# User Manual for MALT V0.3.6

Daniel H. Huson

May 1, 2016

## Contents

# 1 Introduction

MALT, an acronym for *MEGAN alignment tool*, is a sequence alignment and analysis tool designed for processing high-throughput sequencing data, especially in the context of metagenomics. It is an extension of MEGAN6, the *MEGenome Analyzer* and is designed to provide the input for MEGAN6, but can also be used independently of MEGAN6.

The core of the program is a sequence alignment engine that aligns DNA or protein sequences to a DNA or protein reference database in either BLASTN (DNA queries and DNA references), BLASTX (DNA queries and protein references) or BLASTP (protein queries and protein references) mode. The engine uses a banded-alignment algorithm with affine gap scores and BLOSUM substitution matrices (in the case of protein alignments). The program can compute both local alignments (Smith-Waterman) or semi-global alignments (in which reads are aligned end-to-end into reference sequences), the latter being more appropriate for aligning metagenomic reads to references.

By default, MALT produces a MEGAN "RMA6" file that contains taxonomic and functional classifications of the reads that can be opened in MEGAN6. The taxonomic analysis use the naive LCA algorithm (introduced in [5]).

Used as an alignment tool, MALT can produce alignments in BLAST text format, BLAST-tab format or SAM format (both for DNA and protein alignments). In addition, the program can be used as a filter to obtain all reads that have a significant alignment, or do not have a significant alignment, to the given reference database.

MALT can also be used to compute a taxonomic analysis of 16S sequences. Here the ability to compute a semi-global alignment rather than a local alignment is crucial.

When provided with a listing of gene locations and annotations for a given database of DNA sequences, MALT is able to predict genes based on BLASTN-style alignments.

MALT actually consists of two programs, `malt-build` and `malt-run`. The `malt-build` program is first used to build an index for the given reference database. It can index arbitrary large databases, provided the used computer has enough memory. For maximum speed, the program uses a hash-table and thus require a large memory machine. The `malt-run` program is then used to perform alignments and analyses.

MALT does not use a new approach, but is rather a new carefully crafted implementation of existing

approaches. The program uses spaced seeds rather than consecutive seeds [2, 8]. It uses a hash table to store seed matches, see, for example, [10]. It uses a reduced alphabet to determine potential matches between protein sequences [9, 14]. Finally, it uses a banded alignment algorithm [3] that can compute both local and semi global alignments.

Both programs make heavy use of parallelization and require a lot of memory. The ideal *hardware requirements* are a linux server with 64 cores and 512 GB of memory.

MALT performs alignment and analysis of high-throughput sequencing data in a high-throughput manner. Here are some examples:

1. Using the RefSeq microbial protein database (version 50, containing 10 million protein sequences with a total length of 3.2 billion amino acids), a BLASTX-style analysis of taxonomic and functional content of a collection of 11 million Illumina reads takes about 900 wall-clock seconds (using 64 cores). The program found about 4.5 million significant alignments covering about 15% of the total reads.

2. Using the Genbank DNA database (microbes and viruses, downloaded early 2013, containing about 2.3 million DNA sequences with a total length of 11 billion nucleotides), a BLASTN-style analysis of one million reads takes about 70 wall-clock seconds. The program finds about two million significant alignments covering one quarter of the total reads.

3. Using the Silva database (`SSURef_NR99_115_tax_silva.fasta`, containing 479,726 DNA sequences with a total length of 690 million nucleotides), the semi-global alignment of 5000 16S reads takes about 100 seconds (using 64 cores), producing about 100,000 significant alignments.

This document provides both an introduction and a reference manual for MALT.

## 2 Getting Started

This section describes how to get started.

Download the program from http://www-ab.informatik.uni-tuebingen.de/software/malt, see Section 3 for details.

First, use `malt-build` to build an index for MALT. For example, to build an index for all viral proteins in RefSeq, download the following file: ftp://ftp.ncbi.nlm.nih.gov/refseq//release/viral/viral.1.protein.faa.gz

Put this file in a single directory called `references`, say. There is no need to unzip the file because MALT is able to read zipped files. Also, in general, when using more than one file of reference sequences, there is no need to concatenate the files into one file, as MALT can process multiple files.

