viral-ngs Documentation

Release v1.6.0

Broad Institute Viral Genomics

2016-06-30

Contents

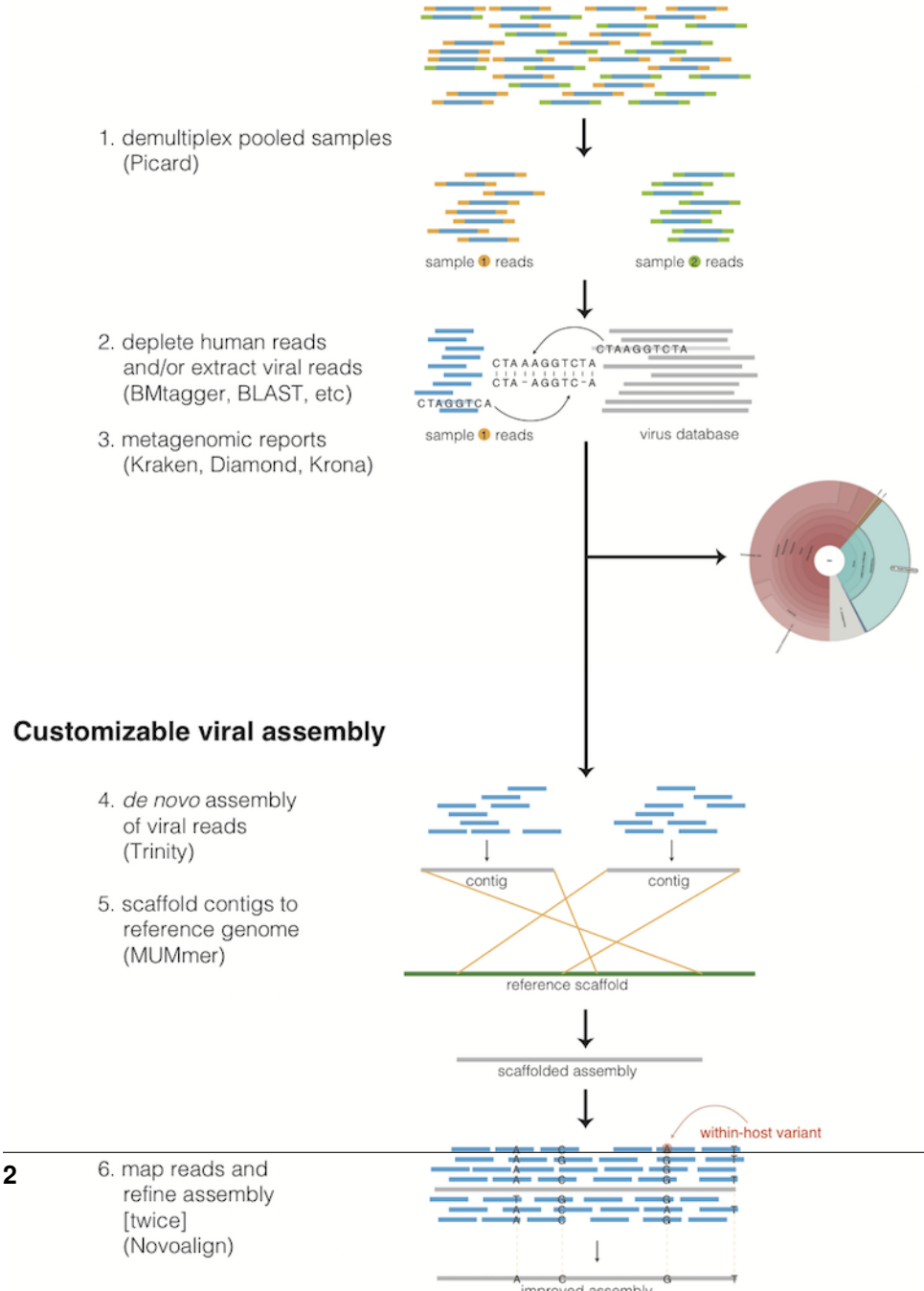
1 Contents **1**

1.1 Description of the methods 2
1.2 Installation 4
1.3 Command line tools 6
1.4 Using the Snakemake pipelines 52
1.5 Developing viral-ngs 59

CHAPTER 1

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 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 are subsampled to at most 100,000 pairs. *de novo* assemby is performed using Trinity. Reference-assisted assembly improvements follow (contig scaffolding, orienting, etc.) with MUMMER and MAFFT.

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 Manual Installation

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
python-software-properties
```

Java >= 1.7 is required by GATK and Picard.

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
pip install -r requirements-tests.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).

Tool dependencies

A lot of effort has gone into writing auto download/compile wrappers for most of the bioinformatic tools we rely on here.

Most tools will attemp a conda-based install first, before falling back to an install handled entirely by our wrappers. To make use of the conda-based install, you will need to have Anaconda or miniconda installed on your system:

http://conda.pydata.org/docs/install/quick.html#miniconda-quick-install-requirements

The tools 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:

```
py.test -v test/unit/test_tools.py
```

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.yaml 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.

1.2.2 Virtualized Installation (Easy Deploy)

The viral-ngs package includes a script that can be used to set up a complete virtualized environment for running viral-ngs either on a local machine via VirtualBox, or on AWS EC2. This is an easiesr way to get the software up and running, as it sets up most dependencies automatically within an environment known to work.

Requirements

As noted above, GATK and NovoAlign cannot be installed automatically due to licensing restrictions. In order to run the easy deployment script, you will first need to license and download these tools, and set the `GATK_PATH` and `NOVOALIGN_PATH` environment variables.

The easy deployment script has been tested to run on OS X 10.10 (Yosemite) and Ubuntu 15.04 (Vivid Vervet).

Requirements for running on AWS EC2

In order to deploy a virtualized viral-ngs environment to AWS EC2, you will first need to set up the appropriate credentials for creating EC2 instances. AWS credentials and SSH keypairs are passed in as environment variables, and `run.sh` will prompt for the values if the environment variables are not set (though the values given interactively are ephemeral).

The following environment variables are needed:

- `EC2_ACCESS_KEY_ID`
- `EC2_SECRET_ACCESS_KEY`
- `EC2_REGION` (ex. "us-west-2")
- `EC2_KEYPAIR_NAME` (ex. "my-ssh-keypair")
- `EC2_PRIVATE_KEY_PATH` (ex. "my-ssh-keypair.pem")
- `EC2_SECURITY_GROUP` (ex. "ssh-only-group")

For more information, see the following AWS pages:

- Getting set up with AWS
- How to create an AWS EC2 key pair
- Defining security group rules
- List of EC2 regions

Note that the EC2 instance created by the easy-deploy script is currently configured to be an m4.2xlarge, which costs ~$0.55/hour to run. It is suggested that the instance be terminated via the AWS web console once processing with viral-ngs is complete. See the AWS page for current pricing .

Limitations

As viral-ngs does not currently build a depletion database for BMTagger or BLAST automatically, it is the responsibility of the user to create a depletion database for use within the virtualized viral-ngs environment. It can be created within the virtual machine (VM), or uploaded after the fact via `rsync`.

Running Easy Deploy

Running Easy Deploy to create a virtualized viral-ngs environment is as simple as running `easy-deploy/run.sh`. Before running this script, copy any data you wish to have in the vm to the `easy-deploy/data` directory on your local machine. During setup, the files will be copied into the `~/data/` directory of virtual machine.

To start, the script `run.sh` installs the necessary dependencies on the user's machine (ansible, vagrant, virtualbox, and virtualbox-aws). The provisioning is handled by Ansible, with Vagrant handling creation of the VMs and EC2 instances. On OSX it depends on Homebrew, and will install it if it is not present. It depends on having apt on linux. Ruby >=2.0 is required for vagrant-aws, so versions of Ubuntu older than 15.04 (notably 14.04 LTS) will need to have ruby >=2.0 installed and made default.

Details on Easy Deploy

Per the Vagrantfile, local VM RAM usage is set to 8GB. On EC2 it currently uses an m4.2xlarge instance with 32GB of RAM and 8 vCPUs.

Ansible clones the master branch of viral-ngs from GitHub, creates a Python 3 virtual environment, and installs the viral-ngs Python dependencies. The viral-ngs tool unit tests are run to download, install, and build all of the viral-ngs tools. A `Snakefile` for viral-ngs is copied to the home directory of the VM (locally: `/home/vagrant/`, on EC2: */home/ubuntu/*), along with an associated `config.yaml` file. Files to contain sample names (`sample-depletion.txt`, etc.) are also created. A directory is created within the VM, `~/data/`, to store data to be processed. This directory on the VM is synced to the `./data/` directory on the host machine, relative to the location of the `easy-deploy/Vagrantfile`. On local VMs, syncing of the directory is two-way and fast. On EC2 instances, the syncing is currently one way (local->EC2) due to Vagrant limitations.

1.3 Command line tools

1.3.1 metagenomics.py - metagenomic analyses

This script contains a number of utilities for metagenomic analyses.

```
usage: metagenomics.py subcommand
```

Sub-commands:

kraken

Classify reads by taxon using Kraken

```
usage: metagenomics.py kraken [-h] [--outReport OUTREPORT]
                              [--outReads OUTREADS]
                              [--filterThreshold FILTERTHRESHOLD]
```

```
                                    [--numThreads NUMTHREADS]
                                    [--loglevel {DEBUG,INFO,WARNING,ERROR,CRITICAL,EXCEPT
                                    [--version] [--tmp_dir TMP_DIR] [--tmp_dirKeep]
                                    inBam db
```

Positional arguments:

inBam	Input unaligned reads, BAM format.
db	Kraken database directory.

Options:

--outReport	Kraken report output file.
--outReads	Kraken per read output file.
--filterThreshold=0.05	Kraken filter threshold (default %(default)s)
--numThreads=1	Number of threads to run. (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
--tmp_dir=/tmp	Base directory for temp files. [default: %(default)s]
--tmp_dirKeep=False	Keep the tmp_dir if an exception occurs while running. Default is to delete all temp files at the end, even if there's a failure.

diamond

Classify reads by the taxon of the Lowest Common Ancestor (LCA)

```
usage: metagenomics.py diamond [-h] [--outM8 OUTM8] [--outLca OUTLCA]
                               [--numThreads NUMTHREADS]
                               [--loglevel {DEBUG,INFO,WARNING,ERROR,CRITICAL,EXCEP
                               [--version] [--tmp_dir TMP_DIR] [--tmp_dirKeep]
                               inBam db taxDb outReport
```

Positional arguments:

inBam	Input unaligned reads, BAM format.
db	Diamond database directory.
taxDb	Taxonomy database directory.
outReport	Output taxonomy report.

Options:

--outM8	Blast m8 formatted output file.
--outLca	Output LCA assignments for each read.
--numThreads=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

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

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

krona

Create an interactive HTML report from a tabular metagenomic report

```
usage: metagenomics.py krona [-h] [--queryColumn QUERYCOLUMN]
                             [--taxidColumn TAXIDCOLUMN]
                             [--scoreColumn SCORECOLUMN] [--noHits] [--noRank]
                             [--loglevel {DEBUG,INFO,WARNING,ERROR,CRITICAL,EXCEPTI
                             [--version]
                             inTsv db outHtml
```

Positional arguments:

inTsv Input tab delimited file.

db Krona taxonomy database directory.

outHtml Output html report.

Options:

--queryColumn=2 Column of query id. (default %(default)s)

--taxidColumn=3 Column of taxonomy id. (default %(default)s)

--scoreColumn Column of score. (default %(default)s)

--noHits=False Include wedge for no hits.

--noRank=False Include no rank assignments.

--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

report_merge

Merge multiple metegenomic reports into a single metagenomic report. Any Krona input files created by this

