CrocoBLAST:UserManual

Below you can find the **CrocoBLAST** user manual, which contains all the information you need in order to make efficient use of **CrocoBLAST**. Note that you need not install **CrocoBLAST**, as it is sufficient to download the files from the webpage and unzip them. You may run **CrocoBLAST** from the graphical interface or directly from the command line. Using the graphical interface currently requires Java 8 - but don't worry, you probably have it already. If you have any trouble, please see the Technical details.

Happy CrocoBLASTing!

I) What can I do with CrocoBLAST?

Everyone loves BLAST. We love BLAST too, which is the main reason for us to develop CrocoBLAST. CrocoBLAST makes it easier to set up and run BLAST, which is the most used local alignment algorithm. Moreover, using CrocoBLAST you can easily get your exactly identical BLAST results several times faster than what you would normally get using BLAST alone. That is achieved by efficiently parallelizing each BLAST job to all computer cores, which is not efficiently implemented by default in BLAST. If your lab produces small sequencing data sets, you are probably familiar with the NCBI BLAST online service. However, if you deal with larger data sets, you need to run your alignments locally. In this case, especially if you are producing data from Next Generation Sequencing (NGS) experiments, you have likely faced the limitations of the currently available BLAST implementations. For this purpose, CrocoBLAST was developed to let you run overnight some BLAST alignments that would normally take weeks to finish.

CrocoBLAST offers a platform for planning, running, monitoring, and managing your BLAST calculations. With **CrocoBLAST**, you will always know *how much time* it will take to run a BLAST job and the current completion status. Further, you will be able to *pause* or interrupt a BLAST job at any time to be resumed later or to *retrieve partial results*.

Another key aspect of **CrocoBLAST** is that it breaks the BLAST job into several small tasks. This enables you to run BLAST jobs with very large input files efficiently, even with minimal computational resources (say, your *desktop machine*), while ensuring that the output is *identical* to the one you would obtain if you were to run BLAST alone. As such large jobs may take a long time, simply running BLAST and waiting for them to finish may be a frustrating experience. Thus, the estimation of time provided by CrocoBLAST, together with its speed-up, may save you time and energy.

II) Terminology

There are a few basic terms you need to keep in mind when running BLAST within CrocoBLAST.

1. Input file and Database

It its essence, BLAST aligns a set of nucleotide or protein sequences (*input file*) against a set of reference sequences (*database*), reporting the score for each alignment in an effort to help you identify the closest matches to the given input sequences. In practice, this translates into taking an *input file* with many query sequences, and aligning each of the query sequences against a *database* of known sequences. Such databases are typically stored in suitable repositories such as NCBI, or may be obtained in-house.

Therefore, in order to run BLAST, you will need to specify an *input file* containing the query sequences, and a *database file* containing the reference sequences. CrocoBLAST accepts input files in FASTA and FASTQ format. BLAST uses a specific *database format* for database file. For including a *database* in CrocoBLAST, you may indicate provide a database file either in the BLAST *database format* or in FASTA or FASTQ format, which will be converted to database format before BLAST is run. Further CrocoBLAST allows you to directly download and update databases from the NCBI server.

2. BLAST program

Depending on the nature of the query and reference sequences, there are several BLAST programs you may use within CrocoBLAST:

- blastp - compares an amino acid query sequence against a protein sequence database
- blastn - compares a nucleotide query sequence against a nucleotide sequence database
- blastx - compares a nucleotide query sequence translated in all reading frames against a protein sequence database
- tblastn - compares a protein query sequence against a nucleotide sequence database dynamically translated in all reading frames
- tblastx - compares the six-frame translations of a nucleotide query sequence against the six-frame translations of a nucleotide sequence database

Therefore, in order to run BLAST, you will need to indicate which BLAST program you intend to use.

3. BLAST options

The BLAST algorithm for sequence alignment is relatively complex, and the default settings are not always optimal for identifying suitable hits in a database or retrieving only the relevant results. In case you wish to fiddle with the default BLAST settings by changing the general BLAST *options* or the *options* specific to each BLAST program. CrocoBLAST accepts almost any advanced BLAST options. Please see the NCBI web pages for a full description of accessible BLAST options. It is important to note that the current version of CrocoBLAST overrides the BLAST option "-num_threads", used for parallelization, and that the BLAST option "-outfmt" for output format can only be used with 3 different formats: the default pairwise format (code 0), the tabular format (code 6), and the comma-separated values format (code 10).

4. Job

Within CrocoBLAST, a *job* is defined by a BLAST program (with or without non-default options), a database, an input file, and an output location (folder). When created, each job receives a unique job ID that can be referenced whenever you wish to perform an operation on that job.

5. Queue

All BLAST jobs created within the CrocoBLAST environment are included in a list, which we further refer to as *queue*. The concept of *queue* is useful because it allows you to plan your work in advance and manage your jobs as you need. While CrocoBLAST only runs one *job* at a time, all your interaction with the created jobs will be via the *queue*. For example, you may pause one job to obtain the partial alignment results, and start another job while you analyze the partial results of the original job. This enables you to retain the settings and progress of the original job, which you may later choose to resume.

III) Using CrocoBLAST (Graphical Interface)

CrocoBLAST is built to help you plan your BLAST jobs and run them efficiently. CrocoBLAST operates with the concept of *queue*, which is basically a list of BLAST jobs scheduled to run. Thus, you can plan several BLAST jobs and let CrocoBLAST manage their execution for you.

All CrocoBLAST functionality is available via the command line utility and the graphical user interface. In fact, the graphical user interface does precisely what its name suggests: it provides an interface for the command line utility. In a nutshell, while you can interact with CrocoBLAST via simple commands, you may also use the interface to generate the commands or read the output of such commands.

