Name

MOCAT - Metagenomics Analysis Toolkit

Description

MOCAT, is a software package to process Illumina metagenomic data. This software provides a pipeline for processing raw reads, assembly, gene prediction, extracting marker genes and mapping reads to external databases

Synopsis

```
> MOCAT.pl [Required Options] [Pipeline Options] [Statistics Options] [Additional Options]
```

Or, try the wrapper script, which runs multiple MOCAT commands at once:

```
> runMOCAT.sh
```

MOCAT modules

Look in the MOCAT/mod folder for examples how to create modules. Run MOCAT.pl without options for available modules and their options.

Example - Process, Assemble, Revise Assembly, Predict Genes, cluster genes into gene catalog, annotate gene catalog, profile against gene catalog

```
MOCAT.pl -sf my.samples -rtf
MOCAT.pl -sf my.samples -a
MOCAT.pl -sf my.samples -gp assembly
MOCAT.pl -sf my.samples -make_gene_catalog -assembly_type assembly
MOCAT.pl -sf my.samples -annotate_gene_catalog
MOCAT.pl -sf my.samples -s my.samples.padded -identity 95
MOCAT.pl -sf my.samples -f my.samples.padded -identity 95
MOCAT.pl -sf my.samples -p my.samples.padded -identity 95 -mode functional
```

Example - Process, Assemble, Predict Genes and fetch marker genes

```
MOCAT.pl -sf my.samples -rtf
MOCAT.pl -sf my.samples -a
MOCAT.pl -sf my.samples -gp assembly
MOCAT.pl -sf my.samples -fmg assembly
MOCAT.pl -sf my.samples -ss
```

Example - Process, Screen against DB, Assemble, and Predict Genes

```
MOCAT.pl -sf my.samples -rtf
MOCAT.pl -sf my.samples -s hg19 -screened_files
MOCAT.pl -sf my.samples -a -r hg19
MOCAT.pl -sf my.samples -gp assembly -r hg19
MOCAT.pl -sf my.samples -ss
```

Example - Process, screen against fasta file, against DB, Assemble, and Predict Genes

```
MOCAT.pl -sf my.samples -rtf
MOCAT.pl -sf my.samples -sff adapters.fa
MOCAT.pl -sf my.samples -s hg19 -r adapters.fa -screened_files
MOCAT.pl -sf my.samples -a -r screened.adapters.fa.on.hg19
MOCAT.pl -sf my.samples -gp assembly -r screened.adapters.fa.on.hg19
MOCAT.pl -sf my.samples -ss
```

Example - Calculate coverage of reads mapping a custom made database

MOCAT.pl -sf my.samples -rtf

MOCAT.pl -sf my.samples -s /User/My/Path/REF_DB.fna

MOCAT.pl -sf my.samples -f REF_DB.fna

MOCAT.pl -sf my.samples -p REF_DB.fna

MOCAT.pl -sf my.samples -ss

Example Sample File

MH0001

GOS_STATION_2

MMMS_0561

FAQ

How do I calculate the read coverage of each contigs, scaffold or scaftig of an assembly?

MOCAT can calculate the total read and base coverages (length normalized and on normalized) for each contig, scaftig or scaffold. To do this, first run MOCAT.pl -sf sample -a to assemble your samples. Then execute MOCAT.pl -sf sample -s [s|c|f] to map reads to the assembled scaftigs, contigs or scaffolds. After this run MOCAT.pl -sf sample -f [s|c|f] to filter the mapped reads are selected percentage and length cutoff. Finally run MOCAT.pl -sf sample -p [s|c|f] to calculate the coverages.

Options

Required Options

`-sf|sample_file 'FILE'` (required)

'FILE' contains the list of folder names (sample names), one per line, in which the raw sample data is located. See section 'Setup MOCAT in a new folder' for more details.

Pipeline Options

`-rtf|read_trim_filter`

Performs trimming and quality filtering of raw reads and stores them in SAMPLE/reads.proocessed/*[pair|single]*.fq.gz.

For supported file formats (old and new Illumina format), see section Supported Formats below.

NOTE! Both 'fastx' and 'solexaqa' requires the two LANE.1.fq and LANE.2.fq input files to contain reads in the same order (the files may have different number of reads, but their respective order must be the same). This is usually the case when the FastQ file comes directly from the sequencing machine, however if a pre-filtering step has been performed, the order may have changed.

Additional note: If 'readtrimfilter_use_precalc_5prime_trimming' in the config file is set to 'yes', then the file MOCAT.cutoff5prime must exist. For more information, see Config File Section.

`-s|screen ['DB1 DB2 ...' or 's' or 'c' or 'f']`

Additionally required options

`-r|reads ['reads.processed', 'DATABASE NAME']` (required)

This is to specify whether reads should be extracted from reads that have been a) trimmed and filtered, or b) trimmed, filtered and screened against a custom databse. See below.

Additional options

`-e|extract` (optional)

Set this flag if you want to screen the extracted reads (reads that matched the database), rather than the screened (reads tat did not match) from a database.

`-screened_files` (optional)

If set, reads are mapped and the files in the .screened folder are produced, but not the .extracted folder.

`-extracted_files` (optional)

If set, reads are mapped and the files in the .extracted folder are produced, but not the .screened folder.

`-use_mem` (optional)

This option can be set when running the -s option. If set, it copies the selected database into memory and then sets the data folder to the memory location. This will speed up loading the database for each sample, because it\s read from memory instead of disk.

Trimmed and filtered reads will be aligned against a custom database (or multiple databases) located in the MOCAT data folder, using SOAPAligner2. This databse has to be previously generated and the name to specify is the filenames of the database, excluding '.index.xxx'. Provided with MOCAT is the 'hg19' database, which can for example be used to screen for human contamination in procaryotic samples. Note, if a paired end read is filtered, then it's pair mate is also removed. For a detailed description of the database files required and created, see the taxonomic profiling section below. Note that if you wish to use multiple databases (a split database) the naming convention must follow DBNAME.1 DBNAME.2 etc.

`-sff|screen_fastafile 'FASTA FILE'`

Additionally required options

`-r|reads ['reads.processed', 'DATABASE NAME']` (required)

This is to specify whether reads should be extracted from reads that have been a) trimmed and filtered, or b) trimmed, filtered and screened against a custom databse. See below.

Reads that have been trimmed and filtered (and possibly screened against a custom database) can be screened against a fasta file with DNA sequences. This screening step is a BLAST search using USearch.