The program `malt-build` will be used to build an index for viral reference sequences. We will write the index directory to a directory called `index`. In the parent directory of the `references` directory, run `malt-build` as follows:

```
set MALT=<path-to-malt-directory>
malt-build -i references/*.* -d index -g2t $MALT/data/gi_taxid_prot-2014Jan04.bin \
                  -tre $MALT/data/ncbi.tre.gz -map$MALT/data/ncbi.tre.gz -L megan5-license.txt
```

The input files are specified using `-i`, the index is specified using `-d`. The option `-g2t` is used to specify a GI to taxon-id mapping which will be used to identify the taxa associated with the reference sequences. A mapping file is supplied in the data directory of MALT. The options `-tre` and `-map` are used to access the NCBI taxonomy, which is needed to perform a taxonomic analysis of the reads as they are aligned. Use `-L` to explicitly provide a MEGAN5 license file to the program, if you have not previously used a licensed version of MEGAN5.

Then, use `malt-run` to analyze a file of DNA reads. Assume that the DNA reads are contained in two files, `reads1.fna` and `reads2.fna`. Call the program as follows:

```
malt-run -i reads1.fna reads2.fna -d index -m BlastX -o . -L megan5-license.txt
```

If either of the two programs abort due to lack insufficient memory, then please edit the files `malt-build-gui.vmoptions` and/or `malt-run-gui.vmoptions` to allocate more memory to the programs; By default, for testing purposes, the memory reserved for the programs is set to $64GB$. For comparison against the NCBI-NR database, for example, you will need about $300GB$.

All input files are specified using `-i`. The index to use is specified using `-d`. The option `-m` defines the alignment mode of the program, in this case `BlastX`. Use `-at` to specify the alignment type.The option `-om` is used to specify the output directory for matches. Here we specify the current directory (`.`). The option `--tax` requests that a taxonomic analysis of the reads be performed and `-om .` requests that the resulting MEGAN file be written to the current directory. The file option `-t` specifies the maximum number of threads.

By default, MALT uses memory mapping to access its index files. If you intend to align a large number of files in a single run of MALT, then it may be more efficient to have the program preload the complete index. To achieve this, use the command-line option `-mem false`.

# 3 Obtaining and Installing the Program

MALT is written in Java and requires a 64-bit Java runtime environment version 7 or latter, freely available from http://www.java.org. The Windows and MacOS X installers contain a suitable Java runtime environment that will be used if a suitable Java runtime environment cannot be found on the computer.

MALT is currently in "open alpha testing" and is available from:

http://www-ab.informatik.uni-tuebingen.de/software/malt.

There are three different installers that target major operating systems:

- `MALT_windows-x64_0.3.6.exe` provides an installer for Windows.
- `MALT_macos_0.3.6.dmg` provides an installer for MacOS X.
- `MALT_unix_0.3.6.sh` provides an installer for Linux and Unix.

Download the installer that is appropriate for your computer. Please note that the memory requirement of MALT grows dramatically with the size of the reference database that you wish to employ. For example, to align sequences against the NR database requires that you have 512GB of main memory.

Double-click on the downloaded installer program to start the interactive installation dialog.

Alternatively, under Linux, change into the directory containing the installer and type

```
./MALT_unix_0.3.6.sh
```

This will launch the MALT installer in GUI mode. To install the program in non-gui console mode, type

```
./MALT_unix_0.3.6.sh -c
```

Finally, when updating the installation under Linux, one can perform a completely non-interactive installation like this (quiet mode):

```
./MALT_unix_0.3.6.sh -q
```

The installation dialog will ask how much memory the program may use. Please set this variable carefully. If the amount needs to be changed after installation, then this can be done by editing the files ending on `vmoptions` in the installation directory.

