

User Guide for 42

Denis BAURAIN and Mick VAN VLIERBERGHE, ULiège

DRAFT - August 3rd, 2020

Contents

1	Background	2
1.1	Aim and features	2
1.2	Design principles	2
2	Functional overview	3
2.1	Orthology-controlling heuristics	3
2.1.1	Collection of queries	3
2.1.2	Preflight check of orthology relationships	3
2.1.3	Search for homologues using queries	4
2.1.4	Identification of best hits for queries	4
2.1.5	Identification of orthologues among homologues	4
2.1.6	Optimized enrichment of multigenic families	5
2.1.7	Consolidation of redundant orthologues	5
2.2	Orthologue post-processing	5
2.2.1	Family affiliation and orthologue naming	5
2.2.2	Contamination detection and handling	6
2.2.3	Alignment and MSA integration	7
2.2.4	Redundancy detection and handling	7
2.2.5	#NEW# tags	7
3	Usage	8
3.1	Installation and dependencies	8
3.2	Input and configuration files	9
3.2.1	MSAs (*.fasta)	9
3.2.2	Reference organisms (ref_orgs, ref_banks)	10
3.2.3	Query organisms (query_orgs)	11
3.2.4	Candidate organisms (orgs, banks)	11
3.2.5	Taxonomic filters (tax_filter)	12
3.3	Running 42	13
3.3.1	Assisted configuration using the wizard	13
3.3.2	Command-line options	13

1 Background

1.1 Aim and features

The aim of 42 is to add (and optionally align) sequences to a preexisting multiple sequence alignment (MSA) while controlling for orthology relationships and potentially contaminating sequences.

Sequences to add are either nucleotide transcripts resulting from transcriptome assembly or already translated protein sequences. In theory, one can also use genomic nucleotide sequences (because 42 can splice introns), but this possibility has not been extensively tested so far.

The working hypothesis of 42 is that its orthology-controlling heuristics can enrich not only MSAs of single-copy genes but also more complicated MSAs including terminally duplicated genes (in-paralogues) and/or corresponding to multigenic families featuring different out-paralogues of different ages. Preliminary tests on a broadly sampled eukaryotic data set suggest that the orthology relationships enforced by 42 are in good agreement with those inferred with *OrthoFinder* software [Emms and Kelly (2015) *Genome Biol* 16:157]. To this end, it relies on complete proteomes of reference organisms.

42 is also able to enrich MSAs resulting from the split of complex multigenic families after phylogenetic analysis. For this, it requires decoy files composed of representative sequences of unwanted out-paralogues. Such PARA files have to be provided by the user.

Regarding contamination, 42 implements a dual system of taxonomic filters (based on *NCBI Taxonomy*) allowing it to flag any new sequence for which the taxonomic affiliation is doubtful. Two main approaches are available: 42 either checks that a new sequence is most similar to (an)other sequence(s) of the expected taxon already present in the MSA (= positive filter) or that a new sequence is more similar to a sequence in the MSA than to any sequence from a set of complete proteomes that do not include the expected taxon (= TOL check decoy). While the power of the first mechanism is dependent on the taxonomic breadth of each MSA, the second approach is more widely applicable.

42 is exclusively setup through a structured text file (e.g., YAML format). Archiving of this file allows a user to document all the configuration details for a given run.

42's verbosity is configured directly on the command line. 42 can be very introspective if asked to be so. At the highest verbosity level, the numerous BLAST reports are not deleted after the run and are thus available for manual inspection (e.g., for debugging purposes).

1.2 Design principles

In a single run, 42 can process an arbitrary large number of MSAs (FASTA files specified using shell jokers on the command line). Moreover, one can search for orthologous sequences in as many organisms as wanted.

The configuration (`config`) file has two main parts: one with the options that apply globally to the run and one that lists the organisms (`orgs`) to search and their specific options, including the path (`bank_dir`) to the corresponding sequence databases (`banks` in 42's parlance). The `config` file includes a mechanism of default values (`defaults`) that apply to all organisms except when otherwise specified in individual `org` subsections (e.g., `code`).

