.
- Create a file, `samples-depletion.txt`, to list all of the samples that should be run through the depletion pipeline, with one samplename per line as `{sample}`, following the format of the input bam file: `{sample}.bam`. For example, if you copied a file called “G1190.bam” into `00_raw/`, then then `samples-depletion.txt` would contain the line:

```
G1190
```

- Create a file, `samples-assembly.txt`, to list all of the samples that should be run through the assembly pipeline.
- Create a file, `samples-runs.txt`, to list all of the samples that should be run through the interhost analysis pipeline.
- Create a blank file, `samples-assembly-failures.txt`, that may be filled in later.

Modifying the `config.json` file

Minimal modification to the config file is necessary, though there are a few things you need to specify:

An email address for when the pipeline fetches reference data from the NCBI via their [Entrez API](#):

```
"email_point_of_contact_for_ncbi" : "someone@example.com"
```

The number of chromosomes (or segments) in the reference organism:

```
"number_of_chromosomes": 2,
```

The path to the depletion databases to be used by BMTagger, along with the file prefixes of the specific databases to use. The process for creating BMTagger depletion databases is described in the [NIH BMTagger docs](#).

```
"bmTaggerDbDir": "/path/to/depletion_databases",
"bmTaggerDbs_remove": [
  "hg19",
  "GRCh37.68_ncRNA-GRCh37.68_transcripts-HS_rRNA_mitRNA",
  "metagenomics_contaminants_v3"
]
```

In addition to the databases used by BMTagger, you will need to specify the location and file prefix of the BLAST database to be used for depletion. The process for creating the BLAST database is described in the [NIH BLAST docs](#), and on [this website](#) from the University of Oxford.

```
"blastDbDir":      "/path/to/depletion_databases",
"blastDb_remove": "metag_v3.ncRNA.mRNA.mitRNA.consensus",
```

An array of the [NCBI GenBank CoreNucleotide](#) accessions for the sequences comprising the reference genome to be used for contig assembly as well as for interhost and intrahost variant analysis. In addition, you will need to specify a file prefix to be used to represent the full reference genome file used downstream.

```
"accessions_for_ref_genome_build": ["KJ660346.2"],
"ref_genome_file_prefix": "ebov",
```

A FASTA file to be used by Trimmomatic during assembly to remove contaminants from reads:

```
"trim_clipDb":      "/path/to/depletion_databases/contaminants.fasta",
```

A FASTA file containing spike-ins to be reported:

```
"spikeinsDb":      "/path/to/references/ercc_spike-ins.fasta",
```

Modifying the Snakefile

Depending on the state of your input data, and where in the pipeline it may enter, it may be necessary to omit certain processing steps. For example, if your sequencing center has already demultiplexed your data and no demultiplexing is needed, you can comment out the following line in the Snakefile:

```
include: os.path.join(pipesDir, 'demux.rules')
```

1.4.3 Running the pipeline

Configuring for your compute platform

Running the pipeline directly

After the above setup is complete, run the pipeline directly by calling `snakemake` within the analysis directory.

Running the pipeline on GridEngine (UGER)

Within `config.json`, set the “project” to one that exists on the cluster system.

Inside the analysis directory, run the job submission command. Ex.:

```
use UGER
qsub -cwd -q long -l m_mem_free=4G ./bin/pipes/Broad_UGER/run-pipe.sh
```

To kill all jobs that exited (`qstat` status “Eqw”) with an error:

```
qdel $(qstat | grep Eqw | awk '{print $1}')
```

Running the pipeline on LSF

Inside the analysis directory, run the job submission command. Ex.:

```
bsub -o log/run.out -q forest ./bin/pipes/Broad_LSF/run-pipe.sh
```

If you notice jobs hanging in the **PEND** state, an upstream job may have failed. Before killing such jobs, verify that the jobs are pending due to their dependency:

```
bjobs -al | grep -A 1 "PENDING REASONS" | grep -v "PENDING REASONS" | grep -v '^--$'
```

To kill all **PEND**ing jobs:

```
bkill `bjobs | grep PEND | awk '{print $1}'` > /dev/null
```

When things go wrong

The pipeline may fail with errors during execution, usually while generating assemblies with Trinity. If this occurs, examine the output, add the failing sample names to `samples-assembly-failures.txt`, keeping the good ones in `samples-assembly.txt`, and re-run the pipeline. Due to sample degradation prior to sequencing in the wet lab, not all samples have the integrity to complete the pipeline, and it may necessary to skip over these samples by adding them to the `samples-assembly-failures.txt`.

1.4.4 Assembly of pre-filtered reads

1.4.5 Taxonomic filtration of raw reads

1.4.6 Starting from Illumina BCL directories