```
usage: metagenomics.py report_merge [-h]
                                    [--outSummaryReport OUT_KRAKEN_SUMMARY]
                                    [--krakenDB KRAKEN_DB]
                                    [--outByQueryToTaxonID OUT_KRONA_INPUT]
                                    [--loglevel {DEBUG,INFO,WARNING,ERROR,CRITICAL,
                                    [--version] [--tmp_dir TMP_DIR]
                                    [--tmp_dirKeep]
                                    metagenomic_reports
                                    [metagenomic_reports ...]
```

Positional arguments:

metagenomic_reports Input metagenomic reports with the query ID and taxon ID in the 2nd and 3rd columns (Kraken format)

Options:

--outSummaryReport Path of human-readable metagenomic summary report, created by kraken-report

--krakenDB Kraken database (needed for outSummaryReport)

--outByQueryToTaxonID Output metagenomic report suitable for Krona input.

--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

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

--tmp_dirKeep=False Keep the tmp_dir 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 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

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]
                                     [--threads THREADS]
                                     [--JVMmemory JVMMEMORY]
                                     [--loglevel {DEBUG,INFO,WARNING,ERROR,CRITICAL
                                     [--version] [--tmp_dir TMP_DIR]
                                     [--tmp_dirKeep]
                                     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: bmtaggerBam 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).

--threads=4 The number of threads to use in running blastn.

--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

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

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

trim_trimmomatic

Trim read sequences with Trimmomatic.

```
usage: taxon_filter.py trim_trimmomatic [-h]
                                        [--leadingQCutoff LEADING_Q_CUTOFF]
                                        [--trailingQCutoff TRAILING_Q_CUTOFF]
                                        [--minlengthToKeep MINLENGTH_TO_KEEP]
                                        [--slidingWindowSize SLIDING_WINDOW_SIZE]
                                        [--slidingWindowQCutoff SLIDING_WINDOW_Q_CU
                                        [--loglevel {DEBUG,INFO,WARNING,ERROR,CRITI
                                        [--version] [--tmp_dir TMP_DIR]
                                        [--tmp_dirKeep]
                                        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:

--leadingQCutoff=15 minimum quality required to keep a base on the leading end (default: %(default)s)

--trailingQCutoff=15 minimum quality required to keep a base on the trailing end (default: %(default)s)

--minlengthToKeep=30 minimum length of reads to be kept (default: %(default)s)

--slidingWindowSize=4 the number of bases to average across (default: %(default)s)

--slidingWindowQCutoff=25 specifies the average quality required in the sliding window (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

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

--tmp_dirKeep=False Keep the tmp_dir 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

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

```
usage: taxon_filter.py filter_lastal_bam [-h]
                                         [-n MAX_GAPLESS_ALIGNMENTS_PER_POSITION]
                                         [-l MIN_LENGTH_FOR_INITIAL_MATCHES]
                                         [-L MAX_LENGTH_FOR_INITIAL_MATCHES]
                                         [-m MAX_INITIAL_MATCHES_PER_POSITION]
                                         [--JVMmemory JVMMEMORY]
                                         [--loglevel {DEBUG,INFO,WARNING,ERROR,CRIT
                                         [--version] [--tmp_dir TMP_DIR]
                                         [--tmp_dirKeep]
                                         inBam db outBam
```

Positional arguments:

inBam Input reads

db Database of taxa we keep

outBam Output reads, filtered to refDb

Options:

-n=1 maximum gapless alignments per query position (default: %(default)s)

-l=5 minimum length for initial matches (default: %(default)s)

-L=50 maximum length for initial matches (default: %(default)s)

-m=100 maximum initial matches per query position (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

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

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

filter_lastal

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]
                                     [-n MAX_GAPLESS_ALIGNMENTS_PER_POSITION]
                                     [-l MIN_LENGTH_FOR_INITIAL_MATCHES]
                                     [-L MAX_LENGTH_FOR_INITIAL_MATCHES]
                                     [-m MAX_INITIAL_MATCHES_PER_POSITION]
                                     [--loglevel {DEBUG,INFO,WARNING,ERROR,CRITICAL
                                     [--version] [--tmp_dir TMP_DIR]
                                     [--tmp_dirKeep]
                                     inFastq refDb outFastq
```

Positional arguments:

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

Options:

-n=1	maximum gapless alignments per query position (default: %(default)s)
-l=5	minimum length for initial matches (default: %(default)s)
-L=50	maximum length for initial matches (default: %(default)s)
-m=100	maximum initial matches per query position (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
--tmp_dir=/tmp	Base directory for temp files. [default: %(default)s]
--tmp_dirKeep=False	Keep the tmp_dir if an exception occurs while running. Default is to delete all temp files at the end, even if there's a failure.

partition_bmtagger

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,CRI
                                          [--version] [--tmp_dir TMP_DIR]
                                          [--tmp_dirKeep]
                                          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 bmtool, 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

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

--tmp_dirKeep=False Keep the tmp_dir 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

Use bmtagger to deplete input reads against several databases.

```
usage: taxon_filter.py deplete_bam_bmtagger [-h] [--threads THREADS]
                                            [--JVMmemory JVMMEMORY]
                                            [--loglevel {DEBUG,INFO,WARNING,ERROR,C
                                            [--version] [--tmp_dir TMP_DIR]
                                            [--tmp_dirKeep]
                                            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 bmtool, and db.srprism.idx, db.srprism.map, etc. by srprism mkindex.

outBam Output BAM file.

Options:

--threads=4 The number of threads to use in running blastn.