Two copies of each of the program `malt-build` and `malt-run` will be installed. The two copies named `malt-build` and `malt-run` are intended in non-interactive, commandline use. The two copies named `malt-build-gui` and `malt-run-gui` provide a very simple GUI interface.

# 4 The MALT index builder

The first step in a MALT analysis is to build an index for the given reference database. This is done using a program called `malt-build` .

In summary, `malt-build` takes a reference sequence database (represented by one or more FastA files, possibly in `gzip` format) as input and produces an index that then can subsequently be used by the main analysis program `malt-run` as input. If MALT is to be used as an taxonomic and/or functional analysis tool as well as an alignment tool, then in addition, `malt-build` must be provided with a number of mapping files that are used to map reference sequences to taxonomic or functional classes, or to locate genes in DNA reference sequences.

The `malt-build` program is controlled by command-line options, as summarized in Figure 1. There are three options for determining input and output:

`--input` Use to specify all files that contains reference sequences. The files must be in FastA format and may be *gzipped* (in which case they must end on `.gz`.)

`--sequenceType` Use to specify whether the reference sequences are `DNA` or `Protein` sequences. (For `RNA` sequences, use the DNA setting).

`--index` Use to specify the name of the index directory. If the directory does not already exist then it will be created. If it already exists, then any previous index files will be overwritten.

There are two performance-related options:

`--threads` Use to set the number of threads to use in parallel computations. Default is 8. Set this to the number of available cores.

`--step` Use to set step size used to advance seed, values greater than 1 reduce index size and sensitivity. Default value: 1.

The most important performance-related option is the maximum amount of memory that `malt-build` is allowed to use. This cannot be set from within the program but rather is set during installation of the software.

MALT uses a seed-and-extend approach based on "spaced seeds" [2, 8]. The following options control this:

`--shapes` Use this to specify the seed shapes used. For DNA sequences, the default seed shape is: `111110111011110110111111`. For protein sequences, by default the program uses the following four shapes: `111101101110111`, `1111000101011001111`, `1110100100100010010111 1` and `1110100100001010001010011 1`. These seeds were suggested in [6], see `http://www.biomedcentral.com/content/supplementary/1471-2164-12-280-s1.pdf`.

`--maxHitsPerSeed` Use to specify the maximum number of hits per seed. The program uses this to calculate a maximum number of hits per hash value.

`--proteinReduct` Use this to specify the alphabet reduction in the case of protein reference sequences. By default, the program reduces amino acids to 8 different letters, grouped as follows: [LVIMC] [AG] [ST] [P] [FYW] [EDNQ] [KR] [H]. This is referred to as the *BLOSUM50_8* reduction in MALT and was suggested in [9].

MALT is able to generate RMA files that can be directly opened in MEGAN.

`--classify` Use this option to determine which classifications should be computed, such as Taxonomy, EGGNOG, INTERPRO2GO, KEGG and/or SEED.

There are numerous options that can be used to provide mapping files to `malt-build` for classification support. These are used by the program to map reference sequences or genes to taxonomic and/or functional classes.

`-g2taxonomy -a2taxonomy -s2taxonomy` Use to specify mapping files to map reference sequences to taxonomic identifiers (NCBI taxon integer ids). Use `-g2taxonomy` for a file mapping GI numbers to taxon ids. Use `-r2taxonomy` for a file mapping RefSeq identifiers to taxon ids. Use `-s2taxonomy` for a file that maps *synonyms* to taxon ids. A synonym is any word that may occur in the header line of a reference sequence.

`-g2interpro2go -r2interpro2go -s2interpro2go` Use to specify mapping files to map reference sequences to InterPro numbers [1, **?**] . The detailed usage of three different options is analogous to above.

`-g2seed -r2seed -s2seed` Use to specify mapping files to map reference sequences to SEED [11] classes. Unfortunately, the SEED classification does not assign numerical identifiers to classes. As a work-around, `malt-build` uses the numerical identifiers defined and used by MEGAN [5]. The detailed usage of three different options is analogous to above.