FASTA FILE is an input file with sequences, against which, the reads will be blasted. a) Trimmed and filter reads OR b) reads that have been both trimmed & filter and additionally screened using a database, can be screened against a fasta file. To screen reads that have been only trimmed and filtered specify 'read_trim_filter'. To screen reads from previsouly screened reads (using -db), specify the database name.

IMPORTANT NOTE: To maximize accuracy, RAW reads are blasted against the specified fasta file. Then, reads that have a blast mastch are removed from either reads that have been a) trimmed and filter only, or b) trimmed and filterd and additionally screened using a database.

`-a|assembly`

Additionally required options

`-r|reads ['reads.processed', 'DATABASE NAME']` (required)

This is to specify whether reads should be extracted from reads that have been a) trimmed and filtered, or b) trimmed, filtered and screened against a custom databse. See below.

The option specifies from which type of reads the assembly should be made. a) 'reads.processed' refers to reads that were only trimmed and filtered. b) 'DATABASE NAME' are reads that were additionally screened using a database. c) 'FASTA FILE' are reads that were screened against a fasta file.

Reads can be assembled using SOAPDenovo. This step assembles trimmed and filtered reads (-ar|assembly_reads 'reads.processed') or reads that were previously screened using a database and/or fasta file (-ar|assembly_reads 'DATABASE NAME' or -ar|assembly_reads 'FASTA FILE') into scaftigs and contigs. The assembly is done in two steps: 1. calculate insert sizes, 2. assembly. Calculating insert sizes can be done by aligning a subset of reads to a database (we have provided '1506MG'), or by pre-assembly. Mapping reads to a database may be more accurate and faster. The choice of aligning reads or assembly, is specified in the MOCAT config file using the tag 'assembly_calculate_insert_size_using' set to 'mapping' or 'assembly'.

Additional note: If 'assembly_calculate_insert_size_using' in the config file is set to 'assembly', the file 'MOCAT.preliminary_insert_sizes' must exist. For more information, see Config File Section.

`-ar|assembly_revision`

Additionally required options

`-r|reads ['reads.processed', 'DATABASE NAME']` (required)

This is to specify whether reads should be extracted from reads that have been a) trimmed and filtered, or b) trimmed, filtered and screened against a custom databse. See below.

The option specifies from which type of reads the assembly was made. a) 'reads.processed' refers to reads that were only trimmed and filtered. b) 'DATABASE NAME' are reads that were additionally screened using a database. c) 'FASTA FILE' are reads that were screened against a fasta file.

Corrects assemblies for base pair and short indel errors. First, reads are aligned to the scaftigs using BWA to detect single base and short indel errors. For regions with low coverage (<10x): a) if there is a majority of one base (>0.7), the base of the scaftig, in that position, is modified accordingly. b) Scaftigs have short indels either inserted or deleted, if the read support is >0.5. Then, reads are aligned to the 'new' scaftigs using SOAPAligner2. The assembly revision can be made on assemblies that were made using reads that were a) trimmed and filtered ('reads.processed'), or b) trimmed, filtered and screened against a custom databse ('DATABASE NAME'), or c) trimmed, filtered, screened against a custom database and against a fasta file ('FASTA FILE')

`-gp|gene_prediction ['assembly' or 'assembly.revised']`

Additionally required options

`-r|reads ['reads.processed', 'DATABASE NAME' or 'FASTA FILE']` (required)

Specifies from which type of reads the assembly was made. Reads that were only trimmed and filtered, or also additionally screened using either a database and/or a fasta file.

Main option specifies whether to use a non revised, or revised assembly.

Predicts genes (ORF and protein sequences) from an assembly using MetaGeneMark or Prodigal (specified in config file).

`-fmg|fetch_mg ['assembly' or 'assembly.revised']`

Additionally required options

`-r|reads ['reads.processed', 'DATABASE NAME' or 'FASTA FILE']` (required)

Specifies from which type of reads the assembly was made. Reads that were only trimmed and filtered, or also additionally screened using either a database and/or a fasta file.

Main option specifies whether to use a non revised, or revised assembly.

Fetches (extracts) 40 universial marker genes (Ciccarelli et al., Science, 2006 and Sorek et al., Science, 2007) from the set of predicted genes from each sample. In brief, Pre-built HMM models are used to identify protein coding sequencing matching these 40 universal marker genes. It is possible to train this model to retrieve a different set of genes, for instructions how to do this, execute 'MOCATFetchMGs03.pl' found in the MOCAT bin directory. To use these newly configured bit score cutoffs to retrieve sequences, replace the generated 'MG_BitScoreCutoffs.txt', with the original one in the MOCAT lib/fetchMG folder.

`-f|filter ['DB1 DB2 ...','s','c','f','r']`

Additionally required options

`-r|reads ['reads.processed', 'DATABASE NAME' or 'FASTA FILE']` (required)

Specifies from which type of reads the assembly was made. Reads that were only trimmed and filtered, or also additionally screened using either a database and/or a fasta file.

Additional options

`-e|extract`

Set this flag if, in the screen step, this flag was set.

`-shm`

If set, faster, but saves temporary data to /dev/shm/<USER> instead of TEMP directory

Additional filtering of reads that have been mapped using the s|screen command. Percentage ID and length cut off are defined in the config file. With this option you can map reads at a lower cutoff and then filter them at different (higher) cutoffs. It is required to run this step before running profiling below, even though you do not want to filter at a higher cut off. If you do not want to filter at a higher cut off, set the values in the config file to the same ones as for the screen step.

`-p|profiling ['DB1 DB2 ...','s','c','f','r']`

Additionally required options

`-r|reads ['reads.processed', 'DATABASE NAME' or 'FASTA FILE']` (required)

Specifies from which type of reads the assembly was made. Reads that were only trimmed and filtered, or also additionally screened using either a database and/or a fasta file.

`-m|mode ['gene', 'NCBI', 'mOTU', 'functional']`

Basic mode (gene): The (gene) coverages of the database are calculated. Of course, the sequences in the database does not have to be genes, but as MOCAT was designed to be used with gene catalogs, we call this mode 'gene'. :)

Taxonomic modes (NCBI, mOTU): Specify whether the taxonomic profiles to be calculated are based on reference marker genes or reference genomes (NCBI), or metagenomic OTUs (mOTU). These two settings both have specific requirements, both on the database file and also additionally requireed files.