When 42 enriches a MSA, it processes each organism in turn following the order of `org` subsections in the `config` file. Several *out-of-order* optimisations ensure that similar computations (e.g., BLAST searches) are not repeated uselessly.

2 Functional overview

2.1 Orthology-controlling heuristics

Each run of 42 must specify a set of candidate organisms `orgs` that are going to be mined for orthologues, a set of reference organisms (`ref_orgs`), for which the complete proteomes have to be available (`ref_bank_dir`, `ref_org_mapper`), and a set of query organisms (`query_orgs`), which should be represented in most MSAs to be enriched. These two latter sets of organisms do not need to be identical but certainly can. They will apply to all organisms (`orgs`) to be added to yield the new MSAs (`out_suffix`).

2.1.1 Collection of queries

For each `org`, 42 extracts all sequences belonging to the `query_orgs` in order to assemble a list of `query_seqs`. Those are used to mine `orgs` for homologs (candidate orthologues) and to generate a list of ‘validating’ orthologues out of `ref_orgs`. If a MSA does not contain any sequence fulfilling the selection criteria, 42 warns the user and falls back to selecting the longest sequence instead, which leads to a singleton `query_seqs`.

2.1.2 Preflight check of orthology relationships

To ensure that it can accurately enrich MSAs in orthologous sequences, 42 verifies that `query_seqs` and `ref_orgs` themselves satisfy its orthology criteria. This two-step process is carried out separately for each MSA.

First, an average BLASTP bit score is computed for each `ref_org` based on the individual best hits of each `query_seq` against the corresponding complete proteomes. `query_seqs` without any hit in a given `ref_org` are taken into account by contributing a value of zero to the average bit score for the `ref_org`. How exactly first hits are considered best hits is explained in “Identification of best hits for queries”.

`ref_orgs` without any hit to `query_seqs` are automatically discarded, whereas the remaining ones are ranked in descending order on the average bit score. Low-scoring `ref_orgs` can be optionally discarded by specifying a value < 1.0 for the `ref_org_mul` parameter of the `config` file. For example, assuming the user lists 10 different `ref_orgs` and set `ref_org_mul` to 0.7, at most 7 `ref_orgs` will be retained for assessing orthology relationships. This could be the result of the automatic removal of two `ref_orgs` without any hit and of an additional low-scoring one to honor the `ref_org_mul` setting.

Second, the best hits for each `ref_org` are BLASTed (BLASTP) against the complete proteomes of other `ref_orgs` to check that they indeed recover the same best hits as the `query_seqs`. If any `ref_org` fails with any of the other `ref_orgs`, a message is issued to warn the user, but 42 proceeds normally. More details about the logic behind this are available in “Identification of orthologues...”. Otherwise, the preflight check is considered successful.

2.1.3 Search for homologues using queries

Each one of the `query_seqs` is BLASTed in turn against each one of the banks for the current `org`. The exact BLAST flavour is either TBLASTN or BLASTP, depending on the sequence type of `org`'s banks. Moreover, default options of this first BLAST can be overridden by specifying key/value pairs in the subsection `homologues` under the section `blast_args` of the `config` file (e.g., low-complexity filters, E-value threshold, maximum number of hits).

The whole set of hits corresponding to all `query_seqs` is consolidated into a single list of **homologous** sequences. These sequences can be optionally trimmed to the segment really covered by the matching `query_seqs`. This behaviour is especially useful when using (complete) genome assemblies for enrichment, but also to avoid non-core regions to perturb orthology assessment and improves the reliability of the intron splicing step. It is controlled by the `trim_homologues` parameter of the `config` file. The details of this trimming step can be fine-tuned by editing the other `trim_*` parameters of the `config` file. Briefly, `trim_max_shift` corresponds to the maximum length allowed for an intron before breaking a homologous sequence into multiple (exon-bounded) subsequences, whereas `trim_extra_margin` controls how many additional nucleotides are extracted at each homologue extremities.

