

User Guide for 42

Denis BAURAIN [denis.baurain@ulg.ac.be]

Version 0.153130 / Nov 21, 2015

Contents

1	Aim and features	1
2	Design principles	2
3	Functional overview	2
3.1	Orthology-controlling heuristics	2
3.1.1	Collection of queries	2
3.1.2	Preflight check of orthology relationships	4
3.1.3	Search for homologues using queries	4
3.1.4	Identification of best hits for queries	4
3.1.5	Identification of orthologues among homologues	5
3.2	Orthologue post-processing	5
3.2.1	Family affiliation and orthologue naming	5
3.2.2	Contamination detection and handling	5
3.2.3	Alignment and MSA integration	6
3.2.4	Redundancy detection and handling	6
4	Annexes	7
4.1	Command line interface	7
4.2	Annotated YAML config file	9
4.3	Alternative simple YAML config file	14

1 Aim and features

The aim of 42 is to add (and to 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 (not yet demonstrated) 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].

Regarding contaminations, 42 implements a system of taxonomic filters (based on *NCBI Taxonomy*) allowing it to flag (or to reject) any new sequence for which the taxonomic affiliation is doubtful. However, the power of this mechanism is currently dependent on the taxonomic breadth of each MSA.

42 is exclusively setup through a structured text file (e.g., *YAML* format). Storage of this file allows a user to document all the configuration details for a given run. An example of a complete `config` file is available in Annotated *YAML config* file.

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).

2 Design principles

In a single run, 42 can process an arbitrary large number of MSAs (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., `bank_type`, `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.

3 Functional overview

A graphical overview of 42's pipeline is available at next page (Figure 1).

3.1 Orthology-controlling heuristics

Each run of 42 must specify 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 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`).

3.1.1 Collection of queries

For each `org`, 42 extracts *all* sequences belonging to the `query_orgs` in order to assemble a list of `query_seqs`. If a sequence is flagged as a contamination (`c#`) or as a diverging sequence (`d#`), it is automatically excluded. Eventually, 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`.

3.1.2 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_mul_org` 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_mul_org` 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 among homologues. Otherwise, the preflight check is considered successful.

3.1.3 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 useful to avoid non-core regions to perturb orthology assessment. It is controlled by the `seq_trimming` parameter of the `config` file.

3.1.4 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.

3.1.5 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 at all 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 `brh_mode` of the `config` file is set to `disabled`, all homologues are automatically considered as orthologues.

3.2 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`).

3.2.1 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`.

3.2.2 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, it infers the orthologue's taxonomy by analysing the identifiers of the five closest sequences in the MSA. More precisely, it considers each of them in turn and stops as soon as one of them can be reliably affiliated to a *NCBI Taxonomy* entry.

If the taxon corresponding to the entry satisfies the taxonomic filter (`tax_filter` parameters in the `config` file), the orthologue is added to the MSA. Otherwise, it is either removed or added but tagged as a contamination (`c#`), depending on the value of the `tf_action` parameter. When an orthologue is tagged as a contamination, the binomial of the organism at the origin of the taxonomic inference is appended to its identifier (i.e., ...`Genus_species`).

Taxonomic filters are optional and, if used, they require a local copy of the *NCBI Taxonomy* database (`tax_dir` parameter in the `config` file). It can be installed using `setup-taxdir.pl`. Taxonomic identification based on

organism names is always possible and can be made less strict for binomial names that are not yet available as a *NCBI Taxonomy* entry. This is controlled by the `tf_mode` parameter of the `config` file.

3.2.3 Alignment and MSA integration

To integrate the orthologue into the MSA, 42 chooses the most appropriate **template(s)** for alignment among the five 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.

Experimental — If the `patch_mode` parameter of the `config` file is set to `on`, close sequences belonging to the same `org` as the orthologue to be added cannot be selected as templates. This is to give new orthologues a chance to align better than these pre-existing sequences.