Functional: If this is specified, and an additional functional.map file exists for the database, the gene abundances are first calculated and then summaried at the different funcitonal levels. By default the levels are cog, ko, module and pathway, but this could be modified for a custom database.

`-o|output [OUTPUT FOLDER]`

Easy to use links to the generated abundances tables are saved in the OUTPUT folder. These files are the main output of the taxonomic and mOTU profiling steps.

Additional options

`-e|extract`

Set this flag if, in the screen step, this flag was set.

`-no_horizontal`

Do not calculate horizontal gene & functional coverages

`-verbose`

Prints additional information about profiling status

`-uniq`

Specify this flag if you find duplicated row names (e.g. if you have mapped to a DB where the same reference appears multiple times)

`-shm`

If set, faster, but saves profiling temporary data (2-5 GB per sample) to /dev/shm/<USER> instead of TEMP directory

If mode is set to 'gene', then calculates the base and insert counts (and normalized by reference ID length) for reads that have been mapped and filtered against a database using first the screen and then filter commands.

Specify 'NCBI' to generates taxonomic profiles by summarizing base and insert coverages into abundances of taxa (kingdom up to species), or specify 'mOTU' to summarize into abundances of mOTUs. The two different modes can not be used on the same type of database. The database need to be constructed in a specific way, in order to run this step. MOCAT ships with to different databases, one for each mode. 'RefMG.v1.padded' is used when you wish to calculate taxonomic profiles with NCBI taxa names, and the 'mOTU.v1.padded' is used for generating mOTU species cluster abundances. If you want to use the provided databases, it is enough to run the screen and filter steps and then this step on either the database RefMG.v1.padded or mOTU.v1.padded, and specifying mode to 'NCBI' and 'mOTU', respectively, when running the this step.

By specifying mode to be 'functional' is it possible to summarize gene abundances into functional categorizes. Some publicly available database have been annotated to various functional categories. By default MOCAT is expected to be used to summarize into cog, ko, module and pathway abundances. Note that the mapping file DBNAME.functional.map must either be downloaded for the database, or manually generated.

NCBI mode: Design and requirements for summarizing gene profiles into kingdom up to species abundances

The names within () are example names and the convention of the file names need to be used. This means if the database is called 'DB', the required files are called

'DB.xxxxx'.

The database file (RefMG.v1.padded)

The fasta headers MUST be on the format '>taxaid.XXXX'. This format is required for the taxonomic profiling step to identify from which taxa the sequence is.

The database NCBI map file (RefMG.v1.padded.NCBI.map)

This file must be generated manually. However, MOCAT ships with a version of this file 'RefMG.v1.padded.NCBI.map'. This file can be copied and renamed if you have designed a different databse using the NCBI taxa ids, as described above for the database file. It is a mapping file from taxa ids to different taxonomic levels, and has the format '<TaxID>\t<Kingdom>\t<Phylum>\t<Class>\t<Order>\t<Family>\t<Genus>\t<Species>\t<specI_clusters>\t<length of taxa ID>'

The length field contains the summarized length of all the sequences corresponding to each taxa id in the database.

CuratedSpecies represents the species clusters as described in Mende et al (2013).

The database file .coord file (RefMG.v1.padded.coord)

This file contains information of which bases should be counted when calculating base and insert coverages. Normally this file is generated automatically and contains the fasta entry names with start and stop bases, normally start=1 and stop=length of that sequence. Changing this file may be useful if you want to calculate the coverages based on only a part of a gene or genome.

Example line: 1001582.LL3_00013<TAB>101<TAB>1378

The database file .len file (RefMG.v1.padded.len)

This file contains the fasta entries and their respective lengths, one per line. It is generated automatically and do not need to be changed.

Example line: 1001582.LL3_00013<TAB>1478

The database file .rownames and .rownames.uniq files (RefMG.v1.padded.rownames[.uniq])

These files contains the rownames of the computed base and insert coverages. These files are generated automatically and do not need to be modified. I the .uniq file the _X_X corresponds to the region over which the coverages are calculated. This is normally _1_[length of sequence]. This region can be changed in the .coord file.

mOTU mode: Design and requirements for summarizing gene profiles into species cluster (mOTU) abundances

The names within () are example names and the convention of the file names need to be used. This means if the database is called 'DB', the required files are called 'DB.xxxxx'.

The database file (mOTU.v1.padded) (same as for NCBI mode)

The fasta headers must be defined in the map file. As long as the have been defined in this file, they can have any name.

The database NCBI map file (mOTU.v1.padded.motu.map) (NOT same as for NCBI mode)

This file must be generated manually. However, MOCAT ships with a version of this file '263MetaRefv9MG.cal.v2.seed.padded.motu.map'.

This file can NOT be copied and renamed if you have designed a different databse. It is a mapping file from fasta identifiers in the database to different species clusters, and has the format '<fasta identifier><TAB><sequence length><TAB>COG id><cluster ID>'

Example line:
351605.Gura_3679<TAB>1095<TAB>COG0012<TAB>OTUcal.v2.0

The database file .coord file (mOTU.v1.padded.coord) (same as for NCBI mode)

This file contains information of which bases should be counted when calculating base and insert coverages. Normally this file is generated automatically and contains the fasta entry names with start and stop bases, normally start=1 and stop=length of that sequence. Changing this file may be useful if you want to calculate the coverages based on only a part of a gene or genome.

Example line: 1001582.LL3_00013<TAB>101<TAB>1378

The database file .len file (mOTU.v1.padded.len) (same as for NCBI mode)

This file contains the fasta entries and their respective lengths, one per line. It is generated automatically and do not need to be changed.

Example line: 1001582.LL3_00013<TAB>1478

The database file .rownames and .rownames.uniq files (mOTU.v1.padded.rownames[.uniq]) (same as for NCBI mode)

These files contains the rownames of the computed base and insert coverages. These files are generated automatically and do not need to be modified. I the .uniq file the _X_X corresponds to the region over which the coverages are calculated. This is normally _1_[length of sequence]. This region can be changed in the .coord file.

functional mode: Design and requirements for summarizing gene profiles into functional categories

All that is required to summarize gene profiles into functional profiles, is the additional mapping file: DBNAME.functional.map. Note that this file could be gzipped and saved like: DBNAME.functional.map.gz. MOCAT will first look for the gzipped version. If multiple databases have been mapped against a format example of the name would be: DBNAME.1-3.functional.map.gz.