2.1.4 Identification of best hits for queries

Each `query_seq` is furthermore BLASTed (BLASTP) against the complete proteome of each `ref_org`. Again, BLAST options can be overridden if needed (subsection `references` under section `blast_args`). For each `query_seq`, the best hit in the `ref_org` is recorded. However, when bit scores of subsequent hits are nearly equal to the bit score of the best hit, the corresponding sequences are interpreted as closely related in-paralogues and also added to the list of **best hits**. This behaviour can be tweaked using the `bitscore_mul` parameter of the `config` file.

As a consequence, several best hits can be recorded for a single `query_seq/ref_org` pair, either because several sequences are available for the `query_org` (in-paralogues or out-paralogues in the case of a multi-genic family) or because several sequences match a single `query_seq` in the `org`'s banks (which should be co-orthologues then), or for both reasons. In contrast, if a `ref_org` has no homologue for the current MSA, 42 warns the user and drops it from the list of `ref_orgs` considered by the orthology-controlling engine.

2.1.5 Identification of orthologues among homologues

To sort out orthologous sequences from paralogous sequences, each homologue in the current `org` is BLASTed (BLASTX or BLASTP) against the complete proteome of each `ref_org` (BLAST options in subsection `orthologues` under section `blast_args`). And now, here's the heart of 42's heuristics... To be considered as an orthologue, a homologue must satisfy the following criterion for every one of the (active) `ref_orgs` without exception: its best hit in the corresponding complete proteome must be found in the original list of best hits assembled using the `query_seqs`.

It is important to note that 42 does not care about which particular `query_seq` (or `query_seqs`) recovered the homologue in the `org` nor about those that recovered the best hits in the complete proteomes of the `ref_orgs`. The only thing that matters is that *the loop is closed*. The set of homologues for which this condition holds then become the **orthologues**. If the parameter `ref_brh` of the `config` file is set to `off`, all homologues are automatically considered as orthologues (but see PARA files just below).

2.1.6 Optimized enrichment of multigenic families

For multigenic families split over multiple MSAs, one can also optionally assemble PARA files. Such a file should contain sequences representative of the other sub-families of a multigenic family, so as to help 42 to even better discriminate between orthologous and paralogous sequences. Sequences in PARA files are in FASTA format but do not need to be aligned. To be considered as an orthologue, a homologue must obtain a best hit BLAST bit score that is higher when compared to the sequences of the MSA than those of the PARA file.

For example, let us say we have a family composed of 4 subfamilies (A-D). The initial orthologous group would include the 4 types of paralogues in a single MSA. Based on a phylogenetic analysis of this MSA, we could split this orthologous group into 4 distinct MSAs (A-D). If we consider the enrichment of subfamily A (`famA.fasta`), then the sequences of the other subfamilies (B, C and D) should be used to build the PARA file (`famA.para`). Hence, any homologous sequence that would be more similar to a sequence in the PARA file (say of type B) than to any sequence (of type A) in the MSA would then be rejected as paralogous.

In some cases, PARA files might be a replacement for the main heuristics of 42. Yet, both approaches can be used jointly for maximal accuracy.

2.1.7 Consolidation of redundant orthologues

When using (highly) redundant transcriptome assemblies for MSA enrichment, some genes can be represented by a series of very similar transcripts, either partially overlapping or containing minor sequencing errors. To deal with these situations, 42 provides the parameter `merge_orthologues` in the config file, which is set to `off` by default. When enabled (with `on`), orthologues are first fed to CAP3 in an attempt to merge some of them into contigs. Successfully merged orthologues are identified by a trailing `+N` tag where `N` is the number of orthologous sequences removed in the merging process. The contig itself is named after the longest orthologous sequence composing it. The details of this merging step can be fine-tuned by editing the other `merge_*` parameters in the config file.

2.2 Orthologue post-processing

Once orthologues are identified, each one is BLASTed (BLASTX or BLASTP) against the MSA itself to recover its closest relatives (BLAST options in subsection templates under section `blast_args`).

2.2.1 Family affiliation and orthologue naming

If the most closely related sequence in the MSA belongs to a given family (e.g., `mt-`), the orthologue is affiliated to the same family, as did the original forty. This allows enriching MSAs corresponding to multigenic families. Note that only the most closely related sequence can be used to infer the orthologue's family.