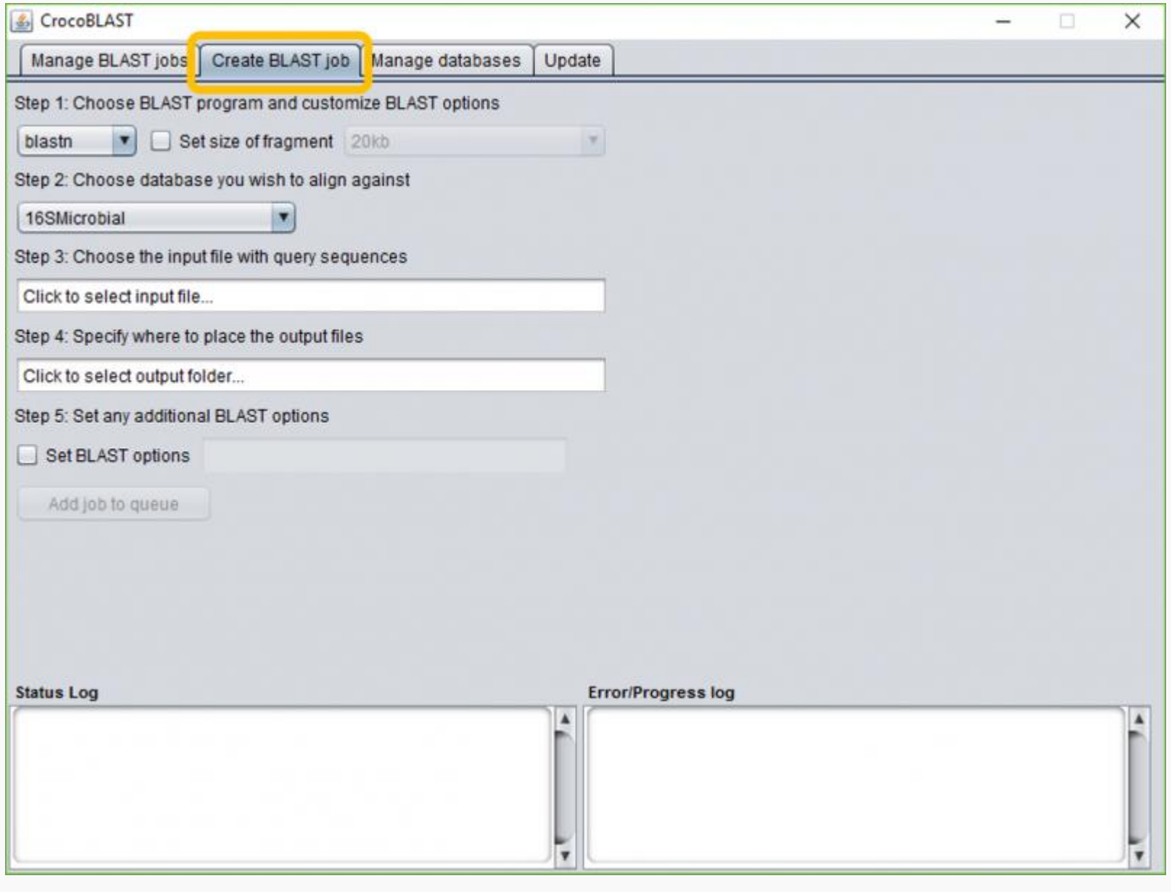
The following guide contains step-by-step instructions for using CrocoBLAST with the **graphical user interface (GUI)**.

1. Manage BLAST jobs

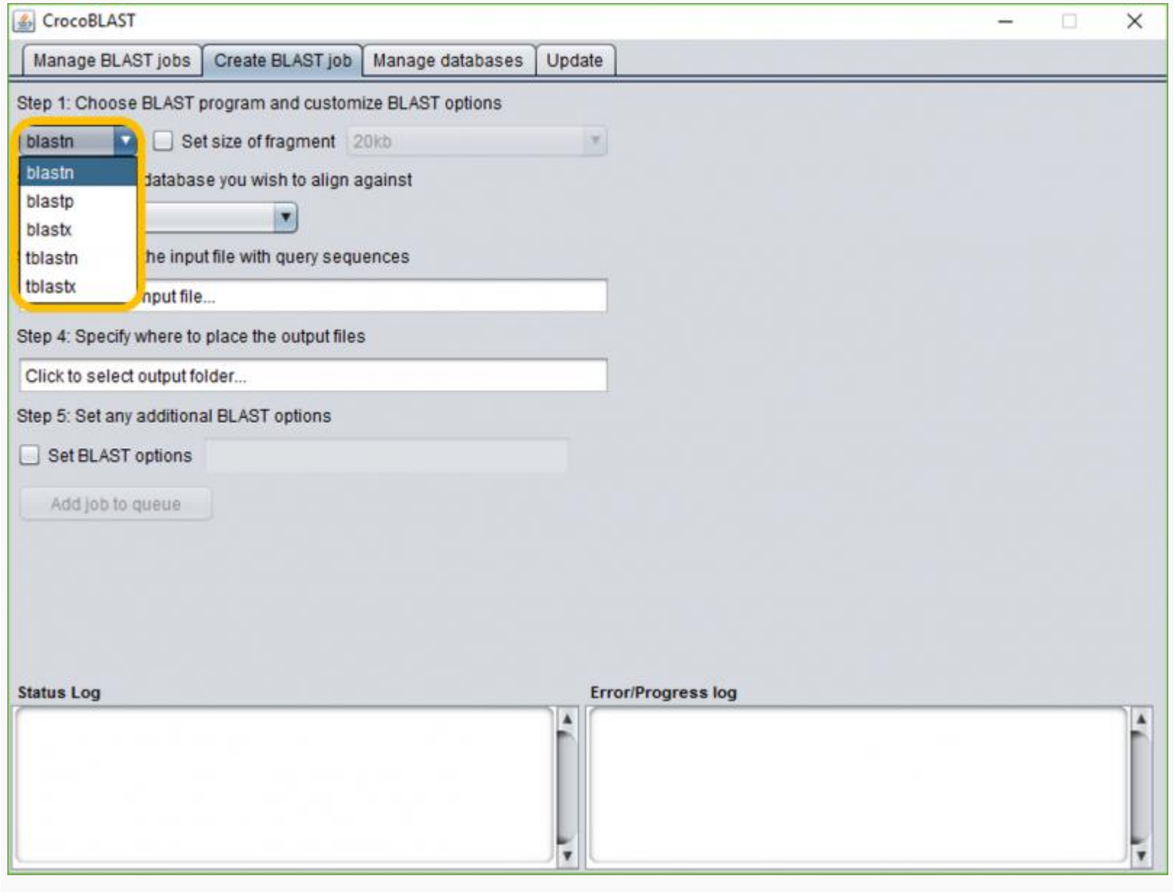
The efficiency of CrocoBLAST lies in its ability to parallelize the execution of your BLAST jobs. This is related to breaking each big calculation into smaller pieces, and then organizing the execution of the pieces. Depending on the job, CrocoBLAST may benefit from using smaller or larger smaller pieces. CrocoBLAST automatically detects the ideal size for breaking the input file and does it for you. Alternatively, the size of each input file fragment can be manually specified in the **GUI**. Further, while CrocoBLAST operates with the concept of queue, it is important to note that only one job (the first on the queue) is active at any given time.