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 is the same as in 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's fragments into the MSA. Accessions in identifiers are modified so as to end with the rank of the template and the rank of the HSP (i.e., `XXXXXXXX.Ht.h`).

When the new `exonerate` aligner is preferred, only the longest selected template is used. The model successfully used by `exonerate` is appended to the accession: `E.lc` for `protein2genome` and `E.bf` for `protein2genome:bestfit --exhaustive`.

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 and has to be discarded. To avoid this, one can enable `BLAST` as a fall-back for `exonerate` failures by setting the `aligner` parameter to `exoblast`.

Aligned orthologues (possibly fragmented due to multiple HSPs) are integrated into the MSA all at once but in the following arrangement: first by family, then by ascending position in the MSA, and finally by descending length. This is similar to what was done by the original `forty` but could be improved in the future, for example by physically regrouping paralogues that belong to the same family.

3.2.4 Redundancy detection and handling

Independently of the aligner, 42 never integrates twice the same fragment for a given organism, even if obtained from multiple orthologues. Further, it filters out fragments 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. When an orthologue fragment 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 `ls_action` in the `config` file.

4 Annexes

4.1 Command line interface

NAME

forty-two.pl - The Answer to the Ultimate Question of Phylogenomics

VERSION

This document refers to forty-two.pl version 0.153130

USAGE

forty-two.pl <infiles> --config=<file> [optional arguments]

REQUIRED ARGUMENTS

<infiles>

Path to input ALI files [repeatable argument].

forty-two should not be called in a shell loop. If so it will run very slowly, especially when using tax filters (because loading the NCBI Taxonomy database is quite long). Use shell jokers instead:

```
forty-two.pl --config=config.yaml rpl*.ali rps*.ali
```

--config=<file>

Path to the configuration file specifying the run details.

In principle, several configuration file formats are available: XML, JSON, YAML. However, forty-two was designed with YAML in mind. See the 'test' directory of the distribution for annotated examples of YAML files.

OPTIONAL ARGUMENTS

--verbosity=<level>

Verbosity level for logging to STDERR [default: 0]. Available levels range from 0 to 6. Level 6 corresponds to debugging mode.

--version

--usage

--help

--man

Print the usual program information

AUTHOR

Denis BAURAIN <denis.baurain@ulg.ac.be>

COPYRIGHT AND LICENSE

This software is copyright (c) 2013 by University of Liege / Unit of

Eukaryotic Phylogenomics / Denis BAURAIN.

This is free software; you can redistribute it and/or modify it under the same terms as the Perl 5 programming language system itself.

