CroCo : a program to detect and remove cross contamination in assembled transcriptomes

User Manual version 1.2

Paul Simion, Khalid Belkhir, Clémentine François, Julien Veyssier, Jochen Rink, Michaël Manuel, Hervé Philippe, Max Telford

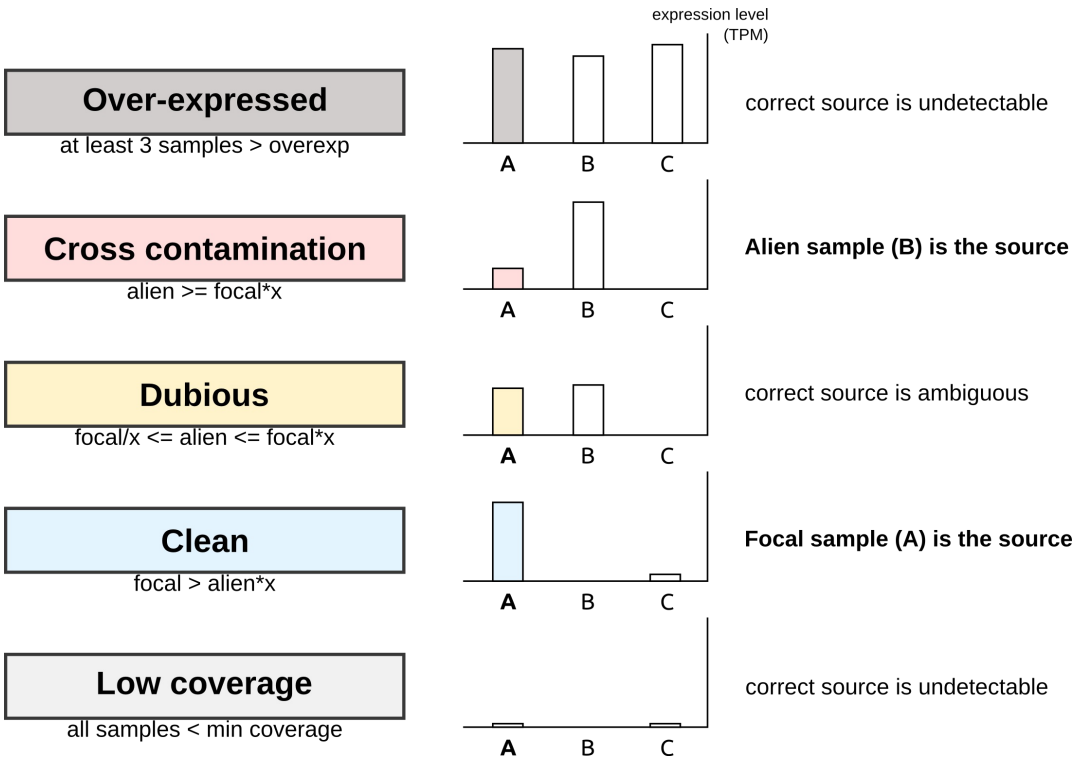
1. Institut des Sciences de l'Evolution (ISEM), UMR 5554, CNRS, IRD, EPHE, Université de Montpellier, Montpellier, France
2. Max Plank Institute of Molecular Cell Biology and Genetics, Pfotenhauerstrasse 108, 01307 Dresden, Germany
3. Sorbonne Universités, UPMC Univ Paris 06, CNRS, Evolution Paris-Seine UMR7138, Institut de Biologie Paris-Seine, Case 05, 7 quai St Bernard, 75005 Paris, France
4. Centre de Théorisation et de Modélisation de la Biodiversité, Station d'Ecologie Théorique et Expérimentale, UMR CNRS 5321, Moulis, 09200, France & Département de Biochimie, Centre Robert-Cedergren, Université de Montréal, Montréal, H3C 3J7 Québec, Canada
5. University College London, Department of Genetics, Evolution and Environment, Darwin Building, Gower Street, London WC1E 6BT, UK

Introduction

CroCo is a program to detect cross contamination events in assembled transcriptomes using sequencing reads to determine the true origin of every transcripts. Such cross contaminations can be expected if several RNA-Seq experiments were prepared during the same period at the same lab, or by the same people, or if they were processed or sequenced by the same sequencing service facility. Our approach first determines a subset of transcripts that are suspiciously similar across samples using a pairwise BLAST procedure. CroCo then combine all transcriptomes into a metatranscriptome and quantifies the "expression level" of all transcripts successively using every sample read data (e.g. several species sequenced by the same lab for a particular study) while allowing read multi-mappings. Several mapping tools implemented in CroCo can be used to estimate expression level (default is RapMap). This information is then used to categorize each transcript in the following 5 categories :