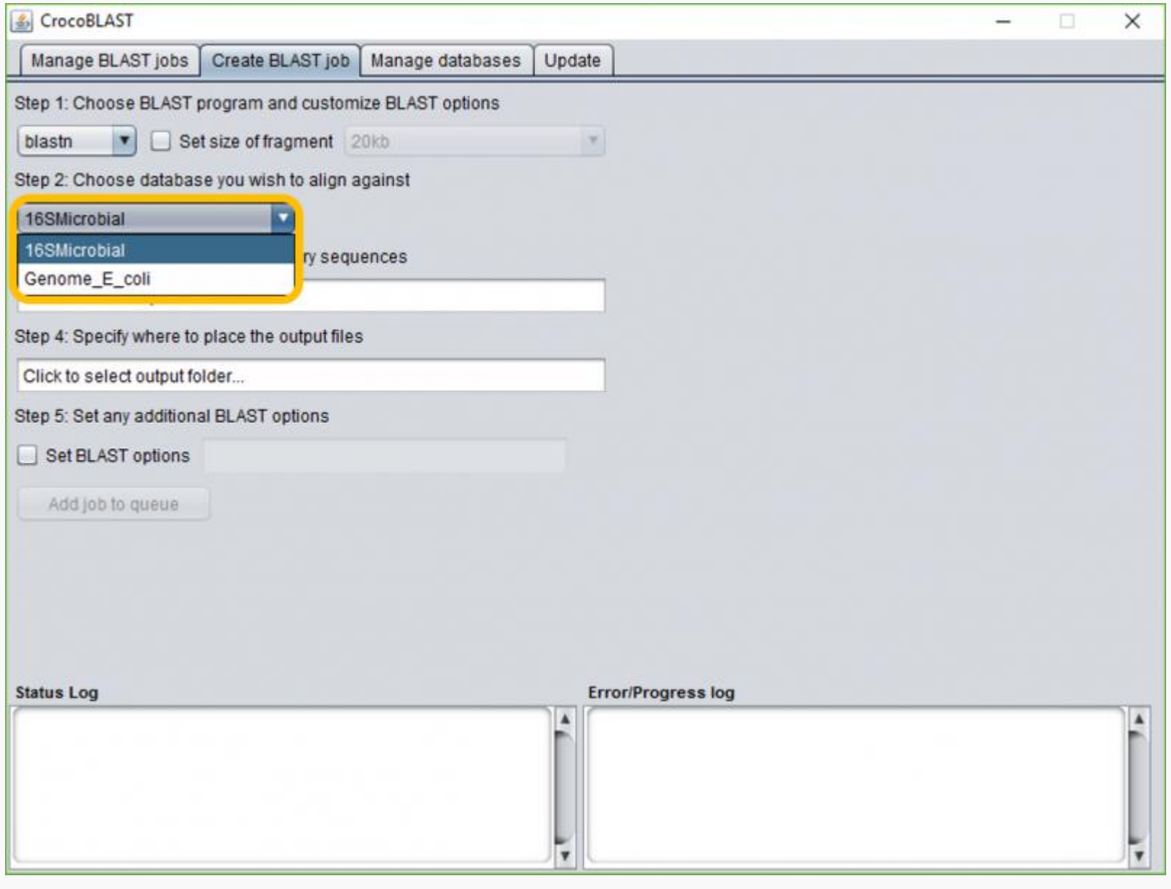
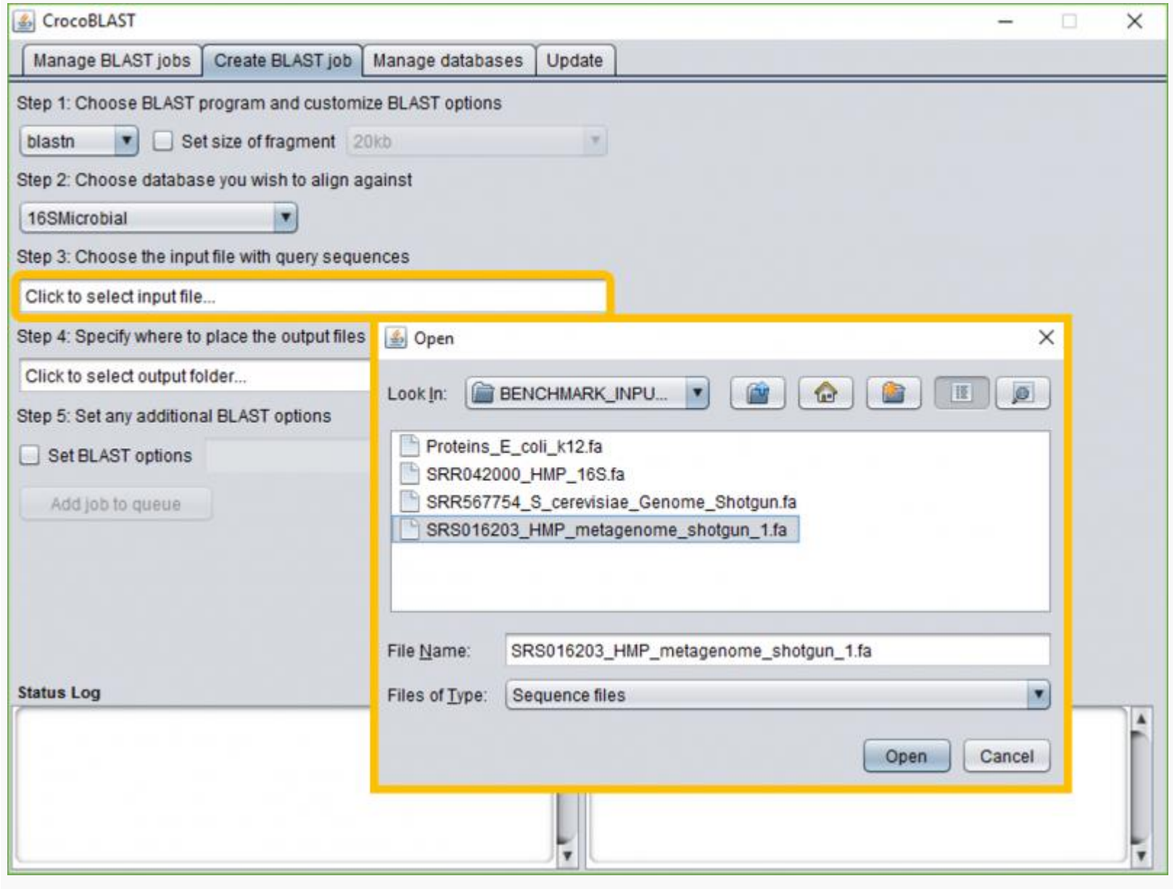
1.1. Creating BLAST jobs

