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

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

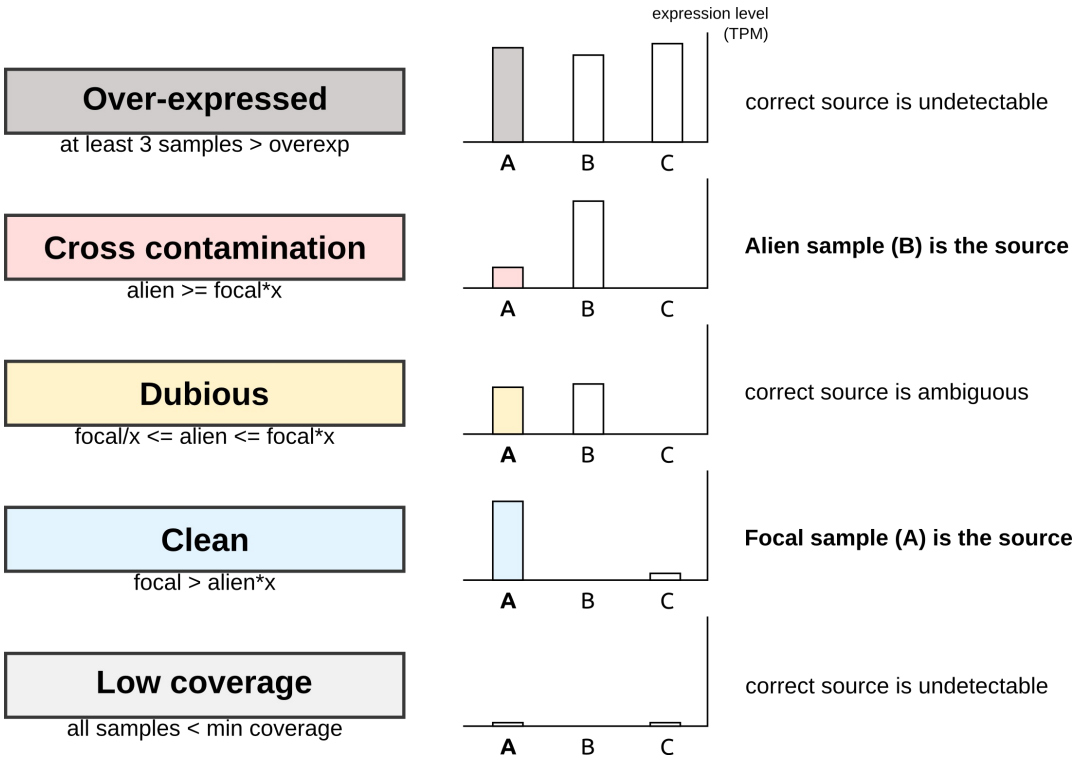
1. Institut des Sciences de l'Evolution (ISEM) - UMR 5554 - CNRS, EPHE, IRD - Université de Montpellier, Montpellier, France
2.
3. Institut Biologie Paris-Seine (IBPS) - UMR 7138 - CNRS - Université Pierre et Marie Curie, Paris, France
4.
5.
6.

Introduction

CroCo is a program to detect cross contaminations 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 company. Our approach first determines a subset of transcripts that are suspiciously similar across samples using a BLAST procedure, and then quantifies the expression level of these transcripts in all samples (e.g. several species sequenced by the same lab for a particular study). This can be done with various mapping tools implemented in CroCo (we use bowtie as default since we recommend favouring mapping accuracy above run speed). CroCo then uses that information to categorize each transcript in the following 5 categories :

- **clean**: the transcript origin is from the expected sample.
- **cross contamination**: the transcript origin is from an unexpected sample of the same experiment.
- **dubious**: expression levels are too close between expected and unexpected 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 more than 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 (such as *Escherichia coli* contaminations).

CroCo outputs cross contaminations statistics, up to the five categorized transcriptome fasta files per sample, and several graphical networks of ascertained cross contaminations as well as for dubious cases.



Important Note - NGS data type

CroCo has been designed for analyzing RNA-Seq data as our approach uses differential expression to detect cross contaminations and thus can only work with sequencing data that informs on expression level. 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 should adjust some of the parameters (see Detailed options) Fundamentaly, the only requirement for CroCo to be effective is that sequence coverage must be correlated to the quantity of molecules in

the original sample.

Important Note - closely-related samples

CroCo's procedure make use of sequence similarity to detect potential cross contaminations events. 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 contaminations, of dubious cases and of over-expressed cases. This seems to be particularly true when using fast quantification tools (such as *Salmon* or *Kallisto*) as opposed to slower mapping tools such as *Bowtie*. As a rule of thumb, we suggest not to analyse samples with less than ~4% sequence divergence. Therefore, we do not recommend using CroCo in the context of population studies.

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

Citation

If you use CroCo in your work, please cite:

xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Table of content

Quick installation guide

Start by downloading CroCo Repository here : http://xxxxxxxxxxxxxxxxxxx. 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 softwares (e.g. *bowtie*, *Salmon*) 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** (e.g. *bowtie*) **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, move into its `utils/` directory and install the dependencies you want.