The orthologue identifier is built using the org name and the accession of the corresponding sequence in the org's banks, which helps tracking down all the sequences added to a MSA by 42 (e.g., for debugging purposes). This is thus different from the original forty, in which most sequences were *contigs* having lost all connection with the nucleotide sequences in the org's banks.

2.2.2 Contamination detection and handling

42 then seeks to determine whether the orthologue is a genuine orthologue or a xenologue contaminating the org's banks. To this end, two main avenues are available: positive taxonomic filters and decoy proteomes sampled across the diversity of the Tree of Life (TOL). While both approaches are in principle combinable, 42 currently implements them as exclusive options. Positive filters are enabled by adding `tax_filter` parameters in the config file, whereas decoy proteomes further require enabling the `tol_check` option.

To use positive filters, 42 must infer the taxonomy of the orthologue by analysing the identifiers of the closest sequences in the MSA. How this is precisely carried out depends on several parameters in the config file. For simplicity, the user can choose between two predefined modes to set these parameters in bulk: `best-hit` and `megan-like`.

In `best-hit` mode, only the most closely related preexisting sequence is used to infer the orthologue's taxonomy, whereas in `<megan-like>` mode, several sequences are considered and a Last-Common-Ancestor (LCA) inference is performed on them. The latter mode often yields more reliable taxonomic affiliations, but at the obvious expense of accuracy (i.e., only at the genus or family level instead of species level).

If the inferred taxonomy for the orthologue satisfies the taxonomic filter, the orthologue is simply added to the MSA. Otherwise, it is tagged as a contaminant (c#). When an orthologue is tagged as a contaminant, the binomial of the organism at the origin of the taxonomic affiliation (or a higher-ranking taxon in case of LCA inference) is further appended to its identifier (i.e., ...`Genus_species`).

Taxonomic filters are optional and require a local copy of the *NCBI Taxonomy* database (`tax_dir` parameter in the config file). It can be installed using `setup-taxdir.pl` (see "Installation..." below).

The logic behind decoy TOL proteomes is similar to the way PARA files work but with a twist. To be considered as uncontaminated, an orthologue must obtain a best hit BLAST bit score that is higher when compared to the sequences of the MSA than those of the decoy TOL proteomes. However, decoy proteomes from organisms taxonomically related to the one to which the orthologue belongs are expected to yield very good bit scores, maybe higher than any bit score from sequences already present in the MSA. To avoid rejecting a genuinely uncontaminated orthologue because of this possibility, 42 skips all decoy hits that satisfy the taxonomic filter specified for the organism being added.

Let us take an example. Imagine that our MSAs only contain sequences from Hymenopterans (e.g., wasps and ants). If we enable decoy TOL proteomes when adding some bees, contaminated orthologues corresponding to say, parasitic mites such as *Varroa destructor*, will be correctly rejected because they match some tick protein included in Ixodes decoy proteome. However, genuine bee sequences would also be wrongly rejected because they match with the *Apis* decoy proteome. To avoid this, we will use a taxonomic filter to skip Hymenoptera (or even Hexapoda) hits in decoy proteomes.

As one can see, both avenues rely on taxonomic filters. Choosing the right level of taxonomic filtration is no easy task and often requires a bit of testing. In short, the more distant potential contaminants are from organisms being added, the easier it is to find an adequate taxonomic filter (see "tax_filter" below for details).

2.2.3 Alignment and MSA integration

To integrate the orthologue into the MSA, 42 chooses the most appropriate **template(s)** for alignment among the closest relatives. As for taxonomic inference, it considers each of them in turn and stops once the coverage of the orthologue cannot be significantly improved. This allows 42 to select a slightly less related sequence as a template provided it aligns with a longer part of the orthologue. By how much exactly coverage has to be improved for a close sequence to be retained as a template can be fine-tuned with the `coverage_mul` parameter of the `config` file.

