23 Oct 2020

AutoCoEv

Manual

by Petar Petrov

Prerequisites
Install CAPS with the unofficial patch (**`caps_verbose.patch`**) for verbose output, found in **`patches/`**
Install the programs that AutoCoEv drives, as well as their own dependencies.

For Slackware, these are all available at the SlackBuilds.org repository:

vCAPS	https://slackbuilds.org/repository/14.2/academic/vCAPS_coevolution/
PhyML	https://slackbuilds.org/repository/14.2/academic/PhyML/
Gblocks	https://slackbuilds.org/repository/14.2/academic/Gblocks/
MAFFT	https://slackbuilds.org/repository/14.2/academic/mafft/
MUSCLE	https://slackbuilds.org/repository/14.2/academic/muscle/
PRANK	https://slackbuilds.org/repository/14.2/academic/prank-msa/
BLAST+	https://slackbuilds.org/repository/14.2/academic/ncbi-blast+/
EMBOSS	https://slackbuilds.org/repository/14.2/academic/seqkit/
Squizz	https://slackbuilds.org/repository/14.2/academic/squizz/
TreeBeST	https://slackbuilds.org/repository/14.2/academic/treebest-ensembl/
Parallel	https://slackbuilds.org/repository/14.2/system/parallel/

Compiling CAPS from source
CAPS requires Bio++ suite (release v1.9) libraries, compiled in this order: bpp-utils (1.5.0), bpp-numcalc (1.8.0), bpp-seq (1.7.0) and bpp-phyl (1.9.0). Sources can be obtained from the suite webpage (http://biopp.univ-montp2.fr/repos/sources/).

TreeTemplateTools.h from bpp-phyl needs to be slightly modified, in order to work with CAPS, therefore a patch (**`caps_TreeTemplateTools.patch`**) is provided in **`patches/`**.

The libraries (and patch) are available at SBo, as part of the bpp1.9 “legacy” Bio++ suite, which can be safely installed along the new version of the suite:

bpp1.9-utils	https://slackbuilds.org/repository/14.2/academic/bpp1.9-utils/
bpp1.9-numcalc	https://slackbuilds.org/repository/14.2/academic/bpp1.9-numcalc/
bpp1.9-seq	https://slackbuilds.org/repository/14.2/academic/bpp1.9-seq/
bpp1.9-phyl	https://slackbuilds.org/repository/14.2/academic/bpp1.9-phyl/

Structure and settings

The AutoCoEv folder contains the following (Box 1). Configuration is done in a single file, called **`settings.conf`** (Box 2). Input files, working folder, databases paths, as well as, run-time and post-run options are configured there.

Box 1. AutoCoEv/

```
start.sh            →   The main script, that needs to be executed.
settings.conf       →   Configuration file.
proteins.tsv        →   Example list of proteins.
species.tsv         →   Example list of species.
species.tre         →   Example species tree.
pairs.tsv           →   Example list of defined pairs.
functions/          →   Folder containing functions that start.sh calls.
doc/                →   Folder containing documentation, licensing and credits.
```

Box 2. settings.conf

```
## INPUT FILES
PROTEIN="proteins.tsv" # Proteins list file
SPECIES="species.tsv"  # Species list file
EXTTREE="species.nwk"  # External species tree file
PAIRLST="pairs.tsv"    # A list of defined protein pairs (only if: PAIRINGMANNER="defined")

## REFERENCE ORGANISM AND ORTHOLOGUES
ORGANISM="10090"       # Taxid of the reference organism (e.g. Mouse)
LEVEL="32523"          # Level at which to search for orthologues (e.g. Tetrapoda)

## DATABASE AND WORKING DIRS
DTB="/var/tmp/DB10v1"  # Folder where databases are
TMP="/tmp/workingDir"  # Working folder

## THREADS UTILIZATION
THREADS="$(nproc)"     # Number of (logical) cores to use (automatically detected)

## BLAST OPTIONS
DETBLAST="yes"         # Detailed BLAST results ("yes", "no")
PIDENT="35"            # Minimum allowed idenity (%) to the reference organism
PGAPS="25"             # Maximum allowed gaps (%) to the reference organism

## MSA OPTIONS
MSAMETHOD="muscle"     # MSA method to use ("mafft-linsi", "muscle", "prank", ...)
MUSCLEOPTIONS=""       # Any additional options to pass to MUSCLE
MAFFTOPTIONS=""        # Any additional options to pass to MAFFT
PRANKOPTIONS=""        # Any additional options to pass to PRANK
PRANKGUIDE="exguide"   # Use external guide tree for PRANK ("exguide", "noguide")?
GBLOCKSOPT="-b5=h"     # Gblocks options, e.g. allowed gaps: "-b5=h" (half)

## PhyML OPTIONS
PHYMLGBLOCKS="gblocks" # Use Gblocks filtered MSA or not ("gblocks", "vanilla")?
PHYMLOPTIONS=""        # Any additional options to pass to PhyML
PHYMLGUIDE="exguide"   # Use external guide tree for PhyML ("exguide", "noguide")?
TREESROOT="rooted"     # Root the generated trees by TreeBeST? ("rooted" or "noroot")

## EXTERNAL TREE OPTIONS
ETREEROOT="noroot"     # Root the external tree after trimming? ("rooted", "noroot")

## PAIRING
PAIRINGMANNER="all"    # Pairing manner ("all" or "defined")
MINCOMMONSPCS="20"     # Minimum number of common species per protein pair
GBLOCKS="gblocks"      # Use Gblocks filtered MSA or not ("gblocks", "vanilla")?
TREESCAPS="phyml"      # Tree to use with CAPS ("phyml", "auto", "external")
INCR="1000"            # Divide folders of protein pairs into groups of e.g. 1000
```

Box 2 continues on the next page ↓

```
## CAPS RUN-TIME OPTIONS
ALPHA="0.01"           # Alpha value for threshold cut-off. Do NOT leave blank
BOOT="0.6"             # Bootstrap threshold. Do NOT leave blank
ALNS=""                # Number of alignments to simulate ("-r 100")
CONV=""                # Convergence option ("-c")
CAPSOPTIONS=""         # Any additional options to pass to CAPS

## POST-RUN OPTIONS
RESGAPS="0.8"          # Gap threshold ("0.8")
RESBOOT="0.6"          # Bootstrap threshold ("0.6")
RESIDEN="0"            # Minimum % of allowed column diversity ("0.05"). Use with care.
PVALUE="$ALPHA"        # Post run P-value cutoff, by default equals to ALPHA
```

Proteins list

The list of proteins (e.g. `PROTEIN="proteins.tsv"`) is a simple text file, containing 3 columns (Box 3). Column 1: protein UniProt identifiers; Column 2: protein names; Column 3: protein group. They should be tab separated, with Unix line endings (LF), no headers, no spaces within the columns and no empty rows, except for the bottom one. Make sure you do not have duplicates in Column 1!

The UniProt identifiers must be from the same organism (e.g. `ORGANISM="10090" for mouse`), referred later as the *reference organism*. Column 3 is useful for the network analyses, as it makes possible to easily select nodes (proteins) belonging to the same group. In case this is not necessary, just put the same identifier for all proteins (e.g. “NNN”) in Column 3, but do not leave it blank.

Box 3. proteins.tsv

```
Q9EPQ1  Tlr1    TLR
Q9QUN7  Tlr2    TLR
Q9EPW9  Tlr6    TLR
Q61696  Hspa1a  HSP
P17879  Hspa1b  HSP
```

Species list

The list of species (e.g. `SPECIES="species.tsv"`) is a simple text file, containing two columns (Box 4). They should be tab separated, with Unix line endings (LF), no headers, no spaces within the columns and no empty rows, except for the bottom one. Column 1: species taxid codes; Column 2: species name. Make sure you do not have duplicates in Column 1!

Depending on the species, an appropriate taxonomic level should be specified in **`settings.conf`** (e.g. `LEVEL="32523"` for Tetrapoda), at which orthologues will be searched.

Box 4. species.tsv

```
9031    Gallus_gallus
9595    Gorilla_gorilla
9606    Homo_sapiens
9993    Marmota_marmota
10090   Mus_musculus
```

External tree

An external tree (e.g. `EXTTREE="species.nwk"`) should be provided if to be used as a guide by PRANK and/or PhyML. The tree should be in Newick format (nwk). Make sure the species names are exactly the same as in **`species.tsv`**. A suitable place to obtain an external tree is the TimeTree knowledge-base (http://www.timetree.org/).

Pairs list (optional)

The list of defined protein pairs (e.g. `PAIRLST="pairs.tsv"`) is a simple text file, containing 2 columns (Box 5). They should be tab separated, with Unix line endings (LF), no headers, no spaces within the columns and no empty rows, except the bottom one. Column 1: protein A UniProt identifiers; Column 2: protein B UniProt identifiers.

This file is needed only if you want to define specific pairs to be searched for co-evolution. By default, AutoCoEv creates all possible pairwise combinations between the proteins and this list is not required.

Box 5. pairs.tsv

```
Q9EPQ1  Q9QUN7
Q9EPQ1  Q9EPW9
Q9EPW9  Q9QUN7
Q61696  P17879
P17879  Q9QUN7
```

Databases

Three databases from OrthoDB (https://www.orthodb.org/?page=filelist) are required. These are *all_fasta*, *gene_xrefs* and *OG2genes* (Box 6). The script will offer to automatically download them in the specified folder (e.g. `DTB="/var/tmp/DB10v1"`) and run the necessary preparations. At the moment, the databases are at version 10v1 and require 30GB of disk space when extracted:

Box 6. /var/tmp/DB10v1

```
odb10v1_all_fasta.tab.gz   8.8 GB (archive) → 17.1 GB (extracted)
odb10v1_gene_xrefs.tab.gz  1.1 GB (archive) → 7.3  GB (extracted)
odb10v1_OG2genes.tab.gz    1.2 GB (archive) → 5.6  GB (extracted)
```

Parallelization

AutoCoEv uses GNU/Parallel for the simultaneous execution of multiple processes. By default, all detected logical cores will be used, but this can be changed if you want to keep some cores free (e.g. `THREADS="6"` on an 8-core CPU). Run once the following, in order to get familiar with the bibliography information of Parallel and silence its citation notice:

```
$ parallel --citation
```

Script run

Navigate to the AutoCoEv directory and start the main script:

```
$ bash ./start.sh
```

The script will check if all required executables are in place and will ask you to verify the working directory (e.g. `TMP="/tmp/workingDir"`). The menu of AutoCoEv is simple: it presents the different steps of the pipeline, as an enumerated list of choices. Typing the corresponding number and pressing ENTER will run the respective step. In fact, the whole workflow can be carried out by simply pressing 1, 2, 3 ...

AutoCoEv will offer to run several preparations, such as databases retrieval and processing (Box 7). Once these have been set up, you can skip the preparations menu next time you run the script, by going straight to step 6.

Box 7. Initial preparations menu:

```
1) Download databases          4) Trim gene_xrefs database
2) Extract databases           5) Trim OG2genes database
3) Index FASTA database        6) [DONE AND CONTINUE]
```

1) Downloads the archived databases from OrthoDB to the specified location.
2) Extracts the downloaded databases. Make sure you have enough space.
3) Creates an index file for `odb10v1_all_fasta.tab`.
4) Extracts from the `odb10v1_gene_xrefs.tab` databases the entries of the specified and *reference organism* (e.g. `ORGANISM="10090"`), creating a sub-database.
5) Extracts from `odb10v1_OG2genes.tab`, the entries of the specified *level* (e.g. `LEVEL="32523"`), creating a sub-database.
6) Continues to the Main menu (Box 8).

The script then verifies the correct names of databases and input files, outputting an excerpt of each. If something is missing, there will be an error message. A summary of the user-specified settings will be displayed and the main menu of the workflow will be presented (Box 8).

Box 8. Main menu:

```
 1) Pair UniProt <-> OrthoDB <-> OGuniqueID
 2) Prepare orthologues list
 3) Get FASTA sequences of all orthologues
 4) Download sequences from UniProt
 5) BLAST orthologues against UniProt sequence
 6) Get FASTA sequences of the best hits
 7) [MSA] Create MSA with selected method
 8) [TRE] Prepare trees
 9) [RUN] Create pairs
10) [RUN] CAPS run
11) [RES] Inspect CAPS results
12) [RES] Generate columns stats
13) [XML] Process CAPS results
14) [Exit script]
```

1) Reads the list of proteins, matches their UniProd identifiers to the ones of OrthoDB and finds the orthologues group (OG) identifier of each. This step will create a folder **`Orthologues/`** with subfoders for each protein (Box 9). Several report files will be generated in **`tsv/`** (Box 10).

Box 9. Orthologues/

```
Q9EPQ1/     Q9QUN7/     Q9EPW9/     Q61696/     P17879/     ...
```

Box 10. tsv/

```
Summary.tsv                →   List of matched identifiers between databases
OrthoDB_Missing.tsv        →   Uniprot identifiers not found in OrthoDB
duplicates_UniProt.tsv     →   UniProt identifiers with more than one OrthoDB ID
duplicates_OrthoDB.tsv     →   OrthoDB IDs that correspond to more than one UniProd ID
Duplicates_OrthoGroup.tsv  →   Proteins belonging to the same OrthoGroup
proteinsFound.tsv          →   The entries from proteins.tsv that were found in ODB
```

2) Prepares a homologues list of each protein for the user provided specie. Check the individual protein folders in (Box 9) for details, such as species where homologues were found.
3) Creates for each protein a subfolder **`FASTA/`**, where homologue sequences are collected (Box 11), by species (taxid).

Box 11. Orthologues/Q9EPQ1/FASTA/

```
9031.fa     9595.fa     9606.fa     9993.fa     ...
```

4) Creates for each protein, a subfolder of the reference organism (e.g. `ORGANISM="10090"`), where the protein sequence is downloaded from UniProt. Any failed downloads will be reported in **`tsv/UniProt.failed`**.
5) Creates a mini BLAST database for each of the downloaded sequences from step 4 (Box 12). Then runs BLAST of all sequences collected at Step 3, against the UniProt sequence from the reference organism, downloaded at step 4. The results are in table format, but BLAST can run a second time to generate detailed output (e.g. if `DETBLAST="yes"`) with sequence alignments. Hits for each organism are stored in a new subfolder called **`BLAST/`** (Box 13).

Box 12. Orthologues/Q9EPQ1/10090/

```
Q9EPQ1.fa       Q9EPQ1.fa.phr   Q9EPQ1.fa.pog   Q9EPQ1.fa.pot   Q9EPQ1.fa.ptf
Q9EPQ1.fa.pdb   Q9EPQ1.fa.pin   Q9EPQ1.fa.pos   Q9EPQ1.fa.psq   Q9EPQ1.fa.pto
```

Box 13. Orthologues/Q9EPQ1/BLAST/

```
9031/       9595/       9606/       9993/       ...
```

6) Retrieves the sequences of the best BLAST hits from each species, that also pass the identity (e.g. `PIDENT="35"`) and gaps (e.g. `PGAPS="25"`) thresholds, specified by the user. The sequences are stored in a new folder **`BestBLASTfasta/`** (Box 14) and two new report files are placed in **`tsv/`** (box 15).

Box 14. BestBLASTfasta/

```
Q9EPQ1.fa      Q9QUN7.fa      Q9EPW9.fa      Q61696.fa      P17879.fa   ...
```

Box 15. tsv/

```
blastBestFasta.tsv     →   Summary of the sequences that passed the filter
blastBestExclude.tsv   →   Summary of the sequences that did not pass the filter
```

7) Creates MSA by the method of choice (e.g `MSAMETHOD="muscle"`) on the sequences from Step 6. PRANK can be run with a guide tree (e.g. `PRANKGUIDE="exguide"`). The generated MSA will be then processed by Gblocks to eliminate poor quality regions, creating a filtered copy of the alignment. Results are stored in folder **`MSA/`**, in a subfolder named after the MSA method used (Box 16).

Box 16. MSA/muscle/

```
Q9EPQ1.fa   Q9EPQ1.fa.gbl   Q9EPQ1.fa.gbl.htm   ...     ...     ...
```

Where:
***.fa**: alignment
***.fa.gbl**: filtered alignment
***.fa.gbl.htm**: filtered alignment in pretty format

8) Calculates phylogenetic trees by PhyML from the MSA created in Step 7 in **`Trees/phyml`**. Either the non-processed or Gblocks filtered MSA can be used (e.g. `PHYMLGBLOCKS="gblocks"`) and the external tree can be provided as a guide (e.g. `PHYMLGUIDE="exguide"`). The produced trees will be placed in a folder named after the MSA method and settings.

E.g. a **`muscle_gblocks-exguide/`** folder (Box 17) would contain trees calculated from:
- MSAs produced by MUSCLE
- MSAs filtered by Gblocks
- an external tree was used as a guide

Box 17. Trees/

```
Trees/
└── phyml
    ├── ext                      →   external tree copies trimmed for each MSA
    └── muscle_gblocks-exguide
        ├── noroot               →   produced trees will be placed here
        ├── phy                  →   PhyML working directory
        └── rooted               →   TreeBeST rooted copies of the produced trees
```

NOTE!
If you do not wish to provide trees and use the ones that CAPS will generate automatically (e.g. `TREESCAPS="auto"`), you can skip Step 9 completely!

Alternatively, you may use the external tree directly (e.g. `TREESCAPS="external"`). The tree will be trimmed for each individual MSA and placed in **`Trees/external/`**. In our experience, this approach yields a very high number of detected co-evolving sites and does not seem to be recommended. Use with care!

9) Prepares all (e.g. `PAIRINGMANNER="all"`) unique pairwise combinations between proteins, screening in the process for the number of species that both MSAs have in common. Pairs of proteins where the number of common species is below a minimum threshold set by the user (e.g. `MINCOMMONSPCS="20"`), are excluded. The script calls SeqKit to remove the “unneeded”

sequences from the MSA, and TreeBeST to trim the trees accordingly. Pairs that pass the minimum common species requirement are placed in **Pairs-all/**, while the ones that do not -- in **Pairs-all-excluded/**, under a sub-folder named after the MSA and trees conditions.

E.g. a **MSA_muscle_gblocks..PhyML_muscle_gblocks-exguide-rooted/** folder contains:
- MSAs produced by MUSCLE and filtered by Gblocks (GBLOCKS="gblocks")
- Trees were calculated by PhyML (TREESCAPS="phyml")
- PhyML used MSAs produced by MUSCLE and filtered by Gblocks
- PhyML used external tree as a guide
- the produced trees were rooted

Each protein pair is placed in its individual sub-folder (Box 18).

Box 18. Protein pair: Q9EPQ1 vs Q9QUN7

```
Q9EPQ1_vs_Q9QUN7/
├──Q9EPQ1.species          →   List of species in common
├──Q9QUN7.species          →   List of species in common
├──msa/                    →   MSA subfolder
│    ├── Q9EPQ1.fa
│    └── Q9QUN7.fa
└──tre/                    →   Trees subfolder
     ├── Q9EPQ1.tre
     └── Q9QUN7.tre
```

Alternatively, you may wish to search for co-evolution between concrete protein pairs only (e.g. PAIRINGMANNER="defined"), in stead of all possible combinations. In this case, a list of protein pairs should be provided (e.g. PAIRLST="pairs.tsv"). Pair folders names will be changed accordingly, with a "**-defined**" suffix.

10) Carries out the actual CAPS run in folder **CAPS-all/**, where several folder levels are created within, following the CAPS run settings.

E.g. **MSA_muscle_gblocks..PhyML_muscle_gblocks-exguide-rooted/Alpha0.01/** contains:
- The MSAs and trees from Step 10
- CAPS run done under with runtime Alpha set to 0.01 (ALPHA="0.01")

This ensures that CAPS runs under different conditions can be performed without the results being overwritten. The folders of protein pairs are further divided into groups (Box 19), for example 1000 pairs per folder (e.g. INCR="1000"), the maximum number being 2000. The script navigates to the first group, runs CAPS in parallel on all 1000 pairs, then moves to the next group and so on. Progress is logged in file **progress-*.txt**.

Box 19. MSA_muscle_gblocks..PhyML_muscle_gblocks-exguide-rooted/Alpha0.01/

```
0/  1000/   2000/   3000/   4000/ ...
```

11) Inspects the CAPS results and sorts them into folder **Results-all/**, creating three subfolders: **coev/**, **fail/** and **nocoev/** (Box 20). Cleanup of the results in **coev/** is performed, preparing them for step 12. Also, summaries of **coev_inter.csv** from each pair are generated.

Box 20. Results/MSA_muscle_gblocks..PhyML_muscle_gblocks-exguide-rooted/Alpha0.01/

```
coev/                      →   Pairs for which coevolution was detected
fail/                      →   Pairs where CAPS run failed.
nocoev/                    →   Pairs for which coevolution was not detected
coev_inter.tsv             →   coev_inter summary for coevolving pairs
coev_inter_all.tsv         →   coev_inter summary for all pairs

columnStatistics.csv       →   Summary of all column statistics
stats.G0.8-I0-B0.6-P0.01   →   Summary statistics of pairs that passed the filtering
```

12) Inspects the MSA columns of residues for which coevolution was detected, and determines if the correlated residues pair passes a set of criteria. These are: column gaps (e.g. `RESGAPS="0.8"`), bootstrap value (`RESBOOT="0.6"`), column identity/diversity (`RESIDEN="0"`), and P-value (`PVALUE="0.01"`). In each protein pair folder, column statistics are saved in **`columnStatistics.tsv`**, while pairs that pass or fail the filtering are exported to **`*.PASS`** and **`*.SKIP`** files, respectively.

    The file name **`stats.G0.8-I0-B0.6-P0.01`** `combines the filtering parameters used:`
    - **`G0.8`**: gaps post-run cutoff (e.g. `RESGAPS="0.8"`)
    - **`I0`**: identity/diversity (`RESIDEN="0"`)
    - **`B0.6`**: bootstrap post-run cutoff (`RESBOOT="0.6"`)
    - **`P0.01`**: P-value post-run cutoff (`PVALUE="0.01"`)

13) Writes XML file of the results summarized in **`stats.G0.8-I0-B0.6-P0.01`**. The file is called **`CAPS.G0.8-I0-B0.6-P0.01.xml`** and is ready to be analysed by Cytoskape. The UniProt identifiers are used as nodes id, while protein names are used as labels. The network has the following values as edges:

    - **Best P-value**: lowest P-value from all coevolving residue pairs between two proteins
    - **Best pair bootstrap**: the bootstrap value of the coevolving residue pair with lowest P-value
    - **Mean P-value**: mean P-value of all pairs that passed the filtering
    - **Mean bootstrap**: mean bootstrap value for all pairs that passed the filtering
    - **Number of Pairs**: total number of coevolving residue pairs that passed the filtering

14) Exits AutoCoEv