Here to install all dependencies on an Ubuntu OS:

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

Here to install only the mapper Kallisto on a MAC OS X:

```
unzip XXXXXXXXXXX
cd XXXXXXXXXXX/utils
bash ./install_dependencies.sh --tool K --os macosx
```

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

We now recommend you to have a look at the Optional requirements and then to 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 :

```
unzip XXXXXXXXXXX
cd XXXXXXXXXXX/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-1.1.2
- Bowtie2-2.2.9
- Kallisto-0.43.0
- Salmon-0.6.0
- Rapmap-0.3.0

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|B2|K|L|S|H  --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 *Salmon* and *RapMap*.

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 can also install the tools 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.

Using 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. 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

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
```

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.

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_v0.1.sh [--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]

--mode p|u :\t 'p' for paired and 'u' for unpaired (default : 'p') [short: -m]
--in STR :\t Name of the directory containing the fasta files to be analyzed (DEFAULT
: working directory) [short: -i]
--tool B|B2|K|R|S|H :\t\t'B' for bowtie, 'B2' for bowtie2, 'K' for kallisto, 'S' for
salmon, 'R' for rapmap (DEFAULT : 'B') [short: -t]
--fold-threshold FLOAT :\tValue between 1 and N (DEFAULT : 2) [short: -f]
--minimum-coverage FLOAT :\tValue in TPM (DEFAULT : 0.2) [short: -c]
--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|3 :\t\tSelect the fasta files to output. '1' for none, '2' for
clean and lowcov, '3' for all (DEFAULT : 2) [short: -l]
--graph yes|no :\t\tProduce graphical output using R (DEFAULT : no) [short: -g]
--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 the previous CroCo output folder of which you wish 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 :
'| tee log_file'

Minimal working example :
CroCo_v0.1.sh --mode p | tee log_file

Exhaustive example :
CroCo_v0.1.sh --mode p --in data_folder_name --tool B --fold-threshold 2 --minimum-
coverage 0.2 --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 | tee
log_file

Exhaustive example using shortcuts :
CroCo_v0.1.sh -m p -i data_folder_name -t B -f 2 -c 0.2 -n 8 -p test1_ -l 2 -g yes -a
'-v 0' -x 0 -y 0 -s 95 -w 40 -r no | tee log_file

Example for re-categorizing previous CroCo results
CroCo_v0.1.sh -i data_folder_name -r previous_CroCo_results_folder_name -f 10 -c 0.5
-g yes | 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 transcriptomes and raw data to be analyzed must be present in a given directory indicated by the user with the option `--in` (by default CroCo will look for them in the current directory). Also, CroCo requires file names unity between transcriptomes and raw reads as follows:

for paired-end reads :
NAME.fasta (assembled transcriptome seqs)
NAME.L.fastq (raw illumina data LEFT)
NAME.R.fastq (raw illumina data RIGHT)

for unpaired reads :
NAME.fasta (assembled transcriptome seqs)
NAME.fastq (raw illumina data)

Transcriptomes should be in fasta format. It is good practice to avoid as much as possible any special characters (e.g. `\/[]()|:;`) in sequence names, 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, a `*.all` file is created. Its contains the quantification of the expression of each 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.

Summary statistics

Two files will also be created by CroCo :

- a `CroCo_summary` file which contains 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), allowing for a raw visualization of the distribution and strengh of their cross contamination

---

Categorized transcriptomes

Up to five fasta files corresponding to the five transcript categories will be created (depending on the parameter value set for the `--output-level` option).

Recommendations for downstream analyses :

- ONLY use the *clean* transcripts for direct qualitative analyses, such as phylogenetic inferences, or presence/absence observations. You might miss some 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 on NCBI non-redundant nucleotidic database is a good start). If still in doubt, discard them.
- use *overexpressed* category on a case-by-case basis, as 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 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 the nodes represents the number of time the transcriptome contaminates other samples, their color represents the percentage of the transcriptome that is cross contaminated, and the links represent the number of contaminant transcripts. 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 last two networks are the conterparts of those described above for dubious cross contamination cases.

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 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 elsewhere (e.g. to store CroCo results), you'll also need to move along their associated folders, respectively named `network_complete_files` and `network_simplified_files`, as the html files need them to display the networks.

Detailed options

--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 either *Salmon* or *Kallisto*, then the options

`--frag-length` and `--frag-sd` are required.

--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.

--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` , default), bowtie2 (`B2`) , kallisto (`K`), salmon (`S`) and rapmap (`R`).

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

The decision to use bowtie as default is based on the observation that its computationally heavier pairwise alignment strategy leads to higher accuracy when comparing samples closely-related. If analysis speed is a criterion of importance for you, we then recommend Salmon as the best trade-off between accuracy and speed.

--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 the expression level in Transcripts Per Million (TPM) under which the transcript will be categorized neither as

"clean", "contamination" nor "dubious, but will instead 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.