Then comes the alignment itself. With nucleotide banks, both BLAST and exonerate aligners are available, whereas only BLAST can be used with protein banks. The preferred aligner can be specified using the `aligner` parameter of the `config` file.

The BLAST aligner has been much improved with respect to the aligner of the original forty. It extracts all the HSPs for the selected template(s) from the XML BLAST report and uses them as guides for integrating the orthologue fragments into the MSA. Then, once all fragments have been integrated for all candidate organisms, it merges them into a single contiguous sequence per orthologue. When fragments overlap, the merger gives precedence to the fragments corresponding to the highest-scoring templates and HSPs.

When the new exonerate aligner is preferred, only the longest selected template is used. In most cases, the orthologue can be aligned as a single large fragment. If not, 42 emits different types of warnings depending on the exact issue. In worst cases (e.g., exonerate crashing), the orthologue cannot be integrated, often due to structural rearrangements between the orthologue and the template. To avoid discarding the orthologue in such cases, one can enable BLAST as a fall-back for exonerate failures by setting the `aligner` parameter to `exoblast`.

Aligned orthologues are integrated into the MSA all together at the end of the file but in the following arrangement: first by family, then by candidate organism and then by accession. Contaminants are interspersed with genuine orthologues but can be easily identified thanks to their tag (`c#`).

2.2.4 Redundancy detection and handling

Independently of the aligner, 42 never integrates twice the same sequence for a given organism, even if obtained from multiple orthologues. Further, it filters out subsequences included in sequences from the same organism that are either already present in the MSA or that are listed in the NON counterpart of the MSA. NON files are a bit like PARA files (non-aligned sequences in FASTA format) except that matches must be exact. Finally, when a newly added orthologue includes a sequence already present in the MSA for the same organism, the latter can be either kept or removed, depending on the value of the parameter `ali_keep_lengthened_seqs` in the `config` file.

2.2.5 #NEW# tags

All newly added orthologues are tagged by a specific `#NEW#` suffix. This tag helps 42 to organize the post-processing of new sequences (e.g., fragment merging and redundancy detection) but is also useful for the end-user to identify which sequences have been added by 42. Therefore any preexisting `#NEW#` tag is cleared when 42 starts processing a MSA.

While automatic untagging can be disabled via the parameter `ali_keep_old_new_tags` in the `config` file, one should note that such preexisting new sequences are basically invisible to 42. This means that they

will not be chosen as queries for mining transcriptomes nor as templates for aligning additional new orthologues. Moreover they will not be considered for taxonomic analyses. That is why the recommended approach is to let this parameter set to its default value.

3 Usage

3.1 Installation and dependencies

42 is written in *Modern Perl* but relies on two external dependencies: *NCBI-BLAST+* and *Exonerate*. You should download and install the corresponding binaries the way you feel the most appropriate for your system.

- <ftp://ftp.ncbi.nlm.nih.gov/blast/executables/blast+/LATEST/>
- <https://www.ebi.ac.uk/about/vertebrate-genomics/software/exonerate> (use classical v2.2.0 not newer v2.4.0)

Most other dependencies can be handled automatically by using `cpanm` in a *Perlbrew* environment (<https://perlbrew.pl/>). Below are a set of commands to setup such an environment on *Ubuntu*.

```
# install development tools
$ sudo apt-get update
$ sudo apt-get install build-essential

# download the perlbrew installer...
$ wget -O - http://install.perlbrew.pl | bash

# initialize perlbrew
$ source ~/perl5/perlbrew/etc/bashrc
$ perlbrew init

# search for a recent stable version of the perl interpreter
$ perlbrew available
# install the last even version (e.g., 5.24.x, 5.26.x, 5.28.x)
# (this will take a while)
$ perlbrew install perl-5.26.2
# install cpanm (for Perl dependencies)
$ perlbrew install-cpanm

# enable the just-installed version
$ perlbrew list
$ perlbrew switch perl-5.26.2

# make perlbrew always available
# if using bash (be sure to use double >> to append)
$ echo "source ~/perl5/perlbrew/etc/bashrc" >> ~/.bashrc
# if using zsh (only the destination file changes)
```



```
$ echo "source ~/perl5/perlbrew/etc/bashrc" >> ~/.zshrc
```