`-g2eggnog -r2eggnog -s2eggnog` Use to specify mapping files to map reference sequences to COG and NOG [13, 12] classes. Unfortunately, COG's and NOG's do not share the same space of numerical identifiers. As a work-around, `malt-build` uses the numerical identifiers defined and used by MEGAN [5]. The detailed usage of three different options is analogous to above.

`-g2kegg -r2kegg -s2kegg` Use to specify mapping files to map reference sequences to KEGG KO numbers [7] . The detailed usage of three different options is analogous to above.

There are a couple of other options:

`--firstWordOnly` Use to specify to save only the first word of each reference header. Default value: false.

`--random` Use to specify the seed used by the random number generator.

`--verbose` Use to run program in verbose mode.

`--help` Report command-line usage.

```
SYNOPSIS
MaltBuild [options]
DESCRIPTION
Build an index for MALT (MEGAN alignment tool)
OPTIONS
 Input:
-i, --input [string(s)]              Input reference file(s). Mandatory option.
-s, --sequenceType [string]          Sequence type. Mandatory option. Legal values: DNA, Protein
 Output:
-d, --index [string]                 Name of index directory. Mandatory option.
 Performance:
-t, --threads [number]               Number of worker threads. Default value: 8.
-st, --step [number]                 Step size used to advance seed, values greater than 1 reduce index size and sensitivity. Default value: 1.
 Seed:
-ss, --shapes [string(s)]            Seed shape(s). Default value(s): default.
-mh, --maxHitsPerSeed [number]       Maximum number of hits per seed. Default value: 1000.
-pr, --proteinReduct [string]        Name or definition of protein alphabet reduction (BLOSUM50_10,BLOSUM50_11,BLOSUM50_15,BLOSUM50_4,BLOSUM50_8,DIAMOND_11,GBMR4,HSDM17,M
 Classification:
-c, --classify [string(s)]           Classifications (any of EGGNOG INTERPRO2GO KEGG SEED Taxonomy). Mandatory option.
-g2eggnog, --gi2eggnog [string]      GI-to-EGGNOG mapping file.
-r2eggnog, --ref2eggnog [string]     RefSeq-to-EGGNOG mapping file.
-s2eggnog, --syn2eggnog [string]     Synonyms-to-EGGNOG mapping file.
-g2interpro2go, --gi2interpro2go [string]   GI-to-INTERPRO2GO mapping file.
-r2interpro2go, --ref2interpro2go [string]   RefSeq-to-INTERPRO2GO mapping file.
-s2interpro2go, --syn2interpro2go [string]   Synonyms-to-INTERPRO2GO mapping file.
-g2kegg, --gi2kegg [string]          GI-to-KEGG mapping file.
-r2kegg, --ref2kegg [string]         RefSeq-to-KEGG mapping file.
-s2kegg, --syn2kegg [string]         Synonyms-to-KEGG mapping file.
-g2seed, --gi2seed [string]          GI-to-SEED mapping file.
-r2seed, --ref2seed [string]         RefSeq-to-SEED mapping file.
-s2seed, --syn2seed [string]         Synonyms-to-SEED mapping file.
-g2taxonomy, --gi2taxonomy [string]   GI-to-Taxonomy mapping file.
-a2taxonomy, --ref2taxonomy [string]  Accession-to-Taxonomy mapping file.
-s2taxonomy, --syn2taxonomy [string]   Synonyms-to-Taxonomy mapping file.
-tn, --parseTaxonNames               Parse taxon names. Default value: true.
-gif, -geneInfoFile [string]         File containing gene information.
 Other:
-fwo, --firstWordOnly                Save only first word of reference header. Default value: false.
-rns, --random [number]              Random number generator seed. Default value: 666.
-hsf, --hashScaleFactor [number]     Hash table scale factor. Default value: 0.9.
-v, --verbose                        Echo commandline options and be verbose. Default value: false.
-h, --help                           Show program usage and quit.
```

Figure 1: Summary of command-line usage of malt-build.