4.2 Annotated YAML config file

```
# ===Path to dir holding transcript BLAST databases===
bank_dir: test

# ===Path to dir holding complete proteome BLAST databases===
ref_bank_dir: test

# ===Path to dir holding NCBI Taxonomy database===
# Only required when specifying 'tax_filter' below.
# It can be installed using setup-taxdir.pl.
tax_dir: ~/Documents/Perl/Bio-MUST-Core/test/taxdump

# ===Basenames of complete proteome BLAST databases (keyed by org name)===
# You can list as many databases as needed here.
# Only those specified as 'ref_orgs' below will actually be used for BRH.
ref_org_mapper:
  Homo sapiens: Homo_sapiens.GRCh37.70.pep.all.fa
  Arabidopsis thaliana: Athaliana_167_protein.fa

# ===Orgs from where to select BLAST queries===
# Depending on availability at least one query by family and by org will be
# picked for the 'homologues' and 'references' BLAST rounds.
query_orgs:
  - Homo sapiens
  - Arabidopsis thaliana

# ===Orgs to be used for BRH checks===
# To be considered as an orthologue, a candidate seq must be in transitive BRH
# for all listed orgs (and not for only one of them).
# Listing more orgs thus increases the stringency of the BRH check. Note that
# 'ref_orgs' do not need to match 'query_orgs'.
ref_orgs:
  - Homo sapiens
  - Arabidopsis thaliana

# ===Optional args for each BLAST round===
# Any valid command-line option can be specified (see NCBI BLAST+ docs).
# Note the hyphens (-) before option names (departing from API consistency).
# -query, -db, -out, -outfmt, -max_target_seqs, -db_gencode, -query_gencode
# will be ignored as they are directly handled by forty-two itself.
blast_args:
  # TBLASTN vs banks
  homologues:
    -evaluate: 1e-10
    -seg: yes
    -num_threads: 4
  #
  -max_target_seqs: 1
```

```

# BLASTP vs ref banks (for transitive BRH ; actually two steps)
references:
    -evalue: 1e-10
# BLASTX vs ref banks (for transitive BRH)
orthologues:
    -evalue: 1e-10
# BLASTX vs ALI (for tax filters and alignment)
templates:
    -evalue: 1e-10
    -seg: no

# ===Step(s) where to apply seq trimming===
# Currently, only one value is available: 'homologues'. In the future, a value
# 'queries' will be implemented too. Since multiple values are allowed, they
# must be specified as a list.
# If 'homologues' is specified, each candidate seq is first trimmed to the max
# range covered by the queries that retrieved it. This should help discarding
# non-homologous extensions that might be part of a fine transcript. When not
# specified, 'seq_trimming' internally defaults to no value.
# seq_trimming:
#     - homologues

# ===BRH mode for assessing orthology===
# Currently, two values are available: 'strict' and 'disabled'.
# In 'strict' mode, a candidate seq must be in BRH with all reference
# proteomes to be considered as an orthologous seq. In contrast, all candidate
# seqs are considered as orthologous seqs when BRH is disabled.
# When not specified, 'brh_mode' internally defaults to 'strict'.
# To limit the number of candidate seqs, use the '-max_target_seqs' option of
# the BLAST executable(s) at the homologues step.
brh_mode: strict
# brh_mode: disabled

# ===Fraction of ref_orgs to really use when assessing orthology===
# This parameter introduces some flexibility when using reference proteomes.
# If set to a fractional value (below 1), only the best proteomes will be
# considered during BRHs. The best proteomes are those against which the
# queries have the highest average scores. This helps discarding ref_orgs that
# might hinder orthology assessment because they lack the orthologous gene(s).
# When not specified, 'ref_org_mul' internally defaults to 1.0, which is the
# strictest mode where all reference proteomes are used during BRHs.
ref_org_mul: 1.0

# ===Bit Score reduction tolerated when including non-1st hits among best hits===
# This parameter applies when collecting best hits for queries to complete
# proteomes, so that close in-paralogues can all be included in the set of
# best hits. During BRH checks, only the very first hit for the candidate seq
# is actually tested for inclusion in this set but for all complete proteomes.

```

```

# Currently at most ten hits are considered but this might change if needed.
# When not specified 'coverage_mul' internally defaults to 1.0, which is the
# strictest mode where only equally-best hits are retained.
# bitscore_mul: 1.00
bitscore_mul: 0.99

# ===Coverage improvement required for aligning a new seq more than once===
# When not specified 'coverage_mul' internally defaults to 1.1.
# This means that if the BLAST alignment with the second template is at least
# 110% of the BLAST alignment with the first template, the new seq will be
# added twice to the ALI (under the ids *.H1.N and *.H2.N).
# Currently five templates are considered but this might change if needed.
coverage_mul: 1.1

# ===Template selection mode for aligning new seqs
# Two values are available: 'on' and 'off'.
# If set to 'on', closest relatives belonging to the same org as the new seqs
# will not be selected as templates, thus allowing the latter to align better.
# When not specified, 'patch_mode' internally defaults to 'off'.
patch_mode: off
# patch_mode: on

# ===Engine to be used for aligning new seqs===
# Four values are available: 'blast', 'exonerate', 'exoblast' and 'disabled'.
# If the alignment engine is disabled, new seqs are added 'as is' to the ALI.
# Consequently, they will be full length but not aligned to existing seqs.
# This mode is meant for protein seqs only and thus cannot be used when adding
# transcripts from nucleotide banks.
# The exonerate mode sometimes fails to align orthologous seqs due to a bug in
# the exonerate executable. This causes the new seqs to be discarded. To retry
# aligning them using BLAST instead, use the 'exoblast' mode.
# When not specified, 'aligner' internally defaults to 'blast'.
# aligner: disabled
# aligner: blast
# aligner: exonerate
aligner: exoblast

# ===Taxonomic mode for identifying contaminations===
# This arg is only meaningful when specifying 'tax_filter' below.
# Currently, two values are available: 'strict' and 'fuzzy'.
# If set to 'fuzzy', closest hits devoid of taxonid will be more aggressively
# mapped to existing NCBI Taxonomy entries (for example by ignoring species
# and relying only on genera in case of missing binomials).
# When not specified, 'tf_mode' internally defaults to 'strict'.
tf_mode: strict
# tf_action: fuzzy

# ===Action to perform when a tax_filter identifies a contamination===