--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

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

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

deplete_blastn

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

```
usage: taxon_filter.py deplete_blastn [-h] [--threads THREADS]
                                      [--loglevel {DEBUG,INFO,WARNING,ERROR,CRITICA
                                      [--version] [--tmp_dir TMP_DIR]
                                      [--tmp_dirKeep]
                                      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:

--threads=4	The number of threads to use in running blastn.
--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
--tmp_dir=/tmp	Base directory for temp files. [default: %(default)s]
--tmp_dirKeep=False	Keep the tmp_dir 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

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

```
usage: taxon_filter.py deplete_blastn_paired [-h] [--threads THREADS]
                                             [--loglevel {DEBUG,INFO,WARNING,ERROR,
                                             [--version] [--tmp_dir TMP_DIR]
                                             [--tmp_dirKeep]
                                             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:

--threads=4	The number of threads to use in running blastn.
--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
--tmp_dir=/tmp	Base directory for temp files. [default: %(default)s]

--tmp_dirKeep=False Keep the tmp_dir 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

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

```
usage: taxon_filter.py deplete_blastn_bam [-h] [--threads THREADS]
                                          [--chunkSize CHUNKSIZE]
                                          [--JVMmemory JVMMEMORY]
                                          [--loglevel {DEBUG,INFO,WARNING,ERROR,CRI
                                          [--version] [--tmp_dir TMP_DIR]
                                          [--tmp_dirKeep]
                                          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:

--threads=4 The number of threads to use in running blastn.

--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

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

--tmp_dirKeep=False Keep the tmp_dir 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

build a database for use with last based on an input fasta file

```
usage: taxon_filter.py lastal_build_db [-h]
                                       [--outputFilePrefix OUTPUTFILEPREFIX]
                                       [--loglevel {DEBUG,INFO,WARNING,ERROR,CRITIC
                                       [--version] [--tmp_dir TMP_DIR]
                                       [--tmp_dirKeep]
                                       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

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

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

blastn_build_db

Create a database for use with blastn from an input reference FASTA file

```
usage: taxon_filter.py blastn_build_db [-h]
                                       [--outputFilePrefix OUTPUTFILEPREFIX]
                                       [--loglevel {DEBUG,INFO,WARNING,ERROR,CRITIC
                                       [--version] [--tmp_dir TMP_DIR]
                                       [--tmp_dirKeep]
                                       inputFasta outputDirectory
```

Positional arguments:

inputFasta Location of the input FASTA file

outputDirectory Location for the output files

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

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

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

bmtagger_build_db

Create a database for use with Bmtagger from an input FASTA file.

```
usage: taxon_filter.py bmtagger_build_db [-h]
                                         [--outputFilePrefix OUTPUTFILEPREFIX]
                                         [--loglevel {DEBUG,INFO,WARNING,ERROR,CRIT
                                         [--version] [--tmp_dir TMP_DIR]
                                         [--tmp_dirKeep]
                                         inputFasta outputDirectory
```

Positional arguments:

inputFasta Location of the input FASTA file

outputDirectory Location for the output files (Where *.bitmask and *.srprism files will be stored)

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

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

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

1.3.3 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

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,CRITICA
                                      [--version] [--tmp_dir TMP_DIR]
                                      [--tmp_dirKeep]
                                      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

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

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

assemble_trinity

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] [--always_succeed]
                                    [--JVMmemory JVMMEMORY]
                                    [--threads THREADS]
                                    [--loglevel {DEBUG,INFO,WARNING,ERROR,CRITICAL,
                                    [--version] [--tmp_dir TMP_DIR]
                                    [--tmp_dirKeep]
                                    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
--always_succeed=False	If Trinity fails (usually because insufficient reads to assemble), emit an empty fasta file as output. Default is to throw a DenovoAssemblyError.
--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
--tmp_dir=/tmp	Base directory for temp files. [default: %(default)s]
--tmp_dirKeep=False	Keep the tmp_dir 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

This step cleans up the de novo assembly with a known reference genome. Uses MUMmer (nucmer or promer) to create a reference-based consensus sequence of aligned contigs (with runs of N's in between the de novo contigs).

```
usage: assembly.py order_and_orient [-h] [--breaklen BREAKLEN]
                                    [--maxgap MAXGAP] [--minmatch MINMATCH]
                                    [--mincluster MINCLUSTER]
                                    [--min_pct_id MIN_PCT_ID]
                                    [--min_contig_len MIN_CONTIG_LEN]
                                    [--min_pct_contig_aligned MIN_PCT_CONTIG_ALIGNE
                                    [--loglevel {DEBUG,INFO,WARNING,ERROR,CRITICAL,
                                    [--version] [--tmp_dir TMP_DIR]
                                    [--tmp_dirKeep]
                                    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:

--breaklen, -b Amount to extend alignment clusters by (if –extend). nucmer default 200, promer default 60.

--maxgap=200, -g=200 Maximum gap between two adjacent matches in a cluster. Our default is %(default)s. nucmer default 90, promer default 30. Manual suggests going to 1000.

--minmatch=10, -l=10 Minimum length of an maximal exact match. Our default is %(default)s. nucmer default 20, promer default 6.

--mincluster, -c Minimum cluster length. nucmer default 65, promer default 20.

--min_pct_id=0.6, -i=0.6 show-tiling: minimum percent identity for contig alignment (0.0 - 1.0, default: %(default)s)

--min_contig_len=200 show-tiling: reject contigs smaller than this (default: %(default)s)

--min_pct_contig_aligned=0.6, -v=0.6 show-tiling: minimum percent of contig length in alignment (0.0 - 1.0, 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

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

--tmp_dirKeep=False Keep the tmp_dir 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

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]
                                         [--aligner {muscle,mafft,mummer}]
                                         [--index]
                                         [--loglevel {DEBUG,INFO,WARNING,ERROR,CRIT
```

```
                                        [--version] [--tmp_dir TMP_DIR]
                                        [--tmp_dirKeep]
                                        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=0.5	minimum length for contig, as fraction of reference (default: %(default)s)
--minUnambig=0.5	minimum percentage unambiguous bases for contig (default: %(default)s)
--replaceLength=0	length of ends to be replaced with reference (default: %(default)s)
--aligner=muscle	which method to use to align inFasta to inReference. "muscle" = MUSCLE, "mafft" = MAFFT, "mummer" = nucmer. [default: %(default)s] Possible choices: muscle, mafft, mummer
--index=False	Index outFasta for Picard/GATK, Samtools, and Novoalign.
--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
--tmp_dir=/tmp	Base directory for temp files. [default: %(default)s]
--tmp_dirKeep=False	Keep the tmp_dir if an exception occurs while running. Default is to delete all temp files at the end, even if there's a failure.

refine_assembly

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,
                                   [--version] [--tmp_dir TMP_DIR]
                                   [--tmp_dirKeep]
```

```
                                    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
--tmp_dir=/tmp	Base directory for temp files. [default: %(default)s]
--tmp_dirKeep=False	Keep the tmp_dir 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

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,CRITICAI
                                     [--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

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] [--tmp_dir TMP_DIR] [--tmp_dirKeep]
                                 [--loglevel {DEBUG,INFO,WARNING,ERROR,CRITICAL,EXC
                                 [--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 & consistant with the ambiguous base (ie Y->C) (default: %(default)s)

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

--tmp_dirKeep=False Keep the tmp_dir 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

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

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,EXCEPT
                              [--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

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,EXC
                                 [--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

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

1.3.4 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

Annotate variants in VCF file with translation consequences using snpEff.