Major 42 dependencies are the Bio::MUST series of modules. Install them as follows.

```
$ cpanm Bio::FastParsers
$ cpanm Bio::MUST::Core
$ cpanm Bio::MUST::Drivers
```

Since Bio::MUST modules rely on external bioinformatics programs and come with complex test suites, they sometimes raise errors during installation. If you encounter any such error, consider enabling `--force` and/or `--notest` options of `cpanm`.

```
$ cpanm --force Bio::MUST::Drivers
```

Install 42 itself. All remaining dependencies can also be taken care of by `cpanm`.

```
$ cpanm Bio::MUST::Apps::FortyTwo
```

Finally install a local mirror of the *NCBI Taxonomy*. It will be used by 42 to taxonomically affiliate inferred orthologous sequences.

```
$ setup-taxdir.pl --taxdir=taxdump/
```

3.2 Input and configuration files

To help with the configuration of the numerous parameters of the software, we designed a config file generator: `yaml-generator-42.pl`. When run with the `--wizard` option, it will guide you through the configuration by prompting for all required parameters (pressing ENTER selects the default value). At the end of process, it will produce a YAML config file named `config-$out_suffix.yaml` and a file (`build-$out_suffix.sh`) providing the command to reproduce the exact same configuration without using the wizard.

3.2.1 MSAs (*.fasta)

42 native file format for MSAs is known as the ALI format. It is very similar to the well-known FASTA format except for a few differences: (1) sequences must appear on a single (long) line; (2) gaps are encoded as asterisk characters (*) instead of dashes (-) and any whitespace is interpreted as missing character states; (3) sequence identifiers accept a single whitespace between genus and species (more on this just below); and (4) comment lines (starting with the hashtag character #) are allowed. Although 42 can read and write FASTA files transparently, its ALI roots sometimes play tricks to the user.

This is especially true for sequence identifiers. Basically, each identifier has to hold the organism name (org) followed by a separator (@) and by a protein/gene accession number. The organism name is usually the binomial name. Genus and species must be separated by a whitespace (if in ALI format) or underscore character (_ if in FASTA format). In addition, strain name and/or NCBI taxon id are also allowed after the species name but each preceded by an underscore character (_). If both are used in the sequence identifier, the taxon id has to come last. Finally, all sequence identifiers must be unique within each MSA. See examples below:

```
# Genus species@protacc
```

```

>Arabidopsis thaliana@AAL15244
# Genus species_taxonid@protacc
>Arabidopsis thaliana_3702@AA044026
# Genus species_subspecies_taxonid@protacc
>Arabidopsis lyrata_lyrata_81972@EFH60692
# Genus species_taxonid@protacc
>Archaeoglobus fulgidus_2234@WP_048095550
# Genus species_strain_taxonid@protacc
>Archaeoglobus fulgidus_DSM4304_224325@AAB90113
# Genus species_strain_taxonid@protacc
>archaeon 13_1_20CM_2_54_9_1805008@OLE74253

```

3.2.2 Reference organisms (ref_orgs, ref_banks)

The reference proteome set must be described in the config file. Firstly, each of the reference proteomes must be in FASTA format in order to be formatted as a BLAST database with the `makeblastdb` command.

```

$ for REFORG in *.faa; do makeblastdb -in $REFORG -dbtype prot \
    -out `basename $REFORG .faa` -parse_seqids; done

```

Then, `yaml-generator-42.pl` will read a file describing the reference proteome set (`ref_org_mapper.idm`). This file is composed of two columns separated by a tabulation character (`\t`) with the first column being the organism name (`ref_org`) and the second being the database basename (`ref_bank`).