```

```

# This arg is only meaningful when specifying 'tax_filter' below.
# Currently, two values are available: 'remove' and 'tag'.
# When not specified, 'tf_action' internally defaults to 'tag'.
tf_action: tag
# tf_action: remove

# ===Action to perform when a preexisting lengthened seq is identified===
# Currently, two values are available: 'remove' and 'keep'.
# When not specified, 'ls_action' internally defaults to 'keep'.
ls_action: keep
# ls_action: remove

# ===Suffix to append to infile basenames for deriving outfile names===
# When not specified 'outsuffix' internally defaults to '-42'.
# Use a bare 'out_suffix:' to reuse the ALI name and to preserve the original
# file by appending a .bak extension to its name.
out_suffix: -my-42-tax-exo

# ===Default args applying to all orgs unless otherwise specified===
# Some of these args can be thus specified on a per-org basis below if needed.
# This especially makes sense for 'code' (but not only).
defaults:
    # ===Seq type of transcript BLAST databases===
    # Two values are available: 'nucl' and 'prot'.
    # When not specified 'bank_type' internally defaults to 'nucl'.
    bank_type: nucl
    # ===Genetic code for translated BLAST searches===
    # When not specified 'code' internally defaults to 1 (standard).
    # See ftp://ftp.ncbi.nih.gov/entrez/misc/data/gc.prt for other codes.
    code: 1

# ===Org-specific args===
# The only mandatory args are 'org' and 'banks'. All other args are taken from
# the 'defaults:' section described above.
# This part can be concatenated on a per-run basis to the previous part, which
# would be the same for several runs. In the future, forty-two might support
# two different configuration files to reflect this conceptual distinction.
orgs:
    # ===Org name as to be added in the ALI===
    # You can use either 'Genus species', 'Genus species_strain' or the newer
    # 'Genus species_taxonid' back-compatible base ids of Bio::MUST.
    # NOTE THAT FORTY-TWO REQUIRES PERFECT NAME MATCHING FOR IDENTIFYING ORGS.
    # It will thus never drop a part of the name (even not the strain) to try
    # to match a closely related org. This is needed to allow it to deal with
    # bacteria en micro-eukaryotes where strains can be quite distinct.
    - org: Asbestopluma hypogea
      # - org: Asbestopluma hypogea_68561      (using Bio::MUST base ids)
    # ===Basenames of transcript BLAST databases===

```

```

# Seq ids are assumed to be unique across all databases. 42 will not crash
# if this assumption is violated but results may become less reliable.
banks:
  - Asbestopluma_hypogea
# ===Specs of the taxonomic filter aimed at flagging contaminations===
# When specified the closest hit in ALI must belong to one of the +taxa
# ... and must not belong to any of the -taxa.
# If it is not the case the new seq is flagged as a contamination.
# +taxa defaults to 'cellular organisms' while -taxa defaults to nothing.
# tax_filter is optional.
tax_filter: [ +Porifera, -Calcarea ]

- org: Oscarella carmela
# - org: Oscarella carmela_386100 (using Bio::MUST base ids)
banks:
  - Oscarella_carmela
tax_filter: [ +Porifera, -Calcarea ]

- org: Oscarella sp._sn2011
# - org: Oscarella sp._1080451 (using Bio::MUST base ids)
banks:
  - Oscarella_sp_SN2011
tax_filter: [ +Porifera, -Calcarea ]

- org: Urticina eques
# - org: Urticina eques_417072 (using Bio::MUST base ids)
banks:
  - Urticina_eques
tax_filter: [ +Anthozoa ]