- **clean**: the transcript origin is from the focal sample.
- **cross contamination**: the transcript origin is from an alien sample of the same experiment.
- **dubious**: expression levels are too close between focal and alien samples to determine the true origin of the transcript.
- **low coverage**: expression levels are too low in all samples, thus hampering our procedure (which relies on differential expression) to confidently assign it to any category.
- **over expressed**: expression levels are very high in at least 3 samples and CroCo will not try to categorize it. Indeed, such a pattern does not correspond to expectations for cross contaminations, but often reflect highly conserved genes such as ribosomal gene, or external contamination shared by several samples (e.g. *Escherichia coli* contaminations).

CroCo outputs cross contamination statistics, transcript quantifications in all samples, optionally up to the five categorized transcriptome fasta files per sample, and optionally several graphical networks of ascertained cross contamination events as well as of dubious cases.



Important Note I - NGS data type and quality

CroCo has been designed with RNA-Seq data in mind as our approach uses differential expression to detect cross contaminations and can only work with sequencing data that bear information reagrding the quantity of molecules present in a sample. This is typically the case for transcriptomic data used to estimate transcripts expression levels (But see

below for the potential use of CroCo on genomic data). Illumina data are therefore currently the best-suited type of RNA-Seq data for CroCo, but other types of transcriptomic data (such as 454 or PacBio) can also be used in which case the user might want to adjust some of the parameters used (see Detailed options).

Additionaly, and regardless of the sequencing technology used, quality of the assembled transcriptomes used as input for CroCo is important to maximize the accuracy of our approach. For example, it is expected that transcriptome redundancy such as natural transcripts variants could potentially create false positive results since the variants might likely display different expression levels.

Important Note II - closely-related samples

CroCo's procedure make use of sequence similarity to detect potential cross contaminations events and to quantify their abundance. As a result, our approach will loose power if too closely-related samples are analysed, leading to a potential over-estimation of the number of cross contamination events, of dubious cases and of over-expressed cases. We therefore recommend not to analyse samples with very low sequence divergence levels (e.g. 1% divergence), such as in the context of population studies or for tumoral vs. normal cell comparisons.

Note, however, that even in such cases, the categorization of transcripts as clean will always be correct : you would only be discarding more transcripts than necessary.

Important Note III - CroCo and genomic data

Fundamentally, the only requirement for CroCo to be effective is that sequence coverage must be correlated to the quantity of molecules present in a given sample. Although CroCo's accuracy and efficiency has been only shown for transcriptomic data in its original publication, it is however clear that some configurations allow for the use of CroCo on genomic data. Recently, some of our colleagues (ISEM, University of Montpellier, France) successfully used CroCo to detect cross contamination events in genomic coding sequences (CDS) of various lepidopteran species (unpublished results). Treating only genomic CDS renders their experimental settings somewhat similar to that of an analysis of transcriptomic data. They detected one single significant cross contamination event which perfectly matched their prior expectation given preliminary results with that particular sample. These results confirmed that using CroCo on genomic data is at the very least sometimes possible and sound.

While more analyses will be needed in order to fully understand the limitations of CroCo when working with genomic data, here are some preliminary guidelines for interested users:

- For each scaffolds, CroCo will compute one single "quantification" estimate (i.e. the average coverage normalized by scaffolds length and by the total quantity of data mapped to the complete set of sequence). Scaffolds used should therefore correspond to genomic regions with relatively homogeneous coverage. This is why restricting the analysis to short genomic segments only (e.g. such as genes) might be necessary.
- As cross contamination events might have occur prior to PCR amplifications of the various samples under study, it is likely that PCR duplicates should be removed before using CroCo.

Citation

If you use CroCo in your work, please cite:

Simion P, Belkhir K, François C, Veyssier J, Rink JC, Manuel M, Philippe H, Telford MJ. A software tool 'CroCo' detects pervasive cross-species contamination in next generation sequencing data. BMC Biology (2018) 16:28 DOI 10.1186/s12915-018-0486-7

Table of content

Quick installation guide

Start by downloading or cloning CroCo Repository here : http://gitlab.mbb.univ-montp2.fr/mbb/CroCo.git. Then go to the section below that corresponds to your Operating System.