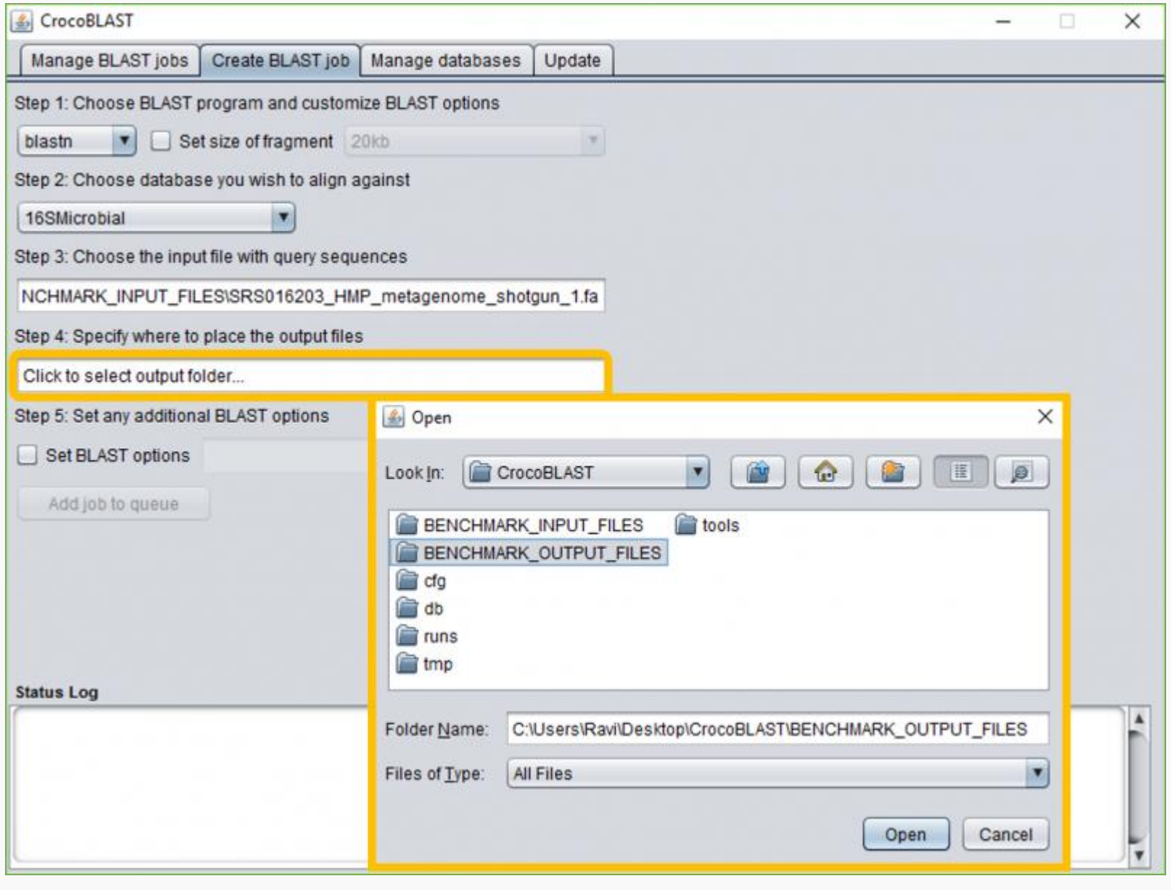
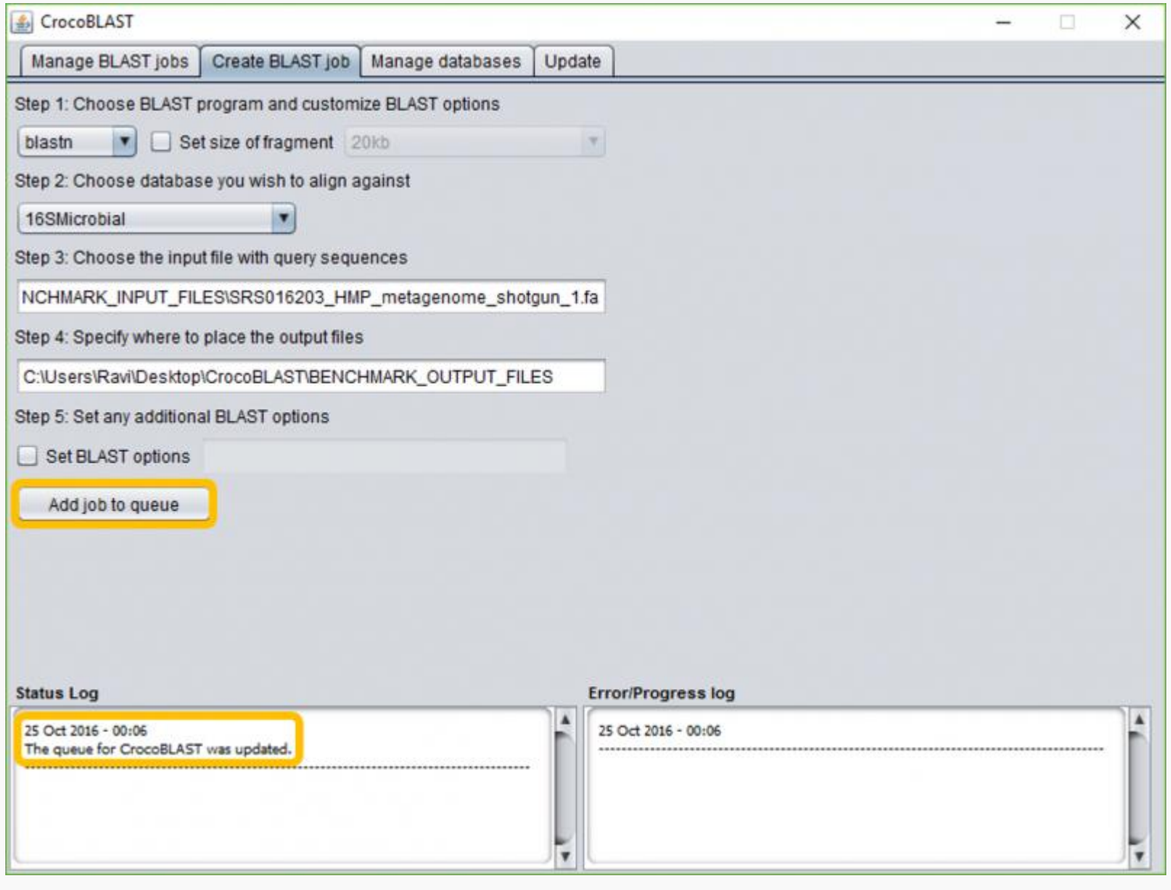
a) Select the "Create BLAST job" tab.