# 5 The MALT analyzer

In summary, the program `malt-run` is used to align one or more files of input sequences (DNA or proteins) against an index representing a collection of reference DNA or protein sequences. In a preprocessing step, the index is computed using the `malt-build`, as described above. Depending on the type of input and reference sequences, the program can be be run in BLASTN, BLASTP or BLASTX mode.

The `malt-run` program is controlled by command-line options (see Figure 2). The first options specifies the program mode and alignment type.

`--mode` Use this to run the program in *BlastN mode* , *BlastP mode* or *BlastX mode* , that is, to align DNA and DNA, protein and protein, or DNA reads against protein references, respectively. Obviously, the former mode can only be used if the employed index contains DNA sequences whereas the latter two modes are only applicable to an index based on protein reference sequences.

`--alignmentType` Use this to specify the type of alignments to be performed. By default, this is set to `Local` and the program performs *local alignment* just like BLAST programs do. Alternatively, this can be set to `SemiGlobal`, in which case the program will perform *semi global alignment* in which reads are aligned end-to-end.

There are two options for specifying the input.

`--inFile` Use this to specify all input files. Input files must be in FastA or FastQ format and may be gzipped, in which case their names must end on `.gz`.

`--index` Use this to specify the directory that contains the index built by `malt-build`.

There is a number of options for specifying the output generated by the program.

`--output` Use to specify the names or locations of the output RMA files. If a single directory is specified, then one output file per input file is written to the specified directory. Alternatively, if one or more output files are named, then the number of output files must equal the number of input files, in which case the output for the first input file is written to first output file, etc.

`--includeUnaligned` Use this to ensure that all unaligned queries are placed into the output RMA file. By default, only queries that have an alignment are included in the output RMA file.

`--alignments` Use to specify the files to which alignments should be written. If a single directory is specified, then one output file per input file is written to the specified directory. Alternatively, if one or more output files are named, then the number of output files must equal the number of input files, in which case the output for the first input file is written to first output file, etc. If the argument is the special value `STDOUT` then output is written to standard-output rather than to a file. If this option is not supplied, then the program will not output any matches.

`--format` Determines the format used to report alignments. The default format is `SAM`. Other choices are `Text` (full text BLAST matches) and `Tab` (tabulated BLAST format).

`--gzipOutput` Use this to specify whether alignment output should be gzipped. Default is true.

`--outAligned` Use this to specify that all reads that have at least one significant alignment to some reference sequence should be saved. File specification possibilities as for `--alignments`.

`--samSoftClip` Request that SAM output uses soft clipping.

`--sparseSAM` Request a sparse version of SAM output. This is faster and uses less memory, but the files are not necessary compatible with other SAM processing tools.

`--gzipAligned` Compress aligned reads output using gzip. Default value: true.

`--outUnaligned` Use this to specify that all reads that do not have any significant alignment to any reference sequence should be saved. File specification possibilities as for `--alignments`.

`--gzipUnaligned` Compress unaligned reads output using gzip. Default value: true.

There are three performance-related options:

`--threads` Use to set the number of threads to use in parallel computations. Default is 8. Set this to the number of available cores. -rqc, Cache results for replicated queries.

`--memoryMode` Load all indices into memory, load indices page by page when needed or use memory mapping (load, page or map).

`--maxTables` Use to set the maximum number of seed tables to use (0=all). Default value: 0.

`--replicateQueryCache` Use to turn on caching of replicated queries. This is especially useful for processing 16S datasets in which identical sequences occur multiple times. Turning on this feature does not change the output of the program, but can cause a significant speed-up. Default value: false.

The most important performance-related option is the maximum amount of memory that `malt-run` is allowed to use. This cannot be set from within the program but rather is set during installation of the software.

The following options are used to filter matches by significance. Matches that do not meet all criteria specified are completely ignored.

`--minBitScore` Minimum bit score. Default value: 50.0.

`--maxExpected` Maximum expected score. Default value: 1.0.

`--minPercentIdentity` Minimum percent identity. Default value: 0.0.