```

4.3 Alternative simple YAML config file

```
# ===Path to dir holding transcript BLAST databases===
bank_dir: test

# ===Orgs from where to select BLAST queries===
# Depending on availability at least one query by family and by org will be
# picked for the 'homologues' and 'references' BLAST rounds.
query_orgs:
  - Homo sapiens
  - Arabidopsis thaliana

# ===Optional args for each BLAST round===
# Any valid command-line option can be specified (see NCBI BLAST+ docs).
# Note the hyphens (-) before option names (departing from API consistency).
# -query, -db, -out, -outfmt, -max_target_seqs, -db_gencode, -query_gencode
# will be ignored as they are directly handled by forty-two itself.
blast_args:
  # TBLASTN vs banks
  homologues:
    -evalue: 1e-10
    -seg: yes
    -num_threads: 4
    -max_target_seqs: 1
  # BLASTX vs ALI (for tax filters and alignment)
  templates:
    -evalue: 1e-10
    -seg: no

# ===BRH mode for assessing orthology===
# Currently, two values are available: 'strict' and 'disabled'.
# In 'strict' mode, a candidate seq must be in BRH with all reference
# proteomes to be considered as an orthologous seq. In contrast, all candidate
# seqs are considered as orthologous seqs when BRH is disabled.
# When not specified, 'brh_mode' internally defaults to 'strict'.
# To limit the number of candidate seqs, use the '-max_target_seqs' option of
# the BLAST executable(s) at the homologues step.
brh_mode: disabled

# ===Suffix to append to infile basenames for deriving outfile names===
# When not specified 'outsuffix' internally defaults to '-42'.
# Use a bare 'out_suffix:' to reuse the ALI name and to preserve the original
# file by appending a .bak extension to its name.
out_suffix: -my-42-simple

# ===Default args applying to all orgs unless otherwise specified===
# Some of these args can be thus specified on a per-org basis below if needed.
# This especially makes sense for 'code' (but not only).
```

```

defaults:
    # ===Seq type of transcript BLAST databases===
    # Two values are available: 'nucl' and 'prot'.
    # When not specified 'bank_type' internally defaults to 'nucl'.
    bank_type: prot

# ===Org-specific args===
# The only mandatory args are 'org' and 'banks'. All other args are taken from
# the 'defaults:' section described above.
# This part can be concatenated on a per-run basis to the previous part, which
# would be the same for several runs. In the future, forty-two might support
# two different configuration files to reflect this conceptual distinction.
orgs:
    # ===Org name as to be added in the ALI===
    # You can use either 'Genus species', 'Genus species_strain' or the newer
    # 'Genus species_taxonid' back-compatible base ids of Bio::MUST.
    # NOTE THAT FORTY-TWO REQUIRES PERFECT NAME MATCHING FOR IDENTIFYING ORGS.
    # It will thus never drop a part of the name (even not the strain) to try
    # to match a closely related org. This is needed to allow it to deal with
    # bacteria en micro-eukaryotes where strains can be quite distinct.
- org: Arabidopsis pseudothaliana
    # ===Basenames of transcript BLAST databases===
    # Seq ids are assumed to be unique across all databases. 42 will not crash
    # if this assumption is violated but results may become less reliable.
    banks:
        - Athaliana_167_protein.fa

- org: Homo pseudosapiens
    banks:
        - Homo_sapiens.GRCh37.70.pep.all.fa

```