Linux & Mac OS X

CroCo is a BASH script written under Linux that uses BLAST+, mapping tools (e.g. *bowtie*, *RapMap*) and do not require any installation : **You can immediately use CroCo If you already have both the BLAST+ suite and the mapping tool you want to use installed on your system and present in your PATH**.

If you lack one or several of these tools, we provide a bash script (`install_dependencies.sh`) that will automatically install them for you. To do so, extract CroCo (not needed if you used `git clone`), move into its `utils/` directory and install the dependencies you want.

Here to install all dependencies on an Ubuntu OS:

```
cd CroCo_directory/utils
bash ./install_dependencies.sh --tool all --os ubuntu
```

Here to only install RapMap on a MAC OS X:

```
bash ./install_dependencies.sh --tool R --os macosx
```

We recommend to install all these dependencies automatically even if some of these softwares are already installed on your computer. Indeed, this will assure you that you are using versions of these softwares that are compatible with the current version of CroCo. Do not worry, they will be installed only within CroCo's directory and will not affect in any way the use of the softwares already installed on your system.

Now, you should have a look at the Optional requirements and then should proceed to the Installation tests to make sure that CroCo runs correctly on your system.

Windows

If you use Windows, you will have to **download and install the software Docker on your system**. A working image of CroCo, already containing all its dependencies, can then easily be build and launched through Docker. If you wish, you can also use that solution under Linux and Mac OS X in order to use CroCo through Docker.

Extract CroCo, move into its `CroCo_dockerbuild` directory and use Docker to build CroCo's image as follows :

```
cd CroCo_directory/utils/CroCo_dockerbuild
docker build -t crocodock .
```

Docker will then be able to use this image to launch CroCo (see Installation tests below to test that CroCo runs correctly on your system).

Installation & Requirements

Install CroCo dependencies

As CroCo is a bash script, and therefore do not require to be installed. However, CroCo requires several other softwares to be installed on your system.

Mandatory dependencies - BLAST+ suite :

- BLAST-2.5.0+

Mandatory dependencies - at least one mapping tool in the following list :

- Bowtie-1.2.1
- Kallisto-0.43.0
- Rapmap-0.1.0

Optional dependencies :

- R-3.1.0 (see [Optional requirements](#optional requirements) below)

The install script (`install_dependencies.sh`) let you install all these dependencies automatically. It can be used to install only one tool at a time (e.g. `--tool B`), or all in once (i.e. `--tool all`). They will be found automatically by CroCo. This script works under Ubuntu, Debian, Fedora, RedHat, CentOS and on Mac OS X.

Script usage :

```
./install_dependencies.sh --tool all|B|K|R|BL  --os
ubuntu|debian|fedora|centos|redhat|macosx
```

If you encounter problems during dependencies installation, take a look at the Troubleshooting section and at the `*_install.log` files created in the `utils/bin/` directory.

Notes for Linux users The installation script will install various libraries (or update them if already present) and will notably install `cmake` in order to be able to install *RapMap*. This requires an internet connexion.

Notes for MAC OS X users The installation script will install (or update if already present) several libraries as well as `Brew` , which will then be used to install necessary linux-like version of several functions used in CroCo and in the install_dependencies script itself (such as getopt, cmake, g++, awk, etc...). For more information on this necessary step, please see the following links :
https://www.topbug.net/blog/2013/04/14/install-and-use-gnu-command-line-tools-in-mac-os-x/
http://apple.stackexchange.com/questions/69223/how-to-replace-mac-os-x-utilities-with-gnu-core-utilities

You are also free to install CroCo dependencies manually, without using our install script. The only important thing is to make sure they are in the PATH when you launch CroCo. CroCo will first look for the tools within its own directory in `utils/bin` , then in the PATH.

Install CroCo through Docker

Using Docker will allow the creation of an image of CroCo self-containing all its dependencies. It is therefore unecessary to install any dependencies, but you are required to download and install Docker on your system (see here). This install method notably enables Windows users to use CroCo but is also available for other OS. Now, move into the `utils/CroCo_dockerbuild` directory within CroCo and then use Docker to build an image of CroCo, as follows :

```
cd utils/CroCo_dockerbuild
docker build -t crocodock .
```

You will then be able to directly launch the CroCo image you just built through Docker (see Installation tests and CroCo Usage).