`-maxAlignmentsPerQuery` Maximum number of alignments per query. Default value: 100.

`--maxAlignmentsPerRef` Maximum number of (non-overlapping) alignments per reference. Default value: 1. MALT reports up to this many best scoring matches for each hit reference sequence.

There are a number of options that are specific to the BlastN mode. They are used to specify scoring and are also used in the computation of expected values.

`--matchScore` Use to specify the alignment match score. Default value: 2.

`--mismatchScore` Use to specify the alignment mis-match score. Default value: -3.

`--setLambda` Parameter Lambda for BLASTN statistics. Default value: 0.625.

`--setK` Parameter K for BLASTN statistics. Default value: 0.41.

For BlastP mode and BlastX mode the user need only specify a substitution matrix. The Lambda and K values are set automatically.

`--subMatrix` Use to specify the protein substitution matrix to use. Default value: `BLOSUM62`. Legal values: `BLOSUM45`, `BLOSUM50`, `BLOSUM62`, `BLOSUM80`, `BLOSUM90`.

If the query sequences are DNA (or RNA) sequences, that is, if the program is running in BlastN mode or BlastX mode, then the following options are available.

`--forwardOnly` Use to align query forward strand only. Default value: false.

`--reverseOnly` Use to align query reverse strand only. Default value: false.

The program uses the LCA algorithm [4] to assign reads to taxa. There are a number of options that control this.

`lca_taxonomy` Use to specify that the LCA algorithm should be applied to the taxonomy classification. Similar switches are available to turn on the use of the LCA algorithm for other classifications. But using the LCA algorithms only makes sense when providing additional taxonomic classifications such as the RDP tree.

`--topPercent` Use to specify the *top percent* value for LCA algorithm. Default value is 10%. For each read, only those matches are used for taxonomic placement whose bit score is within 10% of the best score for that read.

`--minSupport` Use to specify the *min support* value for the LCA algorithm.

There are a number of options that control the heuristics used by `malt-run`.

`--maxSeedsPerFrame` Maximum number of seed matches per offset per read frame. Default value: 100.

`--maxSeedsPerRef` Maximum number of seed matches per read and reference. Default value: 20.

`--seedShift` Seed shift. Default value: 1.

The program uses a banded-aligner as described in [3]. There are a number of associated options.

`--gapOpen` Use this to specify the gap open penalty. Default value: 7.

`--gapExtend` Use this to specify gap extension penalty. Default value: 3.

`--band` Use this to specify width/2 for banded alignment. Default value: 4.

The are a couple of other options:

`eplicateQueryCacheBits` Specify the number of bits used to cache replicate queries (default is 20).

`--verbose` Use to run program in verbose mode.