--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 the quantity of fasta files that will be created : none (`1`); clean and low coverage transcriptomes only (`2`, default); clean, contam, dubious, low coverage and overexpressed transcriptomes (`3`).

--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 libraries already installed : *visNetwork* and *igraph*.

In case you would like to see the cross contamination network of a previous CroCo analysis you ran with the `--graph` option set to `no`, 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.

Suggestions:
If you are analyzing 454 reads, you might want to use a less stringeant mapping to handle their more frequent sequencing errors and frameshifts. In that case you might want to use Bowtie2 with this additional option :

```
bash CroCo_v0.1.sh --mode u --tool B2 --add-option '--very-sensitive-local'
```

This time, if you want to increase Bowtie precision to only allow a maximum of 1 mismatch per alignment :

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

Lastly, if you want to increase Salmon sensitivity :

```
bash CroCo_v0.1.sh --mode u --tool S --add-option '--useVBOpt --extraSensitive'
```

--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 new complete CroCo analysis.

--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* or *Salmon*.

--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* or *Salmon*.

--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

this one should not exist anymore
Error message : `You must install make, cmake and one of clang and g++ to compile RapMap`
This is common problem of installing RapMap on MacOSX. As using RapMap is not mandatory for using CroCo, we

might ignore this problem and continue with the installation of the other dependencies, as follows:

```
bash ./install_dependencies.sh --tool S --os macosx
bash ./install_dependencies.sh --tool K --os macosx
bash ./install_dependencies.sh --tool BL --os macosx
```

---

this one should not exist anymore
Error message : `install_dependencies.sh: line 325: wget: command not found`

RapMap and Salmon 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
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.

---

this one should not exist anymore
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 spacing character is too long (more than 1000 characters). You need to shorten it.

Future Developments

Use of closely-related samples to ignore them !!

Allowing the user to provide a file containing pair of species of which cross contam will not be evaluated. This would allow

to use very closely-related samples in a broader analysis while ignoring the otherwise detected crosscontam between this specific pair. This would allow to decontaminate these two samples from their respective cross contam (that they do not necessarily share!) implementation : while categorizing *.all* results, if the best unexpected source is listed as a *paired* sample, we look at the second best unexpected source, etc...

Exporting the network in a vectorial image format (eps ? svg ? pdf ?)

Would allow a pracical use of CroCo output as a ready-to-publish graph.

allowing using reads with no quality information

In order to be able to use CroCo on sequencing data with no quality info. It will probably only be possiblefor some mapping/quantification tools.

implementing general parameter switches

It would be easier for users to be able to select a general option that will automaticaly sets multiple parameters to adequate values.

Example 1 : a user might use a `--454` switch to automatically select bowtie2 and its `--very-sensitive-local` option.

Example 2 : a user might use a `--strict` switch to automatically select bowtie, increase the value of `--fold-threshold` to 10, increase the value of `--minimum-coverage` to 2 and not allowing any mismatches when mapping with `--add-option '-v 0'`

accelerating blast step

It would be nice to properly parallelize blast step as I believe that, so far, it only uses 1 thread at a time.

add an option to output the LOG file of the run directly in the output directory

it could be done using `> logfile`, but it would be nice that the log file also ends up within the $out directory.

Cleaning sequencing raw data

CroCo has been designed to clean assembled transcriptomes. However, it would be useful to also clean-up the raw data : this would unlock improvement for all RNA-Seq downstream analyses starting with the transcriptome assembly itself.

Fast heuristic for CroCo

Adding an option allowing for a partial but fast analysis in order to **anticipate risks** of cross contaminations in very large sequencing experiments. Such a large-scale pre-screening option could allow to detect whether a full CroCo analysis is required, **before going for very heavy computational analyses**.

Additional checking of transcriptome quality

using the mappings to give info on various quality stats. Also comparing overexpressed transcripts on refined database. Overall, it is about trying to help the user a bit beyond cross-contamination detection.

TRIALS - TEMPORARY SECTION

reglage des problème de sed ??

```
- (brew install gnu-sed)
```

test macosx 10.11.3 (bureau fred)

```
/usr/bin/ruby -e "$(curl -fsSL
```

```
https://raw.githubusercontent.com/Homebrew/install/master/install)"
export PATH="$(brew --prefix coreutils)/libexec/gnubin:/usr/local/bin:$PATH"
brew tap homebrew/dupes
brew install coreutils gnu-getopt gawk grep
brew link --force gnu-getopt
cd utils
bash install_dependencies.sh --tool all --os macosx
```

test macosx 10.6.8 (Andrea vieux portable)

```
=> brew worked but git is not installed => nothing can be installed
```

test macosx 10.11.4 (ordi Fabien)

```
=> wget is not installed => NCBI must be in "local install mode"
=> problem with Salmon ! "dyld: Library not loaded: @rpath/libtbb.dylib"
https://software.intel.com/en-us/forums/intel-math-kernel-library/topic/610990
peut être que quand le script sera executable, cela fonctionnera tout seul...
(espoir)
```