If your banks are like this:

```

$ ls Arabidopsis_thaliana_3702_bank.*

```

```

Arabidopsis_thaliana_3702_bank.faa
Arabidopsis_thaliana_3702_bank.phr
Arabidopsis_thaliana_3702_bank.pin
Arabidopsis_thaliana_3702_bank.pog
Arabidopsis_thaliana_3702_bank.psd
Arabidopsis_thaliana_3702_bank.psi
Arabidopsis_thaliana_3702_bank.psq

```

Then the `ref_org_mapper` file should look like this:

```

Arabidopsis thaliana_3702    Arabidopsis_thaliana_3702_bank

```

If you mainly work with microbes, you may want to name your banks after the NCBI GCA/GCF accessions of the corresponding genome assemblies. In this case, you can use `fetch-tax.pl` to generate a suitable file from a list of such numbers:

```

$ head -n5 banks.idl

```

```

GCA_000008085.1
GCA_000011505.1
GCA_000012285.1

```

```
GCA_000014585.1
GCA_000019605.1
```

```
$ fetch-tax.pl --taxdir=taxdump/ --org-mapper --item-type=taxid banks.idl
```

```
$ head -n5 banks.org-idm
```

```
Nanoarchaeum equitans_GCA_000008085.1      GCA_000008085.1
Staphylococcus aureus_GCA_000011505.1      GCA_000011505.1
Sulfolobus acidocaldarius_GCA_000012285.1  GCA_000012285.1
Synechococcus sp._GCA_000014585.1          GCA_000014585.1
Korarchaeum cryptofilum_GCA_000019605.1    GCA_000019605.1
```

3.2.3 Query organisms (query_orgs)

query_orgs should be listed in a file (queries.txt) and spelled exactly as in your MSAs (excluding the FASTA-specific '>' character preceding each sequence identifier). This file will be processed by yam1-generator-42.pl to populate the config file. To easily draft a list of query_orgs, you can for example use the 10 to 20 most represented organisms across all your MSAs (prior to enrichment).

```
$ grep -h \> *.fasta | cut -f1 -d'@' | cut -c2- | sort | uniq -c | sort -rn | head -n10
```

```
22498 Danio_rerio
21071 Homo_sapiens
20722 Mus_musculus
18933 Monodelphis_domestica
18616 Loxodonta_africana
17762 Latimeria_chalumnae
17678 Canis_familiaris
17114 Xenopus_tropicalis
16665 Anolis_carolinensis
16611 Sarcophilus_harrisii
```

Note: Organism names must follow the same rules as above. This means that no underscore should appear between genus and species. 42 emits a warning when suspecting you got it wrong. However, it cannot fix this for you. When working with native ALI files, this issue does not crop up:

```
$ grep -h \> *.ali | cut -f1 -d'@' | cut -c2- | sort | uniq -c | sort -rn | head -n10
```

```
22498 Danio rerio
21071 Homo sapiens
...
```

3.2.4 Candidate organisms (orgs, banks)

The candidate organisms set must be described in the config file. Firstly, each of the candidate organism files must be in FASTA format in order to produce a BLAST database with the makeblastdb command:

```
$ for ORG in *.fna; do makeblastdb -in $ORG -dbtype nucl \
    -out `basename $ORG .fna` -parse_seqids; done
```

Within each BLAST database, sequence identifiers must be unique. 42 will use the first run of non-whitespace characters as the accession. If this first chunk is composed of multiple parts separated by pipe characters (`|`), only the last part is taken as the sequence accession (see “Orthologue naming...” above).

sequence identifier	accession
>seq37	seq37
>comp12_c0_seq1	comp12_c0_seq1
>EH093040.1 SL_SLB_01N04_T7 SLB ...	EH093040.1
>MMETSP0151_2-20130828 7_1 len=174	7_1
>gi 301500844 ref YP_003795256.1 ...	YP_003795256.1

Then, as for `ref_orgs` above, you need to produce a `bank_mapper.idm` file composed of two columns separated by a tabulation character (`\t`) with the first column being the organism name (`org`) and the second being the database basename (`bank`).

```
Euglena gracilis    Euglena_bank
```

Note: Again, organism names must follow the same rules as above!

3.2.5 Taxonomic filters (`tax_filter`)

Note: this section deals with an advanced use of 42. It can be skipped if you do not plan to check added sequences for potential contaminations (see “Contamination detection...” for theoretical background).