Optional requirements

Using R to generate a graphical network of cross contaminations

Provided additional softwares install, CroCo can automatically outputs a dynamic visualization of the cross contamination network connecting the samples if option `--graph` is set to `yes`. This option will only work if the following tools are installed on your system :

- **R**
- R library package **visNetwork**
- R library package **igraph**

Within R, the two libraries can be installed as follows:

```
install.packages("visNetwork")
install.packages("igraph")
```

In order to successfully install the *igraph* package, R version 3.1.0 or more recent is needed.

If R and these packages are not installed, again, no worries ! You will find among CroCo's outputs a folder named `network_info` that contains every files you need (called *NODES* and *LINKS*) to build your own graphical network using other tools such as DiagrammeR or Gephi.

Adding CroCo in your PATH

It is good practice to add CroCo's location in your PATH, which allow you to use CroCo from any location in your system. This is done by using this command:

```
export PATH=$PATH:/my/complete/path/to/CroCo/directory/src
```

However, this will only temporary add CroCo's location in your PATH. To make it permanent, this previous command needs to written at the end of the file managing your PATH (such as `~/.bashrc` for Linux or `~/.profile` for MacOSX). Adding it to this file can be done manually with any text editor, or using the following command (this is an example for Ubuntu, and do not forget to use the actual location of CroCo in your system!):

```
echo 'export PATH=$PATH:/my/complete/path/to/CroCo/directory/src' >> ~/.bashrc
```

If CroCo is not in your PATH, no worries ! You will simply need to indicate the complete location of CroCo each time you use it, as follows :

```
bash /my/complete/path/to/CroCo/directory/CroCo_v0.1.sh --mode p --tool B
```

Installation tests

You can check if your installation of CroCo is working by starting an analysis on a provided small example dataset (`CroCo_dataset_test`) that conforms to input pre-requisites.

Testing on Linux & MAC OS X

Go in the main directory of CroCo and use the following commands that will first extract the dataset and then run a basic CroCo analysis:

```
tar -xvzf CroCo_dataset_test.tgz
bash src/CroCo_v0.1.sh --mode p --in CroCo_dataset_test -l 1
```

Also, to check the installation of the optional feature allowing the output of a graphical network of cross contaminations (see Optional requirements right below), try the following command with the `--graph` option set to `yes`:

```
bash src/CroCo_v0.1.sh --mode p --in CroCo_dataset_test -l 1 --graph yes
```

Testing on Windows

If you are using *Windows*, you can check your installation with the following command :

```
docker run -v /Users/path/where/data/are:/CroCoData /bin/bash
/home/CroCo_v0.1/CroCo_v0.1.sh --mode p --in CroCo_dataset_test -l 1 | tee
CroCo_test.log
```

If you used Docker to install CroCo while using *UNIX* systems (*i.e.* Ubuntu, Mac OS X), you can check your installation as follows :

```
cd utils/
docker run -t -i -P  -v /home/user/where/data/are/CroCoData:rw crocodock /bin/bash
/home/CroCo_v0.1/CroCo_v0.1.sh --mode p --in CroCo_dataset_test -l 1 | tee
CroCo_test.log
```

If you add `--graph yes` in the command above, you will also check if your R configuration allows for the automated output of a graphical network of cross contaminations.