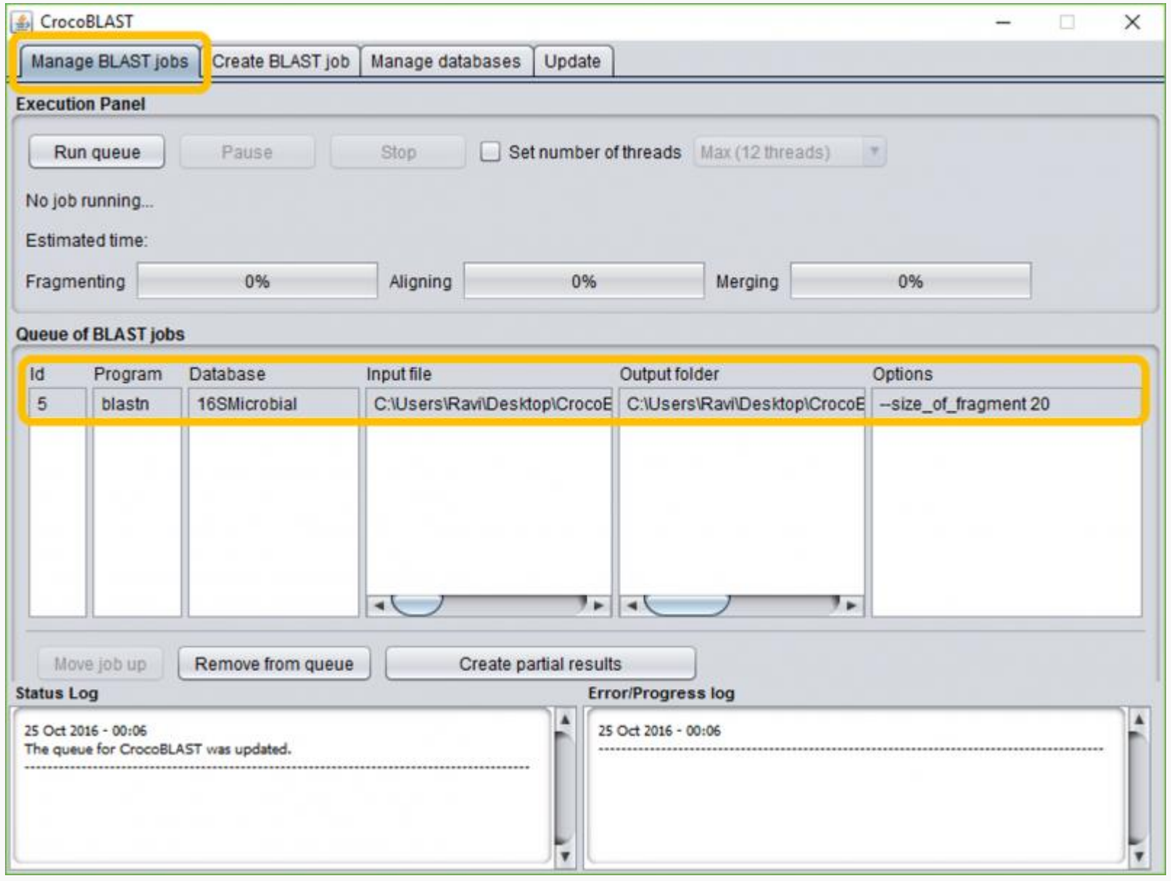
b) Choose the desired BLAST program.



c) Choose the desired database	**d)** Choose the input file that will be aligned against the database
	

e) Choose the output folder in which the final alignment result will be created	**f)** Click on the “Add job to queue” button (note the log message below)
	

g) That is all! Your job should now be visible on the “Manage BLAST jobs” tab

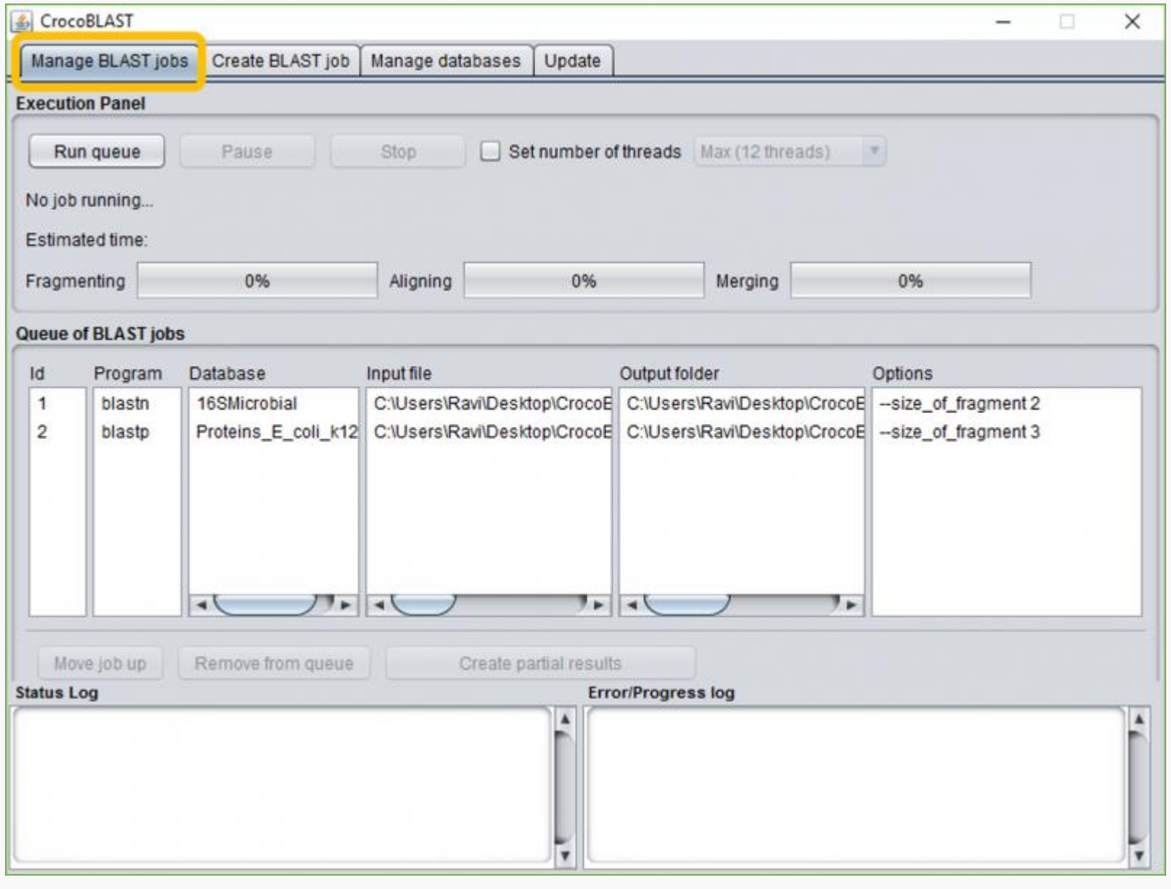


If no BLAST options are specified, CrocoBLAST will use all BLAST default values. If no size of fragment is specified, CrocoBLAST will take up to 5 seconds to auto-detect the ideal size for that job on your computer. When you create a BLAST job, CrocoBLAST automatically assigns each BLAST job a unique job ID, and updates the CrocoBLAST queue.