The format of the mapping file should first have a header with a '#' at the beginning of the line (IMPORTANT!):

#gene <TAB> cog <TAB> ko <TAB> module <TAB> pathway

Then each line has a gene ID form the database and the corresponding category:

geneID <TAB> cogID <TAB> koID <TAB> moduleID <TAB> pathwayID

Statistics Options

`-sfq|stats_fastqc`

Runs FastQC on each lane for each sample. The output is described in detail at *http://www.bioinformatics.bbsrc.ac.uk/projects/fastqc/*.

`-ss|sample_status`

This option prints a progress status of each sample into the file <SAMPLE

FILE>.status AND summarizes most of the stats files in each sample folder and stores in the files <SAMPLE FILE>.*. The columns represent: if a preset 5' trimming base is defined in the file 'MOCAT.cutoff5prime' (M if set in file, A if not - A means it should be automatically determined), if FastQC has been run on the lanes for the sample, if reads have been: trimmed and filtered, screened against database, screened against fasta file, assembled, assembly revised, genes predicted on an assembly and genes predicted on a revised assembly.

Additional Options

`-x|no_execute` (optional)

Only create the job files to execute, but don't execute them. Note that it's normal for the program to check for existing files, but not finding them (because no jobs were executed).

`-nt|no_temp` (optional)

If specified, overrides any temp folder settings. All temporary files will be stored in CWD/SAMPLE/temp/.

`-cpus` (not recommended)

MOCAT has been optimized to be run on a cluster of nodes with a larger number of cores (>8). This menas that, without setting the number of available cores for each sample (job), the number of cores listed below will be used in corresponding option. However, if you are using a system with fewer cores that 8, or if you think it will be better to use more cores than 8 when, for example, assembling samples, you can specify this option. Do note, by doing so, other options, such as read_trim_filter or screen, will also occupy the specified number of cores, without actually using them. Number of cores per option:

-read_trim_filter : 3

-screen : 8

-screen_fastafile : 2

-assembly : 8

-assembly_revision : 8

-gene_prediction : 1

-filter : 8

-profiling : 4

-fetch_mg : 4

modules : 8

Initital setup

Installing required external software

All steps in MOCAT are dependent on external software. Some of these software you have to download manually, as they require a licence key.

Screen Fasta File

If you want to perform a screen using a fasta file (NOT screen against a databse, which uses SOAPaligner2), this step uses Usearch. You can download Usearch from *http://www.drive5.com/usearch/nonprofit_form.html*. After downloading, extract the Usearch executable into the folder 'MOCAT_PATH/ext/usearch' and rename the file to 'usearch'. Renaming the file is required because you may download a newer version than was initially used in developing MOCAT, and by renaming it to only 'usearch', MOCAT can identify the executable. NOTE: make sure that the file is made 'executable', this can be done by typing (when in the ext/usearch directory) 'chmod u+x usearch'.

Gene Prediction

Gene prediction is done by MetaGeneMark, or Prodigal (Default). You can download MetaGeneMark from *http://exon.gatech.edu/GeneMark/license_download.cgi*. After downloading MetaGeneMark, extract the files into 'MOCAT_PATH/ext/metagenemark'. Normally you also have to copy the 'gm_key' file to your home directory, but MOCAT will do this for you, if needed.

Required paths in UNIX

Make sure that the MOCAT/src folder is in your UNIX path variable $PATH. This can be done by typing (in the UNIX shell) 'echo $PATH:/usr/me/MOCAT/src > $PATH', assuming you installed MOCAT into the directory /usr/me. By doing this you can execute the MOCAT.pl script from any directory.

Add the MOCAT/src folder to the UNIX variable PERL5LIB. This can be done by typing 'echo $PERL5LIB:/usr/me/MOCAT/src > $PERL5LIB', or if the $PERL5LIB variable is empty: 'echo /usr/me/MOCAT/src > $PERL5LIB', assuming you installed MOCAT to the /usr/me direcotry.

Setup MOCAT in a new folder

To initiate a new folder structure for MOCAT three steps are required.

1. Copy the config file from the MOCAT/GETTING_STARTED directory into the current direcotry. Make sure all settings in the config are set appropriately. Especially the GLOBAL SETTINGS section.

2. For each sample to be analysed, create a subfolder. Within this subfolder each paired end files should be named: LANE_NAME.1.fq[.gz] and LANE_NAME.2.fq[.gz]. Note that it is possible to have several lanes within one sample.

3. Make a file within the current folder (suggested) where you store the names of the samples to be analysed.

Example of a folder structure

Current folder is: /usr/me/project. Then the following setup would be appropriate to run MOCAT:

Sample files (aoption A):

/usr/me/project/SAMPLE_1/lane1.1.fq

/usr/me/project/SAMPLE_1/lane1.2.fq

/usr/me/project/SAMPLE_1/lane2.1.fq

/usr/me/project/SAMPLE_1/lane2.2.fq

/usr/me/project/SAMPLE_1/lane3.1.fq

/usr/me/project/SAMPLE_1/lane4.2.fq

/usr/me/project/SAMPLE_2/laneID.1.fq

/usr/me/project/SAMPLE_2/laneID.2.fq

Sample files (aoption B):

/usr/me/project/SAMPLE_1/lane1.pair.1.fq

/usr/me/project/SAMPLE_1/lane1.pair.2.fq

/usr/me/project/SAMPLE_1/lane1.single.fq

/usr/me/project/SAMPLE_1/lane2.pair.1.fq

/usr/me/project/SAMPLE_1/lane2.pair.2.fq

/usr/me/project/SAMPLE_1/lane2.single.fq

Required files:

/usr/me/project/MOCAT.cfg

/usr/me/project/MOCAT.samples

Content of MOCAT.samples file:

SAMPLE_1

SAMPLE_2

The MOCAT configuration file - MOCAT.cfg

This file contains settings assumed to not be changed equally often as those entered each time when executing the program. The file is divided into different sections, usually one for each possible pipeline option (but not all pipeline options need additional settings).

Global Settings Section

MOCAT_umask : 0022 [0022]

Sets the UNIX umask options that should be used. If set to 0002, for example, also members of the same group can write to files created by the user. Preferably this is combined with adding 'umask 0002' in the MOCAT_pre_execute option.

Here you can set a command that is executed before the actual job. Here you can set a specific umask, or perhaps run the command 'newgrp' if you want to run all jobs as a specific group.

