# AutoCoEv

# Manual

## 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

The AutoCoEv folder contains the following:

**AutoCoEv/**

```
start.sh          The main script, that needs to be executed.
settings.conf     Configuration file.
proteins.tsv      Example list of proteins. Follow the same structure.
species.tsv       Example list of species. Follow the same structure.
species.tre       Example species tree. Use the same format
functions/        Folder containing functions that autoCOEV.sh calls.
doc/              Folder containing documentation, licensing information and credits.
```

## Proteins list

The list of proteins (e.g. file **`proteins.tsv`**) is a simple text file, containing 3 columns. 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.

**proteins.tsv**

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

`Column 1: protein UniProt identifiers; Column 2: protein names; Column 3: protein group`

The UniProt identifiers must be from the same organism, referred later as the *reference organism*. Make sure you do not have duplicates in Column 1! The code from Column 1 will be used in the final results as nodes ID, while the names in Column 2 will be displayed as node names. 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.

## Species list

The list of species (e.g. file **`species.tsv`**) is a simple text file, containing two columns. 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.

**species.tsv**

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

`Column 1: species taxid codes; Column 2: species name.`

Depending on the species, an appropriate taxonomic node should be specified in the **`settings.conf`** file (see later), as the *level* at which orthologues will be searched. E.g.: 2759 (Eukaryota); 33208 (Metazoa); 7742 (Vertebrata); 32523 (Tetrapoda); 40674 (Mammalia).

## External tree

An external tree (e.g. **`species.nwk`**) should be provided if to be used as a guide tree by PRANK and/or PhyML. The tree should be in Newick format (nwk). A suitable place to obtain an external tree is the TimeTree knowledge-base (http://www.timetree.org/). Make sure the species names are exactly the same as in the Species list file. For example *Marmota_marmota_marmota*, should be renamed to *Marmota_marmota*. AutoCoEv will make a copy of the external tree, where the species names will be substituted by their taxid code (see later).

## Settings

Configuration of AutoCoEv is done in a single file, called **`settings.conf`**. Input files, working folder path, as well as, run-time and post-run options are configured there. Please, get familiar with the whole file before running **`start.sh`**!

**`settings.conf`**

```
## INPUT FILES
PROTEIN="proteins.tsv" # Proteins list file
SPECIES="species.tsv"  # Species list file
EXTTREE="species.nwk"  # External species tree file

## 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)

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

## 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

## OVERLAPPING SPECIES PER PROTEIN PAIR
MINCOMMONSPCS="20"     # Minimum number of common species per protein pair

## 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 oprions, 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")

## CAPS PAIRS, MSA AND TREES
GBLOCKS="gblocks"      # Use Gblocks filtered MSA or not ("gblocks", "vanilla")?
TREESCAPS="phyml"      # Tree to use with CAPS ("auto", "phyml")
INCR="1000"            # Divide folders of protein pairs into groups of e.g. 1000

## 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
```

## Databases

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

**/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)
```

## GNU/Parallel

AutoCoEv uses GNU/Parallel for the simultaneous execution of multiple processes on the number of **THREADS**. Run once the following, in order to see the bibliography information and silence the citation notice:

```
$ parallel --citation
```

## Script run

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 ... Navigate to the AutoCoEv directory and start the main script:

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

## Initial preparations menu

Upon start, the script will offer to run several preparations, such as databases retrieval. Once done, you can just skip the preparations menu next time by going straight to steps 5 and 6.

Do initial preparations:

```
1) Download databases       3) Index FASTA database    5) Convert species to taxid?
2) Extract databases        4) Trim databases          6) [DONE AND CONTINUE]
Your choice:
```

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 databases **odb10v1_gene_xrefs.tab** and **odb10v1_OG2genes.tab**, the entries of the specified *reference organism* (e.g. **ORGANISM="10090"**) and *level* (e.g. **LEVEL="32523"**).

**/var/tmp/DB10v1/**

```
odb10v1_gene_xrefs.tab → odb10v1_gene_xrefs.$ORGANISM.tab
odb10v1_OG2genes.tab   → odb10v1_OG2genes.$LEVEL.tab
```

5) Converts the species names in the provided tree to their taxid code. Do this step, if you are planning to use an external tree as a guide for PRANK and/or PhyML.

6) Continues to the **Main menu**.

**Verification and workflow main menu**
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.

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

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

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

**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 species, at a certain level (e.g. `LEVEL="32523"` for tetrapoda). Check the individual protein folders in **`Orthologues/`** for details, such as species where homologues were found.

3) Creates for each protein a subfolder **`FASTA/`**, where sequences of homologues are collected, sorted by species (taxid).

**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"` for mouse), 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.

**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
```

6) 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 with sequence alignments (e.g. if `DETBLAST="yes"` ). Hits for each organism are stored in a new subfolder called **`BLAST/`**.

**`Orthologues/Q9EPQ1/BLAST/`**

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

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

**`BestBLASTfasta/`**

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

**`tsv/`**

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

8) Creates MSA on the sequences from Step 7, by the method of choice (e.g `MSAMETHOD="muscle"`). PRANK can be used with a guide tree (e.g. `PRANKGUIDE="exguide"`), prepared by Step 5 of the *Initial preparations menu*. The generated MSA will be then processed by Gblocks to eliminate poor quality regions. This step is very fast and is done regardless of whether these filtered sequences will be used for the next steps or not. Results are stored in folder **`/tmp/AutoCoEv/MSA`**, in a subfolder named after the method used (e.g. **`muscle`**, **`mafft-linsi`**, **`prank`**). This means you can do the alignment step with different methods, and the results of each will be in their own folder.

**`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

9) Calculates phylogenetic trees by PhyML from the MSA created in Step 8. Either the non-processed or Gblocks filtered alignments can be used (e.g. `PHYMLGBLOCKS="gblocks"`). A guide tree, prepared by Step 5 of the *Initial preparations*, can be used (e.g. `PHYMLGUIDE="exguide"`). The guide tree will be first trimmed for each MSA and stored in **`/tmp/AutoCoEv/Trees/external/`**. The trees produced by PhyML can be rooted automatically by TreeBeST (e.g. `TREESROOT="rooted"`). Results will be in **`/tmp/AutoCoEv/Trees/phyml/`** in a folder named after the MSA method and settings.

E.g. a **`muscle_gblocks-exguide-rooted/`** folder would contain trees calculated from:
◦ MSAs produced by MUSCLE
◦ MSAs filtered by Gblocks
◦ an external tree was used as a guide
◦ the produced trees were rooted

PhyML works in a subfolder called **`phy/`** and the ready trees are placed in subfolder **`tre/`**.

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

10) Prepares 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 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 **/tmp/AutoCoEv/Pairs/**, while the ones that do not -- in **/tmp/AutoCoEv/exclPairs/**, under a 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 calculated by PhyML (TREESCAPS="phyml") from MSAs produced by MUSCLE and filtered by Gblocks
- PhyML used external tree as a guide
- the produced trees were rooted

Each pair folder has the following structure (e.g. Q9EPQ1 vs Q9QUN7):

```
Q9EPQ1_vs_Q9QUN7/           Pair folder
├──Q9EPQ1.species           Species in common
├──Q9QUN7.species           Species in common
├──msa/                     MSA subfolder
│    ├── Q9EPQ1.fa
│    └── Q9QUN7.fa
└──tre/                     Trees subfolder
     ├── Q9EPQ1.tre
     └── Q9QUN7.tre
```

11) Carries out the actual CAPS run in folder **/tmp/AutoCoEv/CAPS/**. The protein pairs folders are divided into groups of max 2000 pairs (e.g. INCR="1000"), that are then processed sequentially. The script navigates to the first group, runs CAPS in parallel on all 1000 pairs, then moves to the second group and so on. Progress is logged in file **/tmp/AutoCoEv/progress-*.txt**. Several folder levels are created within **CAPS/**, following the CAPS run settings.

E.g. a **MSA_muscle_gblocks..PhyML_muscle_gblocks-exguide-rooted/Alpha0.01/** folder 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 in **/tmp/AutoCoEv/CAPS/** without the results being overwritten.

12) Inspects the CAPS results and sorts them into folder **/tmp/AutoCoEv/Results/**, having the same folder levels as in Step 11. Three subfolders with results are created.

**Results/MSA_muscle_gblocks..PhyML_muscle_gblocks-exguide-rooted/Alpha0.01/**

```
coev/       Pairs for which coevolution was detected (in groups)
fail/       Pairs where CAPS run failed.
nocoev/     Pairs for which coevolution was not detected
```

Cleanup of the results in **coev/** is performed, preparing them for the next step. Also, summaries of the **coev_inter.csv** are generated.

**Results/MSA_muscle_gblocks..PhyML_muscle_gblocks-exguide-rooted/Alpha0.01/**

```
coev_inter.tsv          coev_inter summary for pairs where coevolution was detected
coev_inter_all.tsv      coev_inter summary for all pairs
```

13) 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"), minimum required 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. Two summary files are produced.

**Results/MSA_muscle_gblocks..PhyML_muscle_gblocks-exguide-rooted/Alpha0.01/**

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

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")

14) 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

15) Exits AutoCoEv