```
usage: interhost.py snpEff [-h] [--tmp_dir TMP_DIR] [--tmp_dirKeep]
                           [--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:

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

--tmp_dirKeep=False Keep the tmp_dir 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

Run the mafft alignment on the input FASTA file.

```
usage: interhost.py align_mafft [-h] [--localpair | --globalpair]
                                [--preservecase] [--reorder]
                                [--gapOpeningPenalty GAPOPENINGPENALTY]
                                [--ep EP] [--verbose] [--outputAsClustal]
                                [--maxiters MAXITERS] [--threads THREADS]
                                [--loglevel {DEBUG,INFO,WARNING,ERROR,CRITICAL,EXCE
                                [--version] [--tmp_dir TMP_DIR]
                                [--tmp_dirKeep]
```

```
                                    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: %(default)s). Note: if "–localpair" or "–globalpair" is specified this defaults to 1000.
--threads=-1	Number of processing threads (default: %(default)s, where -1 indicates use of all available cores).
--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
--tmp_dir=/tmp	Base directory for temp files. [default: %(default)s]
--tmp_dirKeep=False	Keep the tmp_dir if an exception occurs while running. Default is to delete all temp files at the end, even if there's a failure.

multichr_mafft

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 GAPOPENINGPENALTY]
                                   [--ep EP] [--verbose] [--outputAsClustal]
                                   [--maxiters MAXITERS] [--threads THREADS]
                                   [--outFilePrefix OUTFILEPREFIX]
                                   [--sampleRelationFile SAMPLERELATIONFILE]
                                   [--sampleNameListFile SAMPLENAMELISTFILE]
                                   [--loglevel {DEBUG,INFO,WARNING,ERROR,CRITICAL,E
                                   [--version] [--tmp_dir TMP_DIR]
                                   [--tmp_dirKeep]
                                   inFastas [inFastas ...] outDirectory
```

Positional arguments:

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

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: %(default)s). Note: if "–localpair" or "–globalpair" is specified this defaults to 1000.
--threads=-1	Number of processing threads (default: %(default)s, where -1 indicates use of all available cores).
--outFilePrefix=aligned	Prefix for the output file name (default: %(default)s)
--sampleRelationFile	If the parameter sampleRelationFile 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 sampleRelationFile 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: %(default)s] Possible choices: DEBUG, INFO, WARNING, ERROR, CRITICAL, EXCEPTION
--version, -V	show program's version number and exit
--tmp_dir=/tmp	Base directory for temp files. [default: %(default)s]
--tmp_dirKeep=False	Keep the tmp_dir if an exception occurs while running. Default is to delete all temp files at the end, even if there's a failure.

1.3.5 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

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,CRITIC
                                       [--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

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,EXCEPTIO
                            [--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

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,E
                                   [--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

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 interepreted 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

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

Convert VCF iSNV data to tabular text

```
usage: intrahost.py iSNV_table [-h]
                               [--loglevel {DEBUG,INFO,WARNING,ERROR,CRITICAL,EXCEP
                               [--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

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
                                     [--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

1.3.6 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

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,E
                                   [--version] [--tmp_dir TMP_DIR]
                                   [--tmp_dirKeep]
                                   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+)/[1|2]$ 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

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

--tmp_dirKeep=False Keep the tmp_dir 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

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,
                                    [--version] [--tmp_dir TMP_DIR]
                                    [--tmp_dirKeep]
                                    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

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

--tmp_dirKeep=False Keep the tmp_dir 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

Index a reference genome for Samtools.

```
usage: read_utils.py index_fasta_samtools [-h]
                                          [--loglevel {DEBUG,INFO,WARNING,ERROR,CRI
                                          [--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

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

```
usage: read_utils.py index_fasta_picard [-h] [--JVMmemory JVMMEMORY]
                                        [--picardOptions [PICARDOPTIONS [PICARDOPTI
                                        [--loglevel {DEBUG,INFO,WARNING,ERROR,CRITI
                                        [--version] [--tmp_dir TMP_DIR]
                                        [--tmp_dirKeep]
                                        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
--tmp_dir=/tmp	Base directory for temp files. [default: %(default)s]
--tmp_dirKeep=False	Keep the tmp_dir if an exception occurs while running. Default is to delete all temp files at the end, even if there's a failure.

mkdup_picard

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,EX
                                  [--version] [--tmp_dir TMP_DIR]
                                  [--tmp_dirKeep]
                                  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, 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

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

--tmp_dirKeep=False Keep the tmp_dir 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

Revert BAM to raw reads

```
usage: read_utils.py revert_bam_picard [-h] [--JVMmemory JVMMEMORY]
                                       [--picardOptions [PICARDOPTIONS [PICARDOPTIC
                                       [--loglevel {DEBUG,INFO,WARNING,ERROR,CRITIC
                                       [--version] [--tmp_dir TMP_DIR]
                                       [--tmp_dirKeep]
                                       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, 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

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

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

picard

Generic Picard runner.

```
usage: read_utils.py picard [-h] [--JVMmemory JVMMEMORY]
                            [--picardOptions [PICARDOPTIONS [PICARDOPTIONS ...]]]
                            [--loglevel {DEBUG,INFO,WARNING,ERROR,CRITICAL,EXCEPTIC
                            [--version] [--tmp_dir TMP_DIR] [--tmp_dirKeep]
                            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

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

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

sort_bam

Sort BAM file