MOCAT_LSF_qsub_add_param : [-l select=mem=6gb]

Additional parameters for LSF queuing system

MOCAT_LSF_queue : []

Sets the specific LSF '-queue' flag.

MOCAT_LSF_memory_limit : []

Sets the specific LSF '-M' flag

qsub_system : SGE [SGE,PBS,LSF,none]

Currently SGE is supported. If you are using a UNIX cluster without a queuing system set it to 'none'. MOCAT has been tested on SGE queuing system version GE 6.2u4 and GE 6.2u5 and a PBS as well as a LSF queuing system.

MOCAT_SGE_qsub_add_param : [-l mem_free=6G -l h_vmem=6G]

Additional parameters for SGE queuing system

MOCAT_PBS_qsub_add_param : [-l select=mem=6gb]

Additional parameters for PBS queuing system

MOCAT_SGE_parallell_env : make [make,orte,...]

This options specifies which parallel environment to use. It is set during the installation process but can easily be changed here. If it's incorrect jobs cannot be submitted. What the different options are only the administrator of your system knows.

MOCAT_dir : /bin/MOCAT

the main MOCAT directory, in which the /scr, /bin, /cnf and /data are located.

MOCAT_temp_dir_1 : HOSTNAME|/tmp

MOCAT_temp_dir_2 : any|/tmp

MOCAT has an advanced system to specify temporary directories. The simplest way, is to not specify any of the tags MOCAT_temp_dir_X, then all temporary files will be stored in

CWD/SAMPLE/temp/. The next level is to specify a temp directory for any and all systems you are using, this can be done by specifying 'MOCAT_temp_dir_1 : any_/tmp'. Here, 'any' refers to the hostname of the clusters, by specifying 'any' the temp directory till be the trailing '/tmp' on any system. The '_' is used internally in MOCAT to differentiate between hostname and start of path. If you wish to specify specific temporary directories for specific host computers, this is possible by adding as many MOCAT_temp_dir_1, MOCAT_temp_dir_2 ... MOCAT_temp_dir_N tags as desired. Please note that the tags are evaluated in order. This means if the specified hostname matches the systems's hostname, this temporary directory is used.

MOCAT_data_type : solexaqa [fastx,solexaqa]

Reads can be trimmed and filtered using either FastX 3' trimming or SolexaQA 3' trimming. Very generally, the SolexaQA trimmming is more strict.

MOCAT_paired_end : yes [yes]

Defines whether sample data is paried end reads or not.

MOCAT_zip_program : pigz [gzip,pigz]

Zip program. PigZ is a parallell version of gzip, provided with MOCAT. pigz is not supported under OSX.

MOCAT_default_reads : reads.processed [-define if desired-,'reads.processed','DATABASE']

By setting this options, you do not have to specify many or the additional 'READS ORIGIN' options. Eg, if you''d do a screen step, you'd normally write MOCAT.pl -sf FILE -sdb DB -sr reads.processed. If this is set to reads.processed, you'd only type: MOCAT.pl -sf FILE -sdb DB.

MOCAT_zip_level : 1 [1-9]

Determines how much the zip files are compressed. 9 is higher but slower. In our tests, fasta and fastq files are compressed very well already using compression level 1 or 2.

MOCAT_mapping_mode : allbest [allbest,random,unique]

Corresponds to SOAPsligner2's 'r' option. Mode 'allbest' is not supported under OSX.

MOCAT_prompt_before_run : no [yes,no]

If set to yes, the settings have to be confirmed each time before running MOCAT

Read Trim and Filter Section

To remove low quality and short reads, the raw reads are trimmed and filtered. This is done using either the FastX or SolexaQA algorithm and additional 5' trimming.

readtrimfilter_length_cutoff : 45

Removes reads that are shorter than this cutoff. We have mainly been using 30 or 45.

readtrimfilter_qual_cutoff : 20

The quality cutoff, at which reads are trimmed. Used internally in the FastX and SolexaQA routines. We recommend 20.

readtrimfilter_use_sanger_scale : auto [yes,no,auto] (if files are in Illumina 1.8+ format, use Sanger scale)

Older Illumina sequences have Illumina quality scale. Later versions have a Sanger quality scale. This would normally be set to 'auto', but can be set to 'yes' or 'no' if you have problems with this for some reason.

readtrimfilter_trim_5_prime : yes [yes,no]

Defines whether the 5' end of each read should be trimmed or not.

readtrimfilter_use_precalc_5prime_trimming : no [yes,no]

> MOCAT normally does 5' trimming automatically (if 'readtrimfilter_trim_5_prime' set to 'yes'). If you wish to specify at which base the reads should be trimmed for each sample, set this to 'yes'. If set to 'yes', the bases to trim at should be provided in the file 'MOCAT.cutoff5prime' (project folder, same location as MOCAT.cfg). The format of the file is: <SAMPLE> TAB <LANE> TAB <FIRST BASE TO KEEP>. Lane here is the file name up to the '.single', or '.pair'. Note that if this is set to 'no', after running read_trim_filter, MOCAT will produce the file 'MOCAT.cutoff5prime_calculated' in the current folder. If you do not want to perform 5' trimming set 'readtrimfilter_trim_5_prime' to 'no'.

Screen Section

Custom databases, or provided database 'hg19', can be used to screen reads and exclude these reads from further steps. Here, SOAPAligner2 is used to align reads to the specified database. The reads that match the database are removed.

screen_length_cutoff : 45

> After alignment, the SOAPAligner2 output file is filtered using a read length cutoff. If the read is shorter than the cutoff, the alignment will be discarded.

screen_percent_cutoff : 95

> Currently this cutoff should not be below 90. below 93.3%, we cannot guarantee that the criterium is fulfilled. While going below will be unreliable. After alignment, the SOAPAligner2 output file is filtered using an alignment percent identity cutoff. If the % id is lower than the cutoff, the alignment will be discarded.

screen_soap_seed_length : 30

> This is the "-l" option in SOAPAligner2. It means that the first 30 bp of a read are taken as a seed. In this seed region, you allow for a maximum of 2 mismatches, which equals 93.3%, see also above. The remainder of the read can have up to x mismatches, specified by the next flag.

screen_soap_max_mm : 10

> Maximum number of mismatches allowed on a read, apart from the seed. This will guarantee that reads of 100 bp can be mapped down to 88% identity.

screen_soap_cmd : -M 4 [-M 4]

> Set of additional paramters parsed to SOAP. They are provided here, so you can change them - however this is not recommended.