Bio::MUST modules provide quite sophisticated taxonomic filters. Hence, in `tax_filter` syntax, wanted taxa are to be prefixed by a `+` symbol, whereas unwanted taxa are to be prefixed by a `-` symbol. Wanted and unwanted taxa are linked by logical ORs. Here are a few examples and their meaning:

[+Poaceae]	# any grass
[-eudicotyledons]	# anything but a dicotyledon flower
[+Amphibia, +Amniota]	# any amphibian or amniote
[+Bacteria, -Cyanobacteria]	# any non-cyanobacterial bacterium

In principle, 42 can also use such filters, but the generator currently supports only the plain `tax_filter` syntax shown in the first example (`+Poaceae`). Yet, it can assist you in finding the adequate taxa to specify based on the organisms you add.

For this to work, you must define at which taxonomic level(s) you want to set the filter. When specifying several levels (`--levels` option), the script will try to check for the next level in case one is missing. You can put as many levels as you want separated by a comma (`,`, no whitespace character) when using the `--wizard` option and by a whitespace character () as a command line argument. Another possibility is to choose manually from the NCBI lineage for those that fail (in this case use the `--choose_tax_filter=1` argument). If you want to select manually for each candidate organisme set the argument `--choose_tax_filter=2`.

Alternatively, you can define a custom taxonomic filter for each organism by adding a third column to the `bank_mapper.idm` file.

3.3 Running 42

3.3.1 Assisted configuration using the wizard

Now that you are done preparing files, let's run the wizard!

```
$ yml-generator-42.pl --wizard
```

Using the `--wizard` option enables an interactive mode where you will be asked to enter each parameter in the terminal.

Note: Pressing the ENTER key selects the default value encoded in 42.

Two `run_mode` are available: `metagenomic` or `phylogenomic`. The `phylogenomic` mode is designed to enrich MSAs with orthologues for subsequent phylogenomic analysis. In contrast, the `metagenomic` mode is designed to estimate the contamination level of transcriptomic data using reference ribosomal protein MSAs. The latter mode does not modify the MSAs but instead produces one taxonomic report per MSA listing the lineage of each identified orthologous sequence. When not specified, `run_mode` internally defaults to `phylogenomic`.

Note: the `phylogenomic` mode also produces taxonomic reports but deprived of taxonomic affiliations for the purpose of `one-on-one.pl` (not currently distributed on *CPAN*).

The wizard does its best to assist you in building your `config` file. In particular, it scans the directories you specify for relevant files. Hence, to identify the `banks` and `ref_banks` files, it looks for files ending with a specific suffix. These suffixes can be provided using the `bank_suffix` and `ref_bank_suffix` options, respectively. If your banks are built from protein sequences, use `.psq`; otherwise, for nucleotide sequences, use `.nsq`.

Because of this scanning behavior, it is better to prepare your files directly on the computer on which you plan to run 42. If you try to prepare your `config` file locally (for subsequent upload on a remote computer), it is very likely that the wizard complains about some directories not being found.

3.3.2 Command-line options

Since the configuration (`config`) file specifies all the details, running 42 boils down to a simple command:

```
$ forty-two.pl --config=config.yaml *.fasta
```

By default, 42 is very terse. Yet it can be made quite verbose using the corresponding `--verbosity` option. If you need all the debugging information, select level 6. In any case, it is useful to redirect the `STDERR` stream to a log file for post-run analysis.

```
$ forty-two.pl --config=config.yaml --verbosity=3 *.fasta 2> 42.log
```

42 supports multithreading by allowing parallel enrichment of multiple MSAs. This is controlled by the `--threads` command line option. MSAs will be arranged in an internal queue and processed in parallel using the specified number of threads. As long as there remain more MSAs to enrich than that number,

42 will makes efficient use of the CPU cores. Obviously, there is no speed gain in specifying more threads than MSAs to process.

```
$ forty-two.pl --config=config.yaml --threads=20 *.fasta
```

Unfortunately, the current parallel implementation scheme leads to completely scrambled log files. There is thus no point to ask for a high verbosity level.