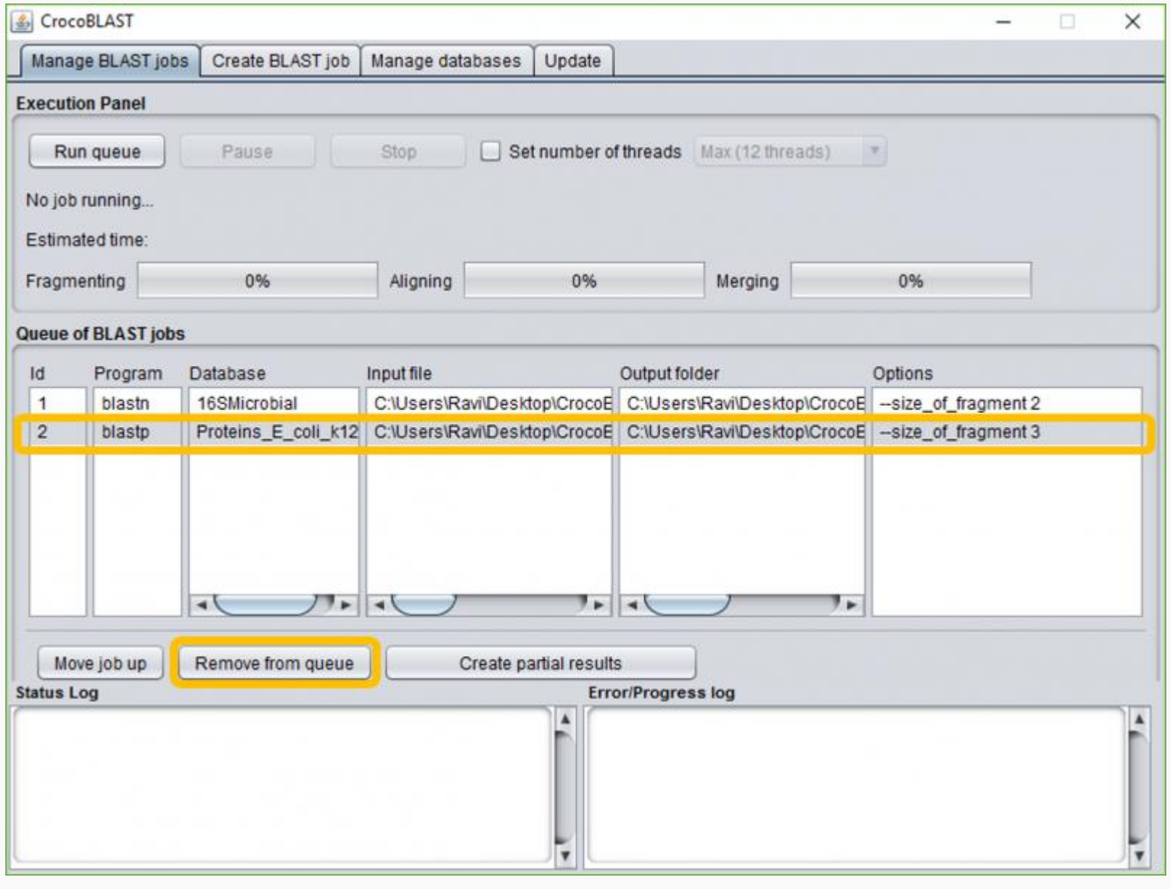
1.2 Removing BLAST jobs

In case you want to remove a BLAST job from the queue, you can follow the instructions below:

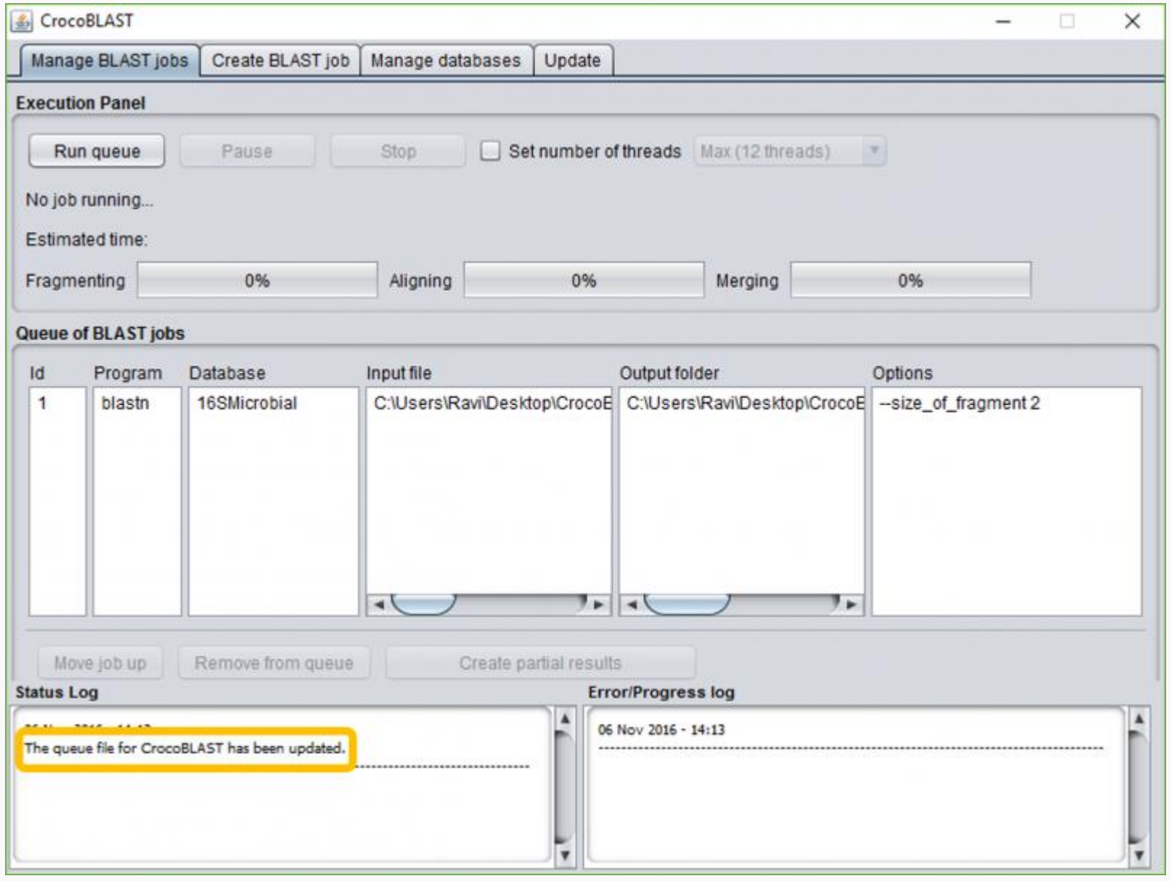
a) Select the “Manage BLAST jobs” tab



b) Select the Job you wish to remove and click the “Remove from queue” button.



c) That is it, your queue has been updated!