screen_save_format : sam [soap,sam]

> Defines the output format of the extracted reads. If set to sam, the soap file is additionally converted into sam format

Screen Fasta File Section

Read that have been trimmed and filter (and perhaps screened against a custom database) can be screened against a fasta file or DNA sequences. This is done by BLAST using USearch.

screen_fasta_file_usearch_version : 6 [5,6]

> Specify Usearch version. We recommend Usearch 5, but Usearch v6 is supported.
>
> item screen_fasta_file_usearch_version_5_exe : usearch (path relative to MOCAT_DIR/ext/usearch/)
>
> Path to executable for Usearch5. These two options have been set if you're running different Usearch versions for different projects, for example.
>
> item screen_fasta_file_usearch_version_6_exe : usearch (path relative to MOCAT_DIR/ext/usearch/)

Path to executable for Usearch6. These two options have been set if you're running different Usearch versions for different projects, for example.

screen_fasta_file_blast_e_value : 0.00001

If a read\s match has an e-value below this specified value, it is removed.

screen_fasta_file_blast_read_min_length : 10

The minimum length of reads to be screened. This applies especially to the sequences in the fasta file. For example, if the fasta file contains very short adapter sequences, this value should be less than, or equal, to the shortest sequence length in the specified fasta file.

screen_fasta_file_additional_usearch_cmd : []

Additional commands that could be sent to Usearch.

Filter Section

After any screen step, reads are filtered, by removing reads below a certain % ID cut off, a certain length cut off, and removing both paired-end reads if one of the two reads match the database.

filter_psort_buffer : 2G

Maximum memory requierment for psort. If you filter reads mapped against a large database, sorting the reads will take longer time. This time is reduced by increasing the amount of RAM allocated for this step. Change this setting depending on your system, if needed. We sometimes use 50GB of RAM for this step.

filter_length_cutoff : 45

The minimum length of a read

filter_percent_cutoff : 95

The minimum % identity for a read not be removed

filter_paired_end_filtering : yes [yes,no]

We recommend setting this to yes. This will remove read A2, if read A1 matches the database. For example if you screen against the hg19 database, and read A1 matches the database. It is likely that also read A2 should be removed, which it will be if this is set to yes.

filter_remove_mapping_files : no [yes.no]

Set to yes, if you want to remove the original mapping files after filtering. This will clear up some disk space.

Assembly Section

Reads are assembled using any of the supported version of SOAPDenovo. First, if paired end reads, the insert size is calculated using either a pre-assembly, or a pre-mapping (pre-align) step. Then the reads are assembled.

assembly_soap_version : 1.06 [1.05,1.06]

Specifies which SOAPDenovo version to use. Version 1.06 has several improvements over version 1.05, but version 1.05 is still supported. When running under OSX it is hard coded in the MOCATAssembly.pm script that version 1.06OSXnobamaio is used. This because there are probably library issues when using 1.06 under OSX. The difference between these two versions is that 2 output files are not printed in the nobamaio version. However, these files are not used and deleted when running MOCAT.

assembly_calculate_insert_size_using : mapping [assembly,mapping]

If reads are paired end, a more accurate insert size, than the estimate from library preparation, is calculated. This can be done by either an assembly step, or by aligning reads to a database. If 'mapping' is specified a database need to be specified as well (see below). If 'assembly' is specified, the file 'MOCAT.preliminary_insert_sizes' is required. The format of the file is <SAMPLE> TAB <LANE> TAB <insert size>, for each sample and lane, one lane per line. LANE here is the name of the files in the sample folder, up to the .1 and .2, excluding the .1 and .2. This file should be located in the project folder (same location as MOCAT.cfg). Using 'mapping' is faster, but not more accurate.

assembly_db_for_calc_insertsize : 1506MG (used if specified 'mapping' above)

A database, against which reads are aligned to calculate insert sizes, if 'mapping' is specified in the previous setting. The database '1506MG' is provided, which consists of 40 marker genes from 1506 reference genomes.

assembly_scaftig_min_length : 500 [500]

Sets the minimum length of a scaftig to be kept.

Assembly Revision Section

Revised assemblies are created by realigning reads to an existing assembly, then correcting for indels and base pair errors.

assembly_revision_scaftig_min_length : 500 [500]

Sets the minimum length of a scaftig to be kept.

Gene Prediction

Predicts protein coding genes on assemblies or revised assemblies.

gene_prediction_software : Prodigal [MetaGeneMark,Prodigal]

If you want to use Prodigal or MetaGeneMark.

gene_prediction_input : scaftig [scaftig,contig,scafSeq] (if revised assembly, can only be 'scaftig')

Determines whether to predict genes on scaftigs, contigs or scaffolds (scafSeq). If predicting on a revised assembly, only scaftig can be selected.

gene_prediction_prodigal_cmd : -f gff [-f xxx, -none-] (-p, -o, -a, -d, -i already set at runtime in MOCAT)

Additional settings for Prodigal

Experimental settings

These settings are by default turned off as they are under development and may not function as expected. However turning them on will not affect any result files.

realtime_status_use : no [yes,no]

If set to yes, you will see automatic updates of the status any memory usage of each sample. This will only have affect under the SGE queuing system.

realtime_status_timer : 5

A timer in seconds how often the data is updated.

realtime_status_log : no

If sets to yes, print more bug searching information.

realtime_status_fix_1 : -hostname-

You can add as many status_fix as you like. Here you would list hostnames that are only on one single machine. It is not needed to set any hosts here, but if you do, the status script will assume that the hosts set here have all jobs submitted to one particular node. Eg, if you have a system with three different nodes and the queuoing system could submit jobs to all these nodes, o not enter the hostname here. But if all jobs from a host will end up on that host, do set that hostname here. It has to do with whether the status script should attempt to SSH into the nodes to get the information about current jobs or not. If a hostname is provided here, no SSH will be attempted.

realtime_status_fix_2 : -hostname-

See above.

Output files

NOTE: This list of output files may not be exhaustive. We recommend you to investigate the different output files manually to know exactly what youre current version of MOCAT provides.

Each step in the pipeline generates a number of files. Here the main files produced in each step are described. For the example below, we assume that the MOCAT_data_type in the config file is solexaqa, a database named 'hg19' and that we are processing paired end reads. The fasta file screen was performed after a database screen. The output files differ slightly if the data is not paired end (apired end not supported in this evrsion).