`--help` Report command-line usage.

```
SYNOPSIS
MaltRun [options]
DESCRIPTION
Align sequences using MALT (MEGAN alignment tool)
OPTIONS
 Mode:
-m, --mode [string]                  Program mode. Mandatory option. Legal values: Unknown, BlastN, BlastP, BlastX, Classifier
-at, --alignmentType [string]        Type of alignment to be performed. Default value: Local. Legal values: Local, SemiGlobal
 Input:
-i, --inFile [string(s)]             Input file(s) containing queries in FastA or FastQ format. Mandatory option.
-d, --index [string]                 Index directory as generated by MaltBuild. Mandatory option.
 Output:
-o, --output [string(s)]             Output RMA file(s) or directory.
-iu, --includeUnaligned              Include unaligned queries in RMA output file. Default value: false.
-a, --alignments [string(s)]         Output alignment file(s) or directory or STDOUT.
-f, --format [string]                Alignment output format. Default value: SAM. Legal values: SAM, Tab, Text
-za, --gzipAlignments                Compress alignments using gzip. Default value: true.
-ssc, --samSoftClip                  Use soft clipping in SAM files (BlastN mode only). Default value: false.
-sps, --sparseSAM                    Produce sparse SAM format (smaller, faster, suitable for MEGAN). Default value: false.
-oa, --outAligned [string(s)]        Aligned reads output file(s) or directory or STDOUT.
-zal, --gzipAligned                  Compress aligned reads output using gzip. Default value: true.
-ou, --outUnaligned [string(s)]      Unaligned reads output file(s) or directory or STDOUT.
-zul, --gzipUnaligned                Compress unaligned reads output using gzip. Default value: true.
 Performance:
-t, --numThreads [number]            Number of worker threads. Default value: 8.
-mem, --memoryMode [string]          Memory mode. Default value: load. Legal values: load, page, map
-mt, --maxTables [number]            Set the maximum number of seed tables to use (0=all). Default value: 0.
-rqc, --replicateQueryCache          Cache results for replicated queries. Default value: false.
 Filter:
-b, --minBitScore [number]           Minimum bit score. Default value: 50.0.
-e, --maxExpected [number]           Maximum expected score. Default value: 1.0.
-id, --minPercentIdentity [number]   Minimum percent identity. Default value: 0.0.
-mq, --maxAlignmentsPerQuery [number]   Maximum number of alignments per query. Default value: 25.
-mrf, --maxAlignmentsPerRef [number]   Maximum number of (non-overlapping) alignments per reference. Default value: 1.
 BlastN parameters:
-ma, --matchScore [number]           Match score. Default value: 2.
-mm, --mismatchScore [number]        Mismatch score. Default value: -3.
-la, --setLambda [number]            Parameter Lambda for BLASTN statistics. Default value: 0.625.
-K, --setK [number]                  Parameter K for BLASTN statistics. Default value: 0.41.
 BlastP and BlastX parameters:
-psm, --subMatrix [string]           Protein substitution matrix to use. Default value: BLOSUM62. Legal values: BLOSUM45, BLOSUM50, BLOSUM62, BLOSUM80, BLOSUM90
 DNA query parameters:
-fo, --forwardOnly                   Align query forward strand only. Default value: false.
-ro, --reverseOnly                   Align query reverse strand only. Default value: false.
 LCA:
-wLCA, --useWeightedLCA              Use the weighted-LCA algorithm. Default value: false.
-wLCAP, --weightedLCAPercent [number]   Set the weighted-LCA percentage of weight to cover. Default value: 80.0.
-top, --topPercent [number]          Top percent value for LCA algorithm. Default value: 10.0.
-supp, --minSupportPercent [number]   Min support value for LCA algorithm as a percent of assigned reads (0==off). Default value: 0.001.
-sup, --minSupport [number]          Min support value for LCA algorithm (overrides --minSupportPercent). Default value: 1.
-mpi, --minPercentIdentityLCA [number]   Min percent identity used by LCA algorithm. Default value: 0.0.
-mag, --magnitudes                   Reads have magnitudes (to be used in taxonomic or functional analysis). Default value: false.
 Heuristics:
-spf, --maxSeedsPerFrame [number]    Maximum number of seed matches per offset per read frame. Default value: 100.
-spr, --maxSeedsPerRef [number]      Maximum number of seed matches per read and reference. Default value: 20.
-sh, --seedShift [number]            Seed shift. Default value: 1.
 Banded alignment parameters:
-go, --gapOpen [number]              Gap open penalty. Default value: 11.
-ge, --gapExtend [number]            Gap extension penalty. Default value: 1.
-bd, --band [number]                 Band width/2 for banded alignment. Default value: 4.
 Other:
-rqcb, --replicateQueryCacheBits [number]   Bits used for caching replicate queries (size is then 2^bits). Default value: 20.
-v, --verbose                        Echo commandline options and be verbose. Default value: false.
-h, --help                           Show program usage and quit.
```

Figure 2: Summary of command-line usage of `malt-run`.

# References


[1] M. Ashburner, C. A. Ball, J. A. Blake, D. Botstein, H. Butler, J. M. Cherry, A. P. Davis, K. Dolinski, S. S. Dwight, J. T. Eppig, M. A. Harris, D. P. Hill, L. Issel-Tarver, A. Kasarskis, S. Lewis, J. C. Matese, J. E. Richardson, M. Ringwald, G. M. Rubin, and G. Sherlock. Gene ontology: tool for the unification of biology. the gene ontology consortium. *Nat Genet*, 25(1):25–29, May 2000.