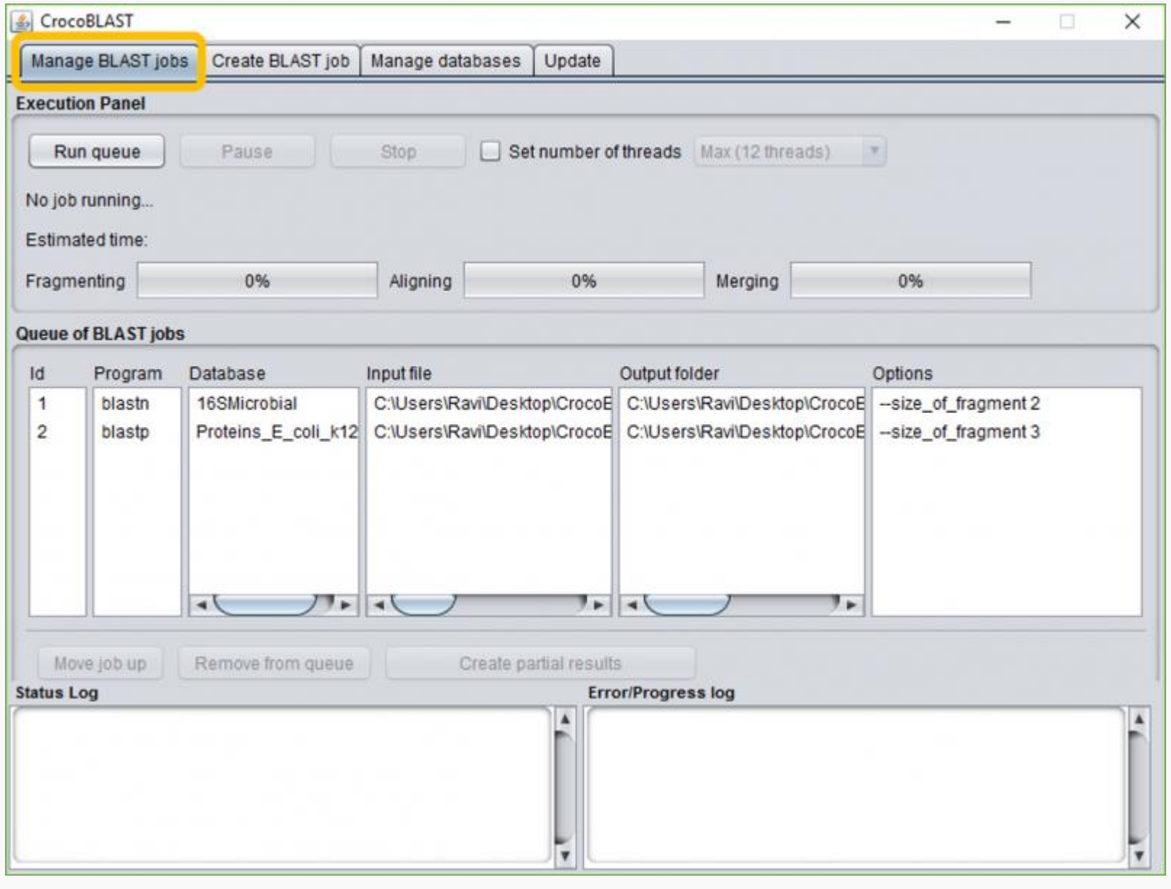


If you wish to remove several jobs at once, you might want to use the command line example "IV.1.2.2".

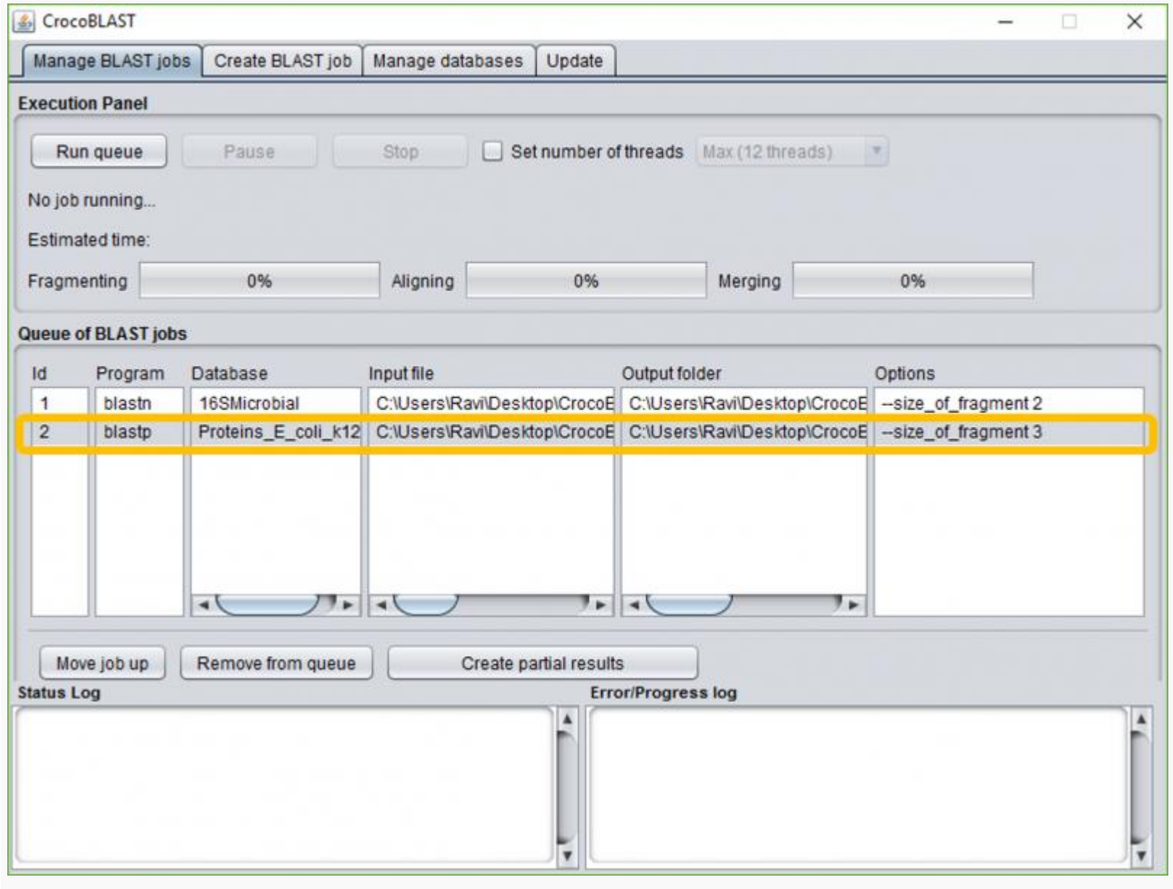
1.3 Moving a BLAST job to the top of the queue

As only one job (the first on queue) is active on CrocoBLAST at each time, moving another job to the top of the queue can be useful for changing the order in which the jobs will be executed. You can move a job to the top of the CrocoBLAST queue by following the steps:

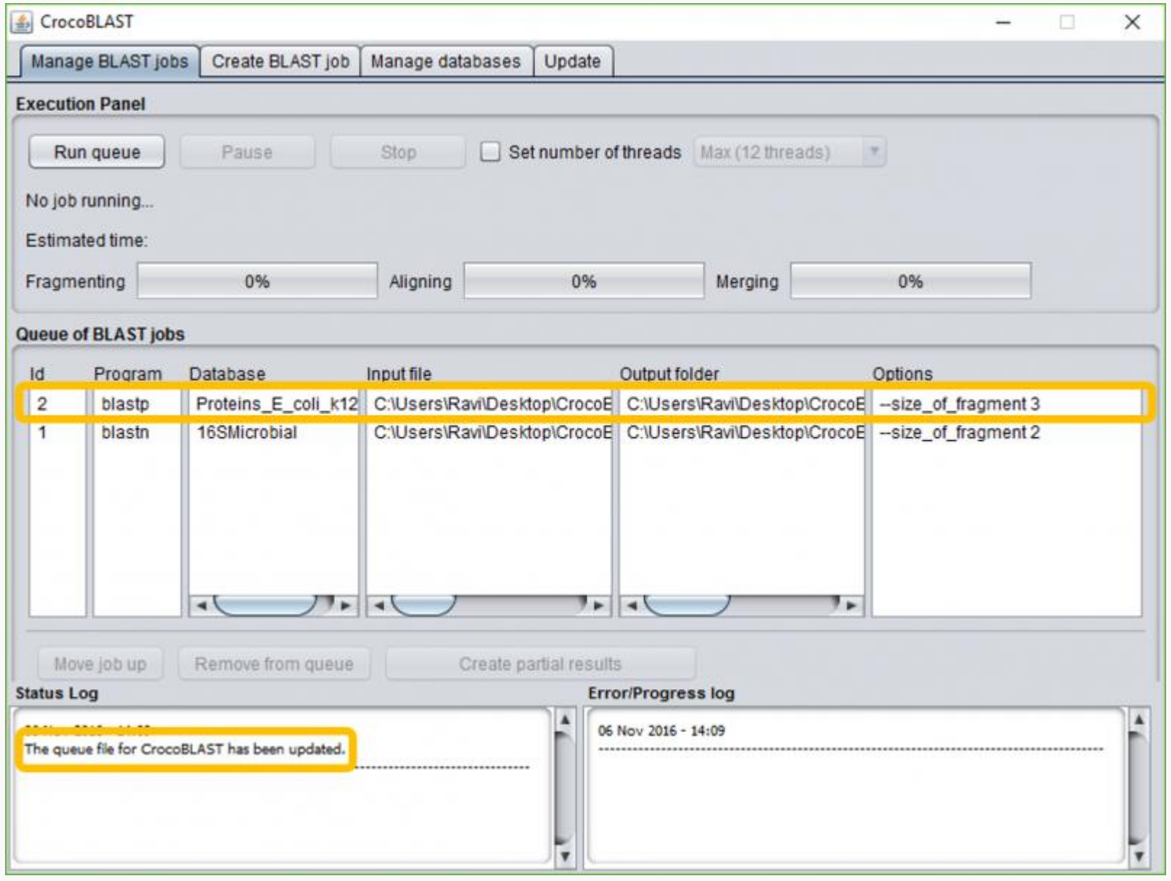
a) Select the "Manage BLAST jobs" tab.



b) Select the Job you wish to move the top of the queue.



c) That is it, your queue was updated!



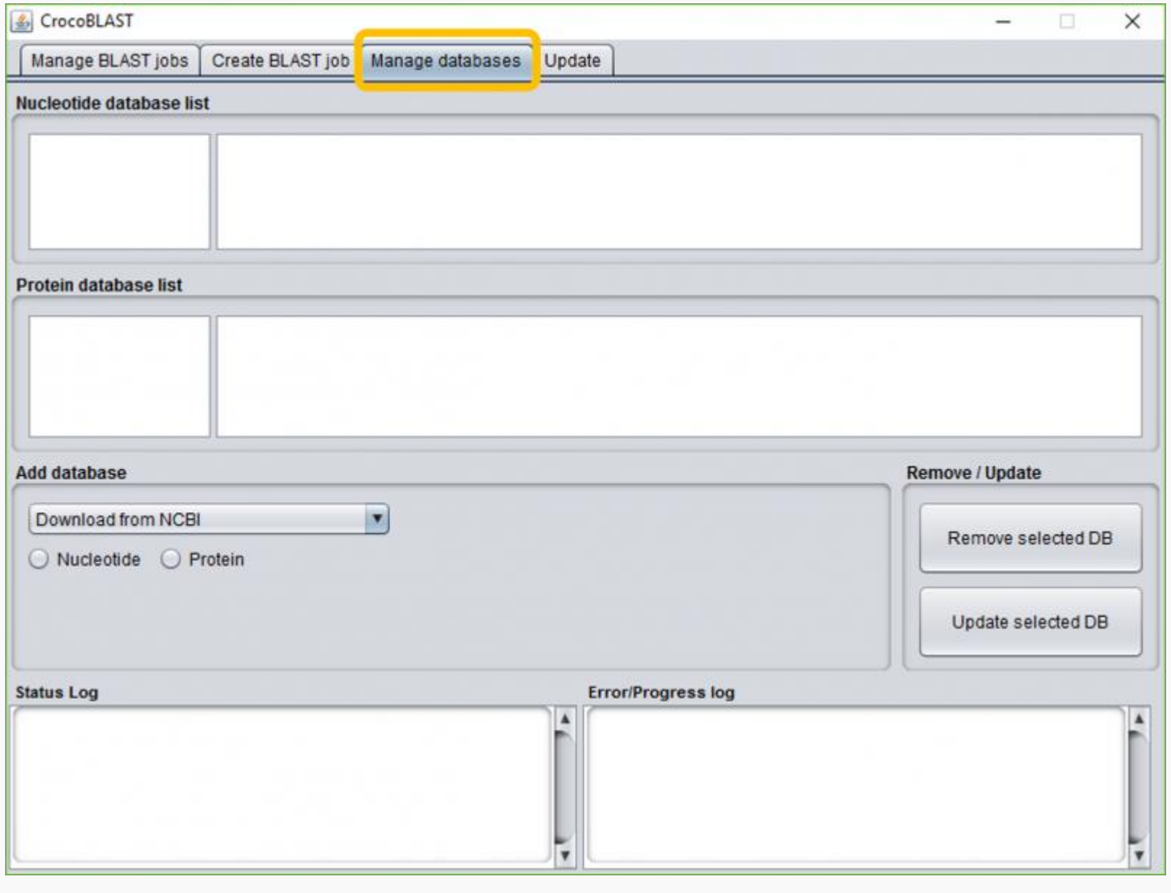
2. Manage databases

The first time you add a database into CrocoBLAST, it will be set as an internal entry with a given name, so that in the future it is easier for you to access this database. All databases entered in CrocoBLAST can be inspected in the “Manage databases” tab.