Additionally, note, that you can summarize these output files using the option -ss, see above. Then the summary files are written to the files <SAMPLE FILE>.XXX.

Read Trim Filter

Main folder: reads.processed.solexaqa/

LANE.1.fq.gz.qual_stats

LANE.2.fq.gz.qual_stats

Internal quality stats files used for trimming.

LANE.pair.1.fq.gz

LANE.pair.2.fq.gz

LANE.single.fq.gz

Contains the processed reads for pair 1, 2, and single reads.

Screen Database

Main folders: reads.screened.hg19.solexaqa AND reads.extracted.hg19.solexaqa AND reads.mapped.hg19.solexaqa

reads.screened.hg19.solexaqa/LANE.screened.hg19.pair.1.fq.gz

reads.screened.hg19.solexaqa/LANE.screened.hg19.pair.2.fq.gz

reads.screened.hg19.solexaqa/LANE.screened.hg19.single.fq.gz

Contains the screened (kept) reads for pair 1, 2, and single reads.

reads.extracted.hg19.solexaqa/LANE.extracted.hg19.pair.1.fq.gz

reads.extracted.hg19.solexaqa/LANE.extracted.hg19.pair.2.fq.gz

reads.extracted.hg19.solexaqa/LANE.extracted.hg19.single.fq.gz

Contains the extracted reads for pair 1, 2, and single reads.

reads.screened.hg19.solexaqa/SAMPLE.all.screened.hg19.aligned.hg19.ids

Contains the identifiers of the reads that matched the database. These reads were removed from the input files and are not in any of the files above.

reads.screened.hg19.solexaqa/SAMPLE.all.screened.hg19.aligned.soap

Internal output from SOAPAligner2 containing information about the read that were removed.

reads.screened.hg19.solexaqa/SAMPLE.all.screened.hg19.aligned.sam.gz

GZipped SAM formatted file of the SOAP output file mentioned above. These are the reads that were removed/extracted in SAM format.

Screen Fasta File

Main folder: reads.screened.'fasta file name'.solexaqa AND reads.extracted.'fasta file name'.solexaqa AND reads.mapped.'fasta file name'.solexaqa

LANE.screened.'fasta file name'.solexaqa/LANE.screened.'fasta file name'.pair.1.fq.gz

LANE.screened.'fasta file name'.solexaqa/LANE.screened.'fasta file name'.pair.1.fq.gz

LANE.screened.'fasta file name'.solexaqa/LANE.screened.'fasta file name'.pair.1.fq.gz

Contains the screened (kept) reads for pair 1, 2, and single reads.

LANE.extracted.'fasta file name'.solexaqa/LANE.extracted.'fasta file name'.pair.1.fq.gz

LANE.extracted.'fasta file name'.solexaqa/LANE.extracted.'fasta file name'.pair.1.fq.gz

LANE.extracted.'fasta file name'.solexaqa/LANE.extracted.'fasta file name'.pair.1.fq.gz

Contains the extracted reads for pair 1, 2, and single reads..

LANE.1.readsMatchingFastaFile

LANE.2.readsMatchingFastaFile

Contains Usearch blast results for each pair (.1, .2). These files are used internally to cerate the .readsToRemove file.

SAMPLE.readsToRemove

Contains the read id's of all reads that were removed to create the results files. Note that the reads here are missing the trailing pair identifier (1 or 2). This is because if paired read 1 is filtered, so is also read 2.

Assembly

Main folder: assembly.hg19.solexaqa.K19/ (19 represents the K-mer size for that assembly)

SAMPLE.assembly.hg19.solexaqa.K19.config

This is the config file used in SOAPDenovo to generate the assembly.

SAMPLE.assembly.hg19.solexaqa.K19.contig

Contains the contigs produced.

SAMPLE.assembly.hg19.solexaqa.K19.scafSeq

Contains the full scafolds.

SAMPLE.assembly.hg19.solexaqa.K19.scaftig

Contains all scaftigs longer than specified (normally 500 bp).

Assembly Revision

Main folder: assembly.revised.hg19.solexaqa.K19/

SAMPLE.assembly.revised.hg19.solexaqa.K19.scaftig

Revised scaftigs longer than specified length (normally 500 bp).

Gene Prediction

Main folder: gene.prediction.assembly.revised.hg19.solexaqa.K19.500.'MetaGeneMark or Prodigal'

Here it is 'assembly.revised' because we used a revised assembly when predicting the genes. The .500 means that genes are predicted on scaftigs that are longer than 500bp, this number is taken from the value in the config file.

SAMPLE.gene.prediction.assembly.revised.hg19.solexaqa.K19.faa

Amino acid sequences of predicted proteins.

SAMPLE.gene.prediction.assembly.revised.hg19.solexaqa.K19.fna

Nucleotide sequences of predicted genes.

SAMPLE.gene.prediction.assembly.revised.hg19.solexaqa.K19.lst

MetaGeneMark output file.

SAMPLE.gene.prediction.assembly.revised.hg19.solexaqa.K19.tab

A summary table of all predicted genes. This information is also available in the headers of the predicted genes and proteins.

Filter

Main folder: reads.filtered.reads.processed.solexaqa

Profiling

Main folders: [base|insert].coverage.'database'.'solexaqa or fastx'

There are two main folders in each sample folder: base and insert coverages. Inside each of these folders are files with length normalized and raw count data. The rownames for each database are stored in the data_dir. These files can be concatenated together with the rowname-file in the database directory to generate custom files with calculated coverages for a combination of samples.

Additionally, if the mode NCBI or mOTU was run, results are saved in the folders taxonomic.profiles.XXX and motu.profiles.XXX

Fetch Marker Genes

Main folder: gene.fetched.MGs.assembly.revised.hg19.solexaqa.K19.500.'MetaGeneMark or Prodigal'

In this folder the fetched (extracted) marker genes are stored.

Statistics files

NOTE: This list of statistics files may not be exhaustive. We recommend you to investigate the different output files manually to know exactly what youre current version of MOCAT provides.

Each step in the pipeline prints different statistics files. These are stoed in the /stats folder in each sample folder.

Read Trim Filter

LANE.1.fq.gz_raw.reads.stats

LANE.2.fq.gz_raw.reads.stats

Number of raw reads and number of raw bases

SAMPLE.readtrimfilter.solexaqa.stats

After filtering: number of reads, bases, maximum read length, average read length, estimated K-mer and inserts.