[2] Stefan Burkhardt and Juha Kärkkäinen. Better filtering with gapped q-grams. *Fundamenta Informaticae*, XXIII:1001–1018, 2001.

[3] Kun-Mao Chao, William R. Pearson, and Webb Miller. Aligning two sequences within a specified diagonal band. *Computer Applications in the Biosciences*, 8(5):481–487, 1992.

[4] D. H. Huson, A. F. Auch, J. Qi, and S. C. Schuster. MEGAN analysis of metagenomic data. *Genome Res*, 17(3):377–386, March 2007.

[5] D. H. Huson, S. Mitra, N. Weber, H.-J. Ruscheweyh, and S. C. Schuster. Integrative analysis of environmental sequences using MEGAN 4. *Genome Research*, 21:1552–1560, 2011.

[6] Lucian Ilie, Silvana Ilie, Shima Khoshraftar, and Anahita Mansouri Bigvand. Seeds for effective oligonucleotide design. *BMC Genomics*, 12:280, 2011.

[7] M. Kanehisa and S. Goto. KEGG: Kyoto encyclopedia of genes and genomes. *Nucleic Acids Res*, 28(1):27–30, Jan 2000.

[8] Bin Ma, John Tromp, and Ming Li. PatternHunter: faster and more sensitive homology search. *Bioinformatics*, 18(3):440–445, 2002.

[9] Lynne Reed Murphy, Anders Wallqvist, and Ronald M. Levy. Simplified amino acid alphabets for protein fold recognition and implications for folding. *Protein Engineering*, 13:149–152(4), 2000.

[10] Z. Ning, A. J. Cox, and J. C. Mullikin. SSAHA: a fast search method for large DNA databases. *Genome Res*, 11(10):1725–1729, 2001.

[11] Ross Overbeek, Tadhg Begley, Ralph M Butler, Jomuna V Choudhuri, Han-Yu Chuang, Matthew Cohoon, Valérie de Crécy-Lagard, Naryttza Diaz, Terry Disz, Robert Edwards, Michael Fonstein, Ed D Frank, Svetlana Gerdes, Elizabeth M Glass, Alexander Goesmann, Andrew Hanson, Dirk Iwata-Reuyl, Roy Jensen, Neema Jamshidi, Lutz Krause, Michael Kubal, Niels Larsen, Burkhard Linke, Alice C McHardy, Folker Meyer, Heiko Neuweger, Gary Olsen, Robert Olson, Andrei Osterman, Vasiliy Portnoy, Gordon D Pusch, Dmitry A Rodionov, Christian Rückert, Jason Steiner, Rick Stevens, Ines Thiele, Olga Vassieva, Yuzhen Ye, Olga Zagnitko, and Veronika Vonstein. The subsystems approach to genome annotation and its use in the project to annotate 1000 genomes. *Nucleic Acids Res*, 33(17):5691–5702, 2005.

[12] Sean Powell, Damian Szklarczyk, Kalliopi Trachana, Alexander Roth, Michael Kuhn, Jean Muller, Roland Arnold, Thomas Rattei, Ivica Letunic, Tobias Doerks, Lars Juhl Jensen, Christian von Mering, and Peer Bork. eggNOG v3.0: orthologous groups covering 1133 organisms at 41 different taxonomic ranges. *Nucleic Acids Research*, 40(Database-Issue):284–289, 2012.

[13] R. L. Tatusov, E. V. Koonin, and D. J. Lipman. A genomic perspective on protein families. *Science*, 278(5338):631–637, Oct 1997.

[14] Yongan Zhao, Haixu Tang, and Yuzhen Ye. RAPSearch2: a fast and memory-efficient protein similarity search tool for next-generation sequencing data. *Bioinformatics*, 28(1):125–126, 2012.

# Index