Usage

Classic Usage

User can run the script from wherever they want. Transcriptomes and sequencing raw data must respect the input files format described in Inputs. CroCo will create a directory containing all results which will be placed within the directory containing input sequencing data. Here is the help message you get by running the script without parameter :

```
Usage :
CroCo_v1.1.sh [--cnf configFile] [--mode p|u] [--tool B|B2|K|R|S] [--fold-threshold
INT] [--minimum-coverage FLOAT] [--threads INT] [--output-prefix STR] [--output-level
1|2|3] [--graph yes|no] [--trim5 INT] [--trim3 INT] [--frag-length FLOAT] [--frag-sd
FLOAT] [--suspect-id INT] [--suspect-len INT] [--add-option STR] [--recat STR] [--
readclean yes|no]

--cnf configFile : a text filename containg a list of transcriptome assemblies to
analyze and their associated fastq reads files [short: -k]
--mode p|u :\t\t\t'p' for paired and 'u' for unpaired (default : 'p') [short: -m]
--in STR :\t\t\tName of the directory containing the input files to be analyzed
(DEFAULT : working directory) [short: -i]
--tool B|K|R :\t\t'B' for bowtie, 'K' for kallisto, 'R' for rapmap (DEFAULT : 'R')
[short: -t]
--fold-threshold FLOAT :\tValue between 1 and N (DEFAULT : 2) [short: -f]
--minimum-coverage FLOAT :\tTPM value (DEFAULT : 0.2) [short: -c]
--overexp FLOAT :\t\t\tTPM value (DEFAULT : 300) [short: -d]
--threads INT :\t\t\tNumber of threads to use (DEFAULT : 1) [short: -n]
--output-prefix STR :\t\tPrefix of output directory that will be created (DEFAULT :
empty) [short: -p]
--output-level 1|2 :\t\tSelect whether or not to output fasta files. '1' for none,
'2' for all (DEFAULT : 2) [short: -l]
--graph yes|no :\t\tProduce graphical output using R (DEFAULT : no) [short: -g]
--readclean yes|no :\t\tSelect whether or not to output fastq files devoid of reads
that mapped onto contaminant transcripts (DEFAULT : no) [short: -z]
--add-option 'STR' :\t\tThis text string will be understood as additional options for
the mapper/quantifier used (DEFAULT : empty) [short: -a]
--recat SRT :\t\t\tName of a previous CroCo output directory you wish to use to re-
categorize transcripts (DEFAULT : no) [short: -r]
--trim5 INT :\t\t\tnb bases trimmed from 5' (DEFAULT : 0) [short: -x]
--trim3 INT :\t\t\tnb bases trimmed from 3' (DEFAULT : 0) [short: -y]
--suspect-id INT :\t\tIndicate the minimum percent identity between two transcripts
to suspect a cross contamination (DEFAULT : 95) [short: -s]
--suspect-len INT :\t\tIndicate the minimum length of an alignment between two
transcripts to suspect a cross contamination (DEFAULT : 40) [short: -w]
--frag-length FLOAT :\t\tEstimated average fragment length (no default value). Only
used in specific combinations of --mode and --tool  [short: -u]
--frag-sd FLOAT :\t\tEstimated standard deviation of fragment length (no default
value). Only used in specific combinations of --mode and --tool [short: -v]

It is good practice to redirect information about each CroCo run into an output log
file using the following structure :
'2>&1 | tee log_file'

Minimal working example :
CroCo_v0.1.sh --cnf sampleconfig.txt --mode p 2>&1 | tee log_file

Exhaustive example :
CroCo_v0.1.sh --cnf configFile --mode p --in data_folder_name --tool K --fold-
threshold 2 --minimum-coverage 0.2 --overexp 300 --threads 8 --output-prefix test1_ -
-output-level 2 --graph yes --add-option '-v 0' --trim5 0 --trim3 0 --suspect-id 95 -
-suspect-len 40 --recat no --readclean no 2>&1 | tee log_file

Exhaustive example using shortcuts :
CroCo_v0.1.sh -k configFile -m p -i data_folder_name -t K -f 2 -c 0.2 -d 300 -n 8 -p
```

```
test1_ -l 2 -g yes -a '-v 0' -x 0 -y 0 -s 95 -w 40 -r no -z no 2>&1 | tee log_file

Example for re-categorizing previous CroCo results
CroCo_v0.1.sh -k configFile -i data_folder_name -r previous_CroCo_results_folder_name
-f 10 -c 0.5 -g yes 2>&1 | tee log_file
```

Usage with Docker

If you launch CroCo through Docker, you will need to add the following before the **classic usage** described above :

```
# for Windows
docker run -v /Users/path/where/data/are:/CroCoData /bin/bash [+ classic usage]
# for Linux and Mac OS X
docker run -t -i -P  -v /home/user/where/data/are:/CroCoData:rw crocodock /bin/bash
[+ classic usage]
```

Inputs

The transcriptome fasta files and their corresponding fastq files to be analyzed must be present in a given directory specified by the user with the option `--in` [short: -i]. Croco will then look for a configuration file specified with the option `--cnf` [short: -k] to get the list of files to use. The format of this configuration file is as follows:

for paired-end reads

species_A.fasta forward-reads-for-speciesA.fastq reverse-reads-for-speciesA.fastq

species_B.fasta forward-reads-for-speciesB.fastq reverse-reads-for-speciesB.fastq

for unpaired reads :

species_A.fasta reads-for-speciesA.fastq

species_B.fasta reads-for-speciesB.fastq

Things to remember when preparing input files :

- The columns of the config file must be tab-separated.
- If the raw illumina reads are gziped, the file extension must be ".gz".
- So far, gziped data are only handled by CroCo when using Kallisto (which is the default mapping tool).
- So far, it is not possible to analyze both single-end reads and paired-end reads in the same CroCo run.