Screen Database

SAMPLE.screen.hg19.solexaqa.stats

After screening using database: number of reads, bases, maximum read length, average read length, estimated K-mer and inserts.

SAMPLE.extracted.hg19.solexaqa.stats

After screening using database, stats on teh extracted reads: number of reads, bases, maximum read length, average read length, estimated K-mer and inserts.

Screen Fasta File

SAMPLE.fastafile.hg19.solexaqa.stats

After screening using fasta file: number of reads, bases, maximum read length, average read length and estimated K-mer.

LANE.1.screen.fastafile.solexaqa.stats

LANE.2.screen.fastafile.solexaqa.stats

Total number of reads screened, total number of hits and total number of unique hits.

Assembly

SAMPLE.assembly.hg19.solexaqa.K19.assembly.stats

SAMPLE.assembly.hg19.solexaqa.K19.scaftig.stats

Specific statistics on each assembly.

SAMPLE.assembly.hg19.solexaqa.K19.inserts.stats

Contains the calculated insert sizes for each lane, for that specific sample.

Assembly Revision

=item SAMPLE.assembly.revised.hg19.solexaqa.K19.baseErrorAndIndelError.stats

Statistics describing how many single base errors, small indels regions were found, and their lengths.

Additional files used by MOCAT

All these files are stored in the working directory containing the sample folders.

MOCAT.cfg

The MOCAT configuration file, described above.

MOCAT.samples

Example name of the required sample file used to specify which samples to analyze.

MOCAT.cutoff5prime

File required as input if 'readtrimfilter_use_precalc_5prime_trimming' is set to 'yes' on the config file. See config file section.

MOCAT.cutoff5prime_calculated

File written by MOCAT, after read trim filter, with the chosen 5' cutoffs for each lane and sample. The number represents the first base to keep.

MOCAT.preliminary_insert_sizes

File required if 'assembly_calculate_insert_size_using' is set to 'assembly' in the config file. See config file section.

The /log directory

The /log directory created contains all temporary output log files created while running MOCAT. These files could be useful for tracking an error. Prior to being loacted in the /log directory, the files are located in the current working directory. All files in the /log directory can safely be removed at any time.

The log folder has a subfolder for each processing step. Eg:

assembly

assembly_revision

profiling

filter

gene_prediction

readtrimfilter

screen

screen_fasta_file

other

Please note that this folder may contain error messages not captured in the other log files. If you run into problems, it could be very fruitful to have a look into this folder!

resources

If you run MOCAT using the experimental settings set to 'yes', this folder will contain detailed information about how much memory, disk space and how many processors were used

Each processing folder has in turn these subfolders:

commands folder

Here is summarized information including which settings used for each tas performed by MOCAT

jobs folder

Here are the specific executed commands for each task performed by MOCAT

samples folder

Here are log files for each sample that contains any error messages from any processing step

startstop

here is information about when each job started and finished and the exit status

arrays

This folder contains the array files used to submit the jobs to the queues.

Citing MOCAT

If you have used MOCAT in your work, please cite:

Kultima JR, Sunagawa S, Li J, Chen W, Chen H, et al. (2012)

MOCAT: A Metagenomics Assembly and Gene Prediction Toolkit. PLoS ONE 7(10): e47656. doi:10.1371/journal.pone.0047656

MOCAT is a wrapper for 3rd party software. Therefore we strongly suggest you also cite the following papers if you use MOCAT:

Initial read trimming and quality control

Cox MP, Peterson DA, Biggs PJ (2010)

SolexaQA: At-a-glance quality assessment of Illumina second-generation sequencing data. BMC bioinformatics 11: 485 doi:10.1186/1471-2105-11-485.

FastX program:

http://hannonlab.cshl.edu/fastx_toolkit/

Mapping reads

Li R, Yu C, Li Y, Lam T-W, Yiu S-M, et al. (2009)

SOAP2: an improved ultrafast tool for short read alignment. Bioinformatics (Oxford, England) 25: 1966 to 1967 doi:10.1093/bioinformatics/btp336.

Edgar RC (2010)

Search and clustering orders of magnitude faster than BLAST. Bioinformatics. 2010 Oct 1;26(19):2460-1. doi: 10.1093/bioinformatics/btq461

Assembly

Li R, Zhu H, Ruan J, Qian W, Fang X, et al. (2010)

De novo assembly of human genomes with massively parallel short read sequencing. Genome research 20: 265 to 272 doi:10.1101/gr.097261.109.

Assembly revision

Li H, Durbin R (2009)

Fast and accurate short read alignment with Burrows-Wheeler transform. Bioinformatics (Oxford, England) 25: 1754 to 1760 doi:10.1093/bioinformatics/btp324.

Gene Prediciton

Hyatt D, Chen G-L, Locascio PF, Land ML, Larimer FW, et al. (2010)

Prodigal: prokaryotic gene recognition and translation initiation site identification. BMC bioinformatics 11: 119 doi:10.1186/1471 to 2105-11-119.

Zhu W, Lomsadze A, Borodovsky M (2010)

Ab initio gene identification in metagenomic sequences. Nucleic acids research 38: 1 to 15 doi:10.1093/nar/gkq275.

Retrieving Marker Genes

Sunagawa et al. (2013)

Metagenomic species profiling using universal phylogenetic marker genes. Nature Methods 10, 1196 to 1199 doi:10.1038/nmeth.2693

Supported Formats

MOCAT supports both older and newer (as of January, 2012) Illumina file formats. The old format has a fasta header descriptor like this: '@HWUSI-EAS100R:6:73:941:1973#0/1'. The new format has a fasta header like this: '@EAS139:136:FC706VJ:2:2104:15343:197393 1:Y:18:ATCACG'. However, when processing the reads in the first read trim filter step, the headers are converted into the old format. This menas the following information is lost: run id, flowcell id, control bit. Reads that are of low quality, ninth field equals to 'Y', are filtered in the first step. For more information, see *http://en.wikipedia.org/wiki/FASTQ_format*.

Author

The **MOCAT** pipeline was developed by Jens Roat Kultima & Shinichi Sunagawa (Bork Group, EMBL) in collaboration with BGI. External software used by the **MOCAT** pipeline are copyright respective authors.

Copyright

Copyright (c) 2011-2012. Jens Roat Kultima, Shinichi Sunagawa, EMBL and BGI. MOCAT is released under the GNU General Public Licence v3 (http://www.gnu.org/licenses/gpl.html).