```
usage: read_utils.py sort_bam [-h] [--index] [--md5] [--JVMmemory JVMMEMORY]
                              [--picardOptions [PICARDOPTIONS [PICARDOPTIONS ...]]]
                              [--loglevel {DEBUG,INFO,WARNING,ERROR,CRITICAL,EXCEPT
                              [--version] [--tmp_dir TMP_DIR] [--tmp_dirKeep]
                              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

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

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

merge_bams

Merge multiple BAMs into one

```
usage: read_utils.py merge_bams [-h] [--JVMmemory JVMMEMORY]
                                [--picardOptions [PICARDOPTIONS [PICARDOPTIONS ...]
                                [--loglevel {DEBUG,INFO,WARNING,ERROR,CRITICAL,EXCE
                                [--version] [--tmp_dir TMP_DIR]
                                [--tmp_dirKeep]
                                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, 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

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

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

filter_bam

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,EXCE
                                [--version] [--tmp_dir TMP_DIR]
                                [--tmp_dirKeep]
                                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, 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

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

--tmp_dirKeep=False Keep the tmp_dir 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

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,EX
                                    [--version] [--tmp_dir TMP_DIR]
                                    [--tmp_dirKeep]
                                    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
--tmp_dir=/tmp	Base directory for temp files. [default: %(default)s]
--tmp_dirKeep=False	Keep the tmp_dir 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

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,EX
                                  [--version] [--tmp_dir TMP_DIR]
                                  [--tmp_dirKeep]
                                  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

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

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

split_reads

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

Split BAM file equally into several output BAM files.

```
usage: read_utils.py split_bam [-h]
                               [--loglevel {DEBUG,INFO,WARNING,ERROR,CRITICAL,EXCEP
                               [--version] [--tmp_dir TMP_DIR] [--tmp_dirKeep]
                               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

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

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

reheader_bam

Copy a BAM file (inBam to outBam) while renaming elements of the BAM header. The mapping file specifies which (key, old value, new value) mappings. For example: LB lib1 lib_one SM sample1 Sample_1 SM sample2 Sample_2 SM sample3 Sample_3 CN broad BI

```
usage: read_utils.py reheader_bam [-h]
                                  [--loglevel {DEBUG,INFO,WARNING,ERROR,CRITICAL,EX
                                  [--version] [--tmp_dir TMP_DIR]
                                  [--tmp_dirKeep]
                                  inBam rgMap outBam
```

Positional arguments:

inBam Input reads, BAM format.

rgMap Tabular file containing three columns: field, old, new.

outBam Output reads, 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

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

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

reheader_bams

Copy BAM files while renaming elements of the BAM header. The mapping file specifies which (key, old value, new value) mappings. For example: LB lib1 lib_one SM sample1 Sample_1 SM sample2 Sample_2 SM sample3 Sample_3 CN broad BI FN in1.bam out1.bam FN in2.bam out2.bam

```
usage: read_utils.py reheader_bams [-h]
                                   [--loglevel {DEBUG,INFO,WARNING,ERROR,CRITICAL,E
                                   [--version] [--tmp_dir TMP_DIR]
                                   [--tmp_dirKeep]
                                   rgMap
```

Positional arguments:

rgMap Tabular file containing three columns: field, old, new.

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

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

--tmp_dirKeep=False Keep the tmp_dir 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

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,CRITIC
                                       [--version] [--tmp_dir TMP_DIR]
                                       [--tmp_dirKeep]
                                       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

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

--tmp_dirKeep=False Keep the tmp_dir 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

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,CRITI
                                        [--version] [--tmp_dir TMP_DIR]
                                        [--tmp_dirKeep]
                                        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

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

--tmp_dirKeep=False Keep the tmp_dir 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

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,CRIT
                                         [--version] [--tmp_dir TMP_DIR]
                                         [--tmp_dirKeep]
                                         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

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

--tmp_dirKeep=False Keep the tmp_dir 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

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,C
                                            [--version] [--tmp_dir TMP_DIR]
                                            [--tmp_dirKeep]
                                            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

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

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

novoalign

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,EXCEP
                               [--version] [--tmp_dir TMP_DIR] [--tmp_dirKeep]
                               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

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

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

novoindex

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,EXCEP
                               [--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

Call genotypes using the GATK UnifiedGenotyper.

```
usage: read_utils.py gatk_ug [-h] [--options OPTIONS] [--JVMmemory JVMMEMORY]
                             [--loglevel {DEBUG,INFO,WARNING,ERROR,CRITICAL,EXCEPTI
                             [--version] [--tmp_dir TMP_DIR] [--tmp_dirKeep]
                             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

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

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

gatk_realign

Local realignment of BAM files with GATK IndelRealigner.

```
usage: read_utils.py gatk_realign [-h] [--JVMmemory JVMMEMORY]
                                  [--loglevel {DEBUG,INFO,WARNING,ERROR,CRITICAL,EX
                                  [--version] [--tmp_dir TMP_DIR]
                                  [--tmp_dirKeep] [--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

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

--tmp_dirKeep=False Keep the tmp_dir 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

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,E
                                   [--version] [--tmp_dir TMP_DIR]
                                   [--tmp_dirKeep]
                                   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

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

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

align_and_count_hits

Take reads, align to reference with Novoalign and return aligned read counts for each reference sequence.

```
usage: read_utils.py align_and_count_hits [-h] [--includeZeros]
                                          [--JVMmemory JVMMEMORY]
                                          [--loglevel {DEBUG,INFO,WARNING,ERROR,CRI
                                          [--version] [--tmp_dir TMP_DIR]
                                          [--tmp_dirKeep]
                                          inBam refFasta outCounts
```

Positional arguments:

inBam	Input unaligned reads, BAM format.
refFasta	Reference genome, FASTA format, pre-indexed by Picard and Novoalign.
outCounts	Output counts file

Options:

--includeZeros=False	Output lines with no hits (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
--tmp_dir=/tmp	Base directory for temp files. [default: %(default)s]
--tmp_dirKeep=False	Keep the tmp_dir if an exception occurs while running. Default is to delete all temp files at the end, even if there's a failure.

1.3.7 reports.py - produce various metrics and reports

Functions to create reports from genomics pipeline data.

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

Sub-commands:

assembly_stats

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: %(default)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. (default: %(default)s)

--raw_reads_dir=data/00_raw Directory with unaligned raw read BAMs. (default: %(default)s)

alignment_summary

Write or print pairwise alignment summary information for sequences in two FASTA files, including SNPs, ambiguous bases, and indels.

```
usage: reports.py alignment_summary [-h] [--outfileName OUTFILENAME]
                                    [--printCounts]
                                    inFastaFileOne inFastaFileTwo
```

Positional arguments:

inFastaFileOne First fasta file for an alignment

inFastaFileTwo First fasta file for an alignment

Options:

--outfileName Output file for counts in TSV format

--printCounts=False Undocumented

consolidate_fastqc

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

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.

1.3.8 illumina.py - for raw Illumina outputs

Utilities for demultiplexing Illumina data.

```
usage: illumina.py subcommand
```

Sub-commands:

illumina_demux

Demultiplex Illumina runs & produce BAM files, one per sample. Wraps together Picard's ExtractBarcodes and IlluminaBasecallsToSam while handling the various required input formats. Also can read Illumina BCL directories, tar.gz BCL directories. TO DO: read BCL or tar.gz BCL directories from S3 / object store.

```
usage: illumina.py illumina_demux [-h] [--outMetrics OUTMETRICS]
                                  [--commonBarcodes COMMONBARCODES]
                                  [--sampleSheet SAMPLESHEET]
                                  [--flowcell FLOWCELL]
                                  [--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]
                                  [--sequencing_center SEQUENCING_CENTER]
                                  [--adapters_to_check [ADAPTERS_TO_CHECK [ADAPTERS
                                  [--platform PLATFORM]
                                  [--max_reads_in_ram_per_tile MAX_READS_IN_RAM_PER
                                  [--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_READS]
                                  [--run_start_date RUN_START_DATE]
                                  [--read_group_id READ_GROUP_ID]
                                  [--JVMmemory JVMMEMORY]
                                  [--loglevel {DEBUG,INFO,WARNING,ERROR,CRITICAL,EX
                                  [--version] [--tmp_dir TMP_DIR]
                                  [--tmp_dirKeep]
                                  inDir lane outDir
```

Positional arguments:

inDir	Illumina BCL directory (or tar.gz of BCL directory). This is the top-level run directory.
lane	Lane number.
outDir	Output directory for BAM files.

Options:

--outMetrics	Output ExtractIlluminaBarcodes metrics file. Default is to dump to a temp file.
--commonBarcodes	Write a TSV report of all barcode counts, in descending order.

--sampleSheet Override SampleSheet. Input tab or CSV file w/header and four named columns: barcode_name, library_name, barcode_sequence_1, barcode_sequence_2. Default is to look for a SampleSheet.csv in the inDir.

--flowcell Override flowcell ID (default: read from RunInfo.xml).

--read_structure Override read structure (default: read from RunInfo.xml).

--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)

--sequencing_center Picard IlluminaBasecallsToSam SEQUENCING_CENTER (default: %(default)s)

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

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

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

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

--num_processors=4 Picard IlluminaBasecallsToSam NUM_PROCESSORS (default: %(default)s)

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

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

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

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

--include_non_pf_reads=False Picard IlluminaBasecallsToSam INCLUDE_NON_PF_READS (default: %(default)s)

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

--read_group_id Picard IlluminaBasecallsToSam 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

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

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

common_barcodes

Extract Illumina barcodes for a run and write a TSV report of the barcode counts in descending order

```
usage: illumina.py common_barcodes [-h] [--truncateToLength TRUNCATETOLENGTH]
                                   [--omitHeader] [--includeNoise]
                                   [--outMetrics OUTMETRICS]
                                   [--sampleSheet SAMPLESHEET]
                                   [--flowcell FLOWCELL]
                                   [--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]
                                   [--JVMmemory JVMMEMORY]
                                   [--loglevel {DEBUG,INFO,WARNING,ERROR,CRITICAL,E
                                   [--version] [--tmp_dir TMP_DIR]
                                   [--tmp_dirKeep]
                                   inDir lane outSummary
```

Positional arguments:

inDir Illumina BCL directory (or tar.gz of BCL directory). This is the top-level run directory.

lane Lane number.

outSummary Path to the summary file (.tsv format). It includes two columns: (barcode, count)

Options:

--truncateToLength If specified, only this number of barcodes will be returned. Useful if you only want the top N barcodes.

--omitHeader=False If specified, a header will not be added to the outSummary tsv file.

--includeNoise=False If specified, barcodes with periods (".") will be included.

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

--sampleSheet Override SampleSheet. Input tab or CSV file w/header and four named columns: barcode_name, library_name, barcode_sequence_1, barcode_sequence_2. Default is to look for a SampleSheet.csv in the inDir.

--flowcell Override flowcell ID (default: read from RunInfo.xml).

--read_structure Override read structure (default: read from RunInfo.xml).

--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)

--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

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

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

miseq_fastq_to_bam

Convert fastq read files to a single bam file. Fastq file names must conform to patterns emitted by Miseq machines. Sample metadata must be provided in a SampleSheet.csv that corresponds to the fastq filename. Specifically, the _S##_ index in the fastq file name will be used to find the corresponding row in the SampleSheet

```
usage: illumina.py miseq_fastq_to_bam [-h] [--inFastq2 INFASTQ2]
                                      [--runInfo RUNINFO]
                                      [--sequencing_center SEQUENCING_CENTER]
                                      [--JVMmemory JVMMEMORY]
                                      [--loglevel {DEBUG,INFO,WARNING,ERROR,CRITICA
                                      [--version] [--tmp_dir TMP_DIR]
                                      [--tmp_dirKeep]
                                      outBam sampleSheet inFastq1
```

Positional arguments:

outBam Output BAM file.

sampleSheet Input SampleSheet.csv file.

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

Options:

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

--runInfo Input RunInfo.xml file.

--sequencing_center Name of your sequencing center (default is the sequencing machine ID from the RunInfo.xml)

--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

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

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

extract_fc_metadata

Extract RunInfo.xml and SampleSheet.csv from the provided Illumina directory

```
usage: illumina.py extract_fc_metadata [-h]
                                       [--loglevel {DEBUG,INFO,WARNING,ERROR,CRITIC
                                       [--version] [--tmp_dir TMP_DIR]
                                       [--tmp_dirKeep]
                                       flowcell outRunInfo outSampleSheet
```

Positional arguments:

flowcell Illumina directory (possibly tarball)

outRunInfo Output RunInfo.xml file.

outSampleSheet Output SampleSheet.csv 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

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

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

1.3.9 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

Find the basecalls directory from a Picard directory

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

Find the sequencing run date from a Picard directory

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

Get all samples

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

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/snakemake/snakemake/wiki/Home

Here is an overview of the Snakemake rule graph:


download_lastal_sources
build_lastal_db
depletion
filter_to_taxon
spikein_report
merge_one_per_sample
assemble_trinity
download_reference_genome
orient_and_impute
consolidate_spike_count
refine_assembly_1
refine_assembly_2
map_reads_to_self
multi_align_mafft
fastqc_report
isnvs_per_sample
consolidate_fastqc
isnvs_vcf
all

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 pipline 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 pipline is intended to be run on an input one or more sequencer bam files, each having a filename represending 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.yaml
Snakefile
```

Since the file `config.yaml` is project-specific, you will need to make changes to it as approprate 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.yaml`.

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 int**er**host analysis pipeline.
- Create a blank file, `samples-assembly-failures.txt`, that may be filled in later.

Modifying the `config.yaml` 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 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.

```
bmtagger_db_dir: "/path/to/depletion_databases"
bmtagger_dbs_remove:
  - "hg19"
  - "GRCh37.68_ncRNA-GRCh37.68_transcripts-HS_rRNA_mitRNA"
  - "metagenomics_contaminants_v3"
```

Pre-built depletion databases are available in both *.tar.gz and *.lz4 format, for removing human reads and common metagenomic contaminants:

- GRCh37.68_ncRNA-GRCh37.68_transcripts-HS_rRNA_mitRNA.tar.gz (*.lz4)
- hg19.tar.gz (*.lz4)

- metagenomics_contaminants_v3.tar.gz (*.lz4)

Note that these databases must be extracted prior to use.

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.

```
blast_db_dir: "/path/to/depletion_databases"
blast_db_remove: "metag_v3.ncRNA.mRNA.mitRNA.consensus"
```

A pre-built depletion database is also available for BLAST:

- metag_v3.ncRNA.mRNA.mitRNA.consensus.tar.gz (*.lz4)

Note that this database must be extracted prior to use.

Additional databases are needed to perform metagenomic classification using Kraken, Diamond, or Krona.

```
kraken_db: "/path/to/kraken_full_20150910"

diamond_db: "/path/to/diamond_db/nr"

krona_db: "/path/to/krona"
```

Pre-built databases for Kraken, Diamond, and Krona are available:

- kraken_db.tar.gz (*.lz4)
- krona_taxonomy_20160502.tar.gz (*.lz4)
- nr.dmnd.gz (*.lz4)

Note that these databases must be extracted prior to use.

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 int**er**host and int**ra**host 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"
```

An optional file containing a list of accessions may be specified for filtering reads via LAST. This is intended to narrow to a genus. If this file is not provided, viral-ngs defaults to using the accessions specified for the reference genome.

```
accessions_file_for_lastal_db_build: "/path/to/lastal_accessions.txt"
```

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

```
trim_clip_db: "/path/to/depletion_databases/contaminants.fasta"
```

Pre-built databases for Trimmomatic:

- contaminants.fasta.tar.gz (*.lz4)

A FASTA file containing spike-ins to be reported:

```
spikeins_db: "/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.yaml`, 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 -b y -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

1.5 Developing viral-ngs

1.5.1 Testing with tox and pyenv

Using pyenv with tox can simplify testing locally against multiple different Python versions and dependency environments. To setup your development environment for testing, you first need to perform the following steps:

1. Install pyenv for your OS https://github.com/yyuu/pyenv#installation
2. Install the appropriate Python versions:

```
$ pyenv install 2.7.10 3.4.3 3.5.0
```

3. Change directory to the *viral-ngs* git checkout.
4. Set the local pyenv versions:

```
$ pyenv local 3.5.0 3.4.3 2.7.10
```

5. Install tox and tox-pyenv

```
$ pip3 install tox tox-pyenv
```

6. Run tox to run tests and check building docs.

```
$ tox
```

1.5.2 Testing easy_deploy with Vagrant and Ansible

By default, easy_deploy installs a version of viral-ngs by cloning the github repository, which makes it difficult to test local changes to the playbook. Testing the easy_deploy setup locally can be done using Vagrant and the Ansible playbook with some custom commands. To do so, we need to take the following steps:

1. Install Vagrant
2. Install Ansible
3. Change to the easy_deploy directory

```
$ cd easy_deploy
```

4. Run the easy deploy script to bootstrap the VM and answer the prompts

```
$ ./run.sh
```

5. From now on, run Ansible playbook manually for provisioning: http://docs.ansible.com/ansible/guide_vagrant.html#running-ansible-manually.
6. Add *deploy=sync* or *deploy=archive* to the *–extra-vars* of the Ansible playbook command

```
$ ansible-playbook ... --extra-vars=deploy=sync
```

To test playbook changes in a local repository, the choice of *deploy=sync* or *deploy=archive* changes the strategy used to "deploy" viral-ngs into the Vagrant VM.

The *deploy=sync* option creates a symlink to the synced folder containing the root of your viral-ngs git repo. Therefore, any tool installation or other changes will be reflected on both the host and the guest machine. This is desirable for fast iteration of changes, but makes it difficult to isolate the host's viral-ngs installation from the installation on the guest VM.

The *deploy=archive* option performs a *git archive* on the host's viral-ngs repository and untars it into the project directory. This is a clean install each time, which can be time consuming due to the need to reinstall all dependencies, and only the current HEAD commit will be reflected in the guest. No uncommitted/dirty changes will be picked up using this method. This is more ideally suited for a finalized clean test of the playbook.