Transciptome assemblies must be in fasta format. The names of the fasta files will be used internally as ID for each sample. Do not use three-characters-long ID for your sample (e.g. AAX.fasta or ED2.fasta) as this will confuse BLAST. Also, it is good practice to avoid as much as possible any special characters (e.g. `\/[]()|:;`) in sequence names within the assembled transcriptomes, as the tools used within CroCo might complain about them. Also, CroCo will temporarily cut names after the first encountered spacing character, so please be sure that the first part (i.e. the first word) of every sequence name is sufficient as a unique ID. This is to handle long sequence names resulting from some assembling softwares, or richly annotated sequences. Of course, CroCo outputs assemblies with the original full sequence names provided.

Reads fastq files should use Phred33 as quality score scheme, which is usually the case by default (If you are unsure in which quality score scheme your fastq files is encoded, see https://en.wikipedia.org/wiki/FASTQ_format#Format or use this nice python script here https://github.com/brentp/bio-playground/blob/master/reads-utils/guess-encoding.py).

Outputs

All output files will be placed within an output directory created by CroCo. Its uniq name will contain both some important parameter values used and the time and date of the run. The `--output-prefix` option allows to add user-specified text to that output directory name.

Detailed results

For every sample, one `*.all` file is created. It contains the quantification of the expression of each suspect transcript in every sample included in the analysis, as well as the expression log2fold change between the expected transcript source and the most highly expressed unexpected source. It also contains the category attributed to every transcript (last column). Note that the `utility_files_CroCo/*.all_quants` files contain the quantification of expression levels for both suspects and unsuspected transcripts.

Summary statistics

Two files will also be created by CroCo :

- a `CroCo_summary` file which is the main output file containing various statistics on transcripts categorization for each samples
- a `CroCo_profiles` file which contains all sorted log2fold change values (computed between focal and most highly expressed alien samples). This file can be used to plot the log2fold change curves and thus visualize the distribution and certainty level of cross contamination events. This could be use to potentialy refine threshold values.

Categorized transcriptomes

By default, five fasta files corresponding to the five transcript categories will be created (depending on the parameter value set for the `--output-level` option, which can be set to 1 if you are not interested in the fasta files).

Recommendations for downstream analyses :

- ONLY use the *clean* transcripts for direct qualitative analyses, such as phylogenetic inferences, or presence/absence observations. There is a slight risk of missing some correct data, but you won't miss rigour.
- use both *clean* and *low coverage* transcripts if downstream analyses can adequatly detect and circumvent potential cross contaminations (such as a refined orthology assignment step).
- if necessary, scavenge transcripts from *dubious* category on a case-by-case basis, always checking for their true origin (e.g. BLASTing them onto NCBI non-redundant nucleotidic database or mapping them onto a genome). If still in doubt, discard them.
- use *overexpressed* category on a case-by-case basis. They are strongly expressed in several samples, which means they might stem from highly conserved genes of which it might not be trivial to determine the exact taxonomical origin. They could also come from external contamination shared by several samples. Users might want to evaluate these transcripts with other tools, such as Busco. Note that if you only analyze two samples, no transcript will ever be categorized as *overexpressed* as it requires that the transcript is highly expressed in at least three samples.

Graphical networks

If the option `--graph` is set to `yes`, CroCo will use R to create 4 graphical networks allowing for a nice overview of cross contamination preferential patterns. Two of them provide a view of cross contamination patterns, and two of them provide the same view for dubious contamination cases. In these networks, the size of a node represents the number of time this sample contaminated others, the color of a node represents the percentage of this transcriptome that is cross contaminated, and the links represent the number of cross contaminating transcripts. Note that these sizes and colors are relative, and can not be compared across CroCo analyses!

This option requires a working installation of R and the installation of two R libraries : *visNetwork* and *igraph*. A first network corresponds to the exhaustive cross contamination patterns (i.e. *network_complete.html*) while a second one is an arbitrarily simplified version of the same network in which all links representing less than 2% of the biggest link are ignored (i.e. *network_simplified.html*). This simplification is useful for visualizing large networks when their is strong discrepancies between cross contamination links. The other two networks created are the conterparts of those described above for dubious cross contamination cases (i.e. *network_dubious_complete.html* and *network_dubious_simplified.html*).

To view these dynamic networks, simply open the corresponding html file with any internet browser (e.g. Firefox, Chromium). The network nodes can be selected (it then highlights them and their associated cross contaminations) and moved around as well. Here is an example command line to open the network with firefox :

```
firefox network_simplified.html &
```

Please note that if you want to move these html files (e.g. to store CroCo results elsewhere), you'll also need to move along their associated folders, respectively named *network_complete_files*, *network_simplified_files*, *network_dubious_complete_files* and *network_dubious_simplified_files* as the html files need them to display the networks!

Clean read files

If the option `--readclean` is set to `yes`, CroCo will map the reads onto their corresponding transcriptomes (using bowtie) and output cleaned fastq files in which all reads mapping onto contaminant transcripts will be excluded. Note that reads that mapped onto low coverage, dubious or over-expressed transcripts will be kept. Lastly, even if input read files were gziped, the cleaned fastq files will be written without compression. This option requires bowtie to be installed on your system.

Detailed options

--in (-i)

This option allows the user to specify the directory containing the input files (transcriptomes and raw reads). If not specified, CroCo will look for these files in the current directory.

--cnf (-k)

This indicates the configuration file containing the names of the transcriptomic assemblies to be analyzed as well as the names of their corresponding read files. Each row corresponds to one sample, the first column is the transcriptome file name, column 2 is the read file name and column 3 corresponds to the second read file name (when using paired-end reads). See details in the Inputs section above.

--mode (-m)

This parameter specify if raw data is paired-end (`p`) or single-end (unpaired, `u`). Don't forget to adjust the input file names accordingly : NAME.fastq for unpaired data, NAME.L.fastq + NAME.R.fastq for paired-end data.

IMPORTANT : if `--mode` is set to *unpaired* AND the tool used is *Kallisto*, then the options `--frag-length` and `--frag-sd` are required.

--suspect-id (-s)

Allows the user to specify the minimal percent of identity required for a BLAST hit between transcripts to consider its query transcript *suspect* (default is `95` %).

--suspect-len (-w)

Allows the user to specify the minimal alignment length required for a BLAST hit between transcripts to consider its query transcript *suspect* (default is `40` nucleotids).

--tool (-t)

This allows you to choose the mapper/quantifier to use from the following list : bowtie (`B`), kallisto (`K` , default), and rapmap (`R`).

IMPORTANT : These tools use different approaches to map and to quantify reads resulting in possibly different levels of precision and speed. The accuracy of CroCo relies on the accuracy of the tool selected.

The decision to use Kallisto as default is based both on the analyses of simulated data which suggested that using this mapping tool lead to high accuracy when comparing samples closely-related and on Kallisto capability to handle compressed read files.

--fold-threshold (-f)

This sets the fold threshold used when comparing expression level to determine if a transcript should be considered clean, dubious or a cross contamination (`2` by default). This means that by default, if a transcript is expressed `2` times more in the expected read set than in any other set, it will be considered clean. If a transcript is expressed `2` times less in the expected read set than in any other set, it will be considered a cross contamination. If a transcripts falls in between the two previous conditions, it will be considered as dubious.

Suggestion:
If you need to be absolutely certain that transcripts are rightfuly categorized as clean or cross contamination, increase the value ot the `--fold-threshold` . The side effect will be that more (possibly many) transcripts will end up categorized as dubious.

--minimum-coverage (-c)

Indicates an expression level in Transcripts Per Million (TPM). If a transcript has lower TPM value than this threshold in all samples it will then be categorized as a "low coverage" transcript.

This "low coverage" category reflects CroCo using a quantitative approach. If not enough information is given to it, it will not have anough resolution power to determine with certainty the true source of a transcript.

Its default value (i.e. `0.2` TPM) has been experimentaly set as to best balance cross contamination detection accuracy and a too high number of "low coverage" transcripts. If this value is set to `0` , no low coverage transcripts will be found, and if this value is set to an unreasonably high number such as `10000` , almost all transcripts will be categorized as low coverage transcripts. We recommend to stay somewhere within `0.1` and `10` , depending on you sequencing experiment design.

--overexp (-d)

Indicates an expression level in Transcripts Per Million (TPM). It a transcript has higher value that this threshold for at least three samples, it will then be categorized as "overexpressed". Its default value is `300` TPM.

Overexpressed transcripts will likely corresponds to highly conserved genes or external contaminations shared by multiple samples under study.

--threads (-n)

Allow the user to specify the number of parallel threads to be used by CroCo (default = `1`).

--output-prefix (-p)

This specifies a string of characters to be used as a prefiw for folder name in which all output files will be placed (empty string by default). For conveniency, we suggest to add an `_` at the end of the string you would like to use. If you want the folder name to start with `test1`, use the value `test1_` instead.

--output-level (-l)

This allows you to choose if fasta files will be created : no fasta file (`1`); all fasta files (`2` , default) thus including clean, contam, dubious, low coverage and overexpressed transcriptomes.

--graph (-g)

Setting this parameter to `yes` (default is `no`) will enable the automated rendering of a graphical cross contamination network in a dynamic html file.

IMPORTANT : this option requires a working install of *R* with the two following R libraries already installed : *visNetwork* and *igraph*.

In case you would like to see the cross contamination network of a previous CroCo analysis you ran without having activated the `--graph` option, no worries: you can use the `--recat` option and re-use your previous results with the same parameters as before, this time setting `--graph` to `yes` .

--add-option (-a)

This parameter allows experimented users to specify additional options to the mapper/quantifier tool used (default is empty). It is possible this way to change these tools default parameter values according to your type of data in order to improve mapping/quantification efficiency.

Example:
Assuming you want to increase Bowtie precision by only allowing a maximum of 1 mismatch per alignment :

```
bash CroCo_v0.1.sh --cnf configfile.txt --mode u --tool B --add-option '-v 1'
```

--recat (-r)

This option will activate the "re-categorization" switch of CroCo, and its value will indicate the location in which previous CroCo results to re-use are (default is `no`). Here is a practical example in which we re-categorize transcript using higher values than default for `--fold-threshold` and `--minimum-coverage` :

```
bash CroCo_v0.1.sh --in data_location --recat data_location/CroCo-B-id95-len40-fold2-
mincov0.2-2017_02_14-04_47 --fold-threshold 10 --minimum-coverage 2
```

Recategorization is very fast as neither BLAST nor mapping steps need to be computed anew. This `--recat` switch allows the user to quickly try several categorization parameterizations. Note that if you want to try different parameterizations for the BLAST step, then you will be required to run a complete new CroCo analysis.

--readclean (-z)

If the option `--readclean` is set to `yes` (default is `no`), CroCo will map the reads onto their corresponding transcriptomes and output cleaned and uncompressed fastq files in which all reads mapping onto contaminant transcripts will be excluded.

IMPORTANT : This option requires *bowtie* to be installed on your system.

--frag-length (-u)

This optional parameter indicates the estimated mean length of fragments that where then sequenced. This option must only be used in particular CroCo set up : using *unpaired* reads with *Kallisto*.

--frag-sd (-v)

This optional parameter indicates the estimated standard deviation of fragments length. This option must only be used in particular CroCo set up : using *unpaired* reads with *Kallisto*.

--trim5 and --trim3 (-x and -y)

These parameters indicates the number of nucleotids that will be trimmed from the reads (on either ends) prior to CroCo analyses. Default values for these two parameters are `0`.

Troubleshooting

During the installation of CroCo

RapMap installations failed
This is likely because `cmake` is not installed (or not up-to-date) on your system. Normally, the `install_dependencies.sh` script should install or update `cmake` automatically, but it can only work with an internet connexion. Install `cmake` manually and retry the two following commands:

```
bash ./install_dependencies.sh --tool S --os ubuntu
bash ./install_dependencies.sh --tool R --os ubuntu
```

---

installing R packages

installation of R packages failed
Error message : `Avis : unable to access index for repository http://_CRAN_mirror_adress`

This might simply be a problem with the CRAN mirror you used to download and install the packages (several mirrors do not work) : try to select other mirrors (and be patient as it might require several tries to find a working one).

During CroCo run

Problems with contig names in transcriptomes (1)
Error message : `Error: [blastdbcmd] contig_name: OID not found`
This message indicates that the `contig_name` is not recognised by BLAST+, probably because of the presence of special characters. Try to replace them with another character, such as an underscore.

Problems with contig names in transcriptomes (2)
Error message :

```
Warning: (1431.1) CFastaReader: Title is very long: XXXX characters (max is 1000), at line
xxxxxxxx
```

This message indicates that the sequence name before the first encountered spacing character is too long (more than 1000 characters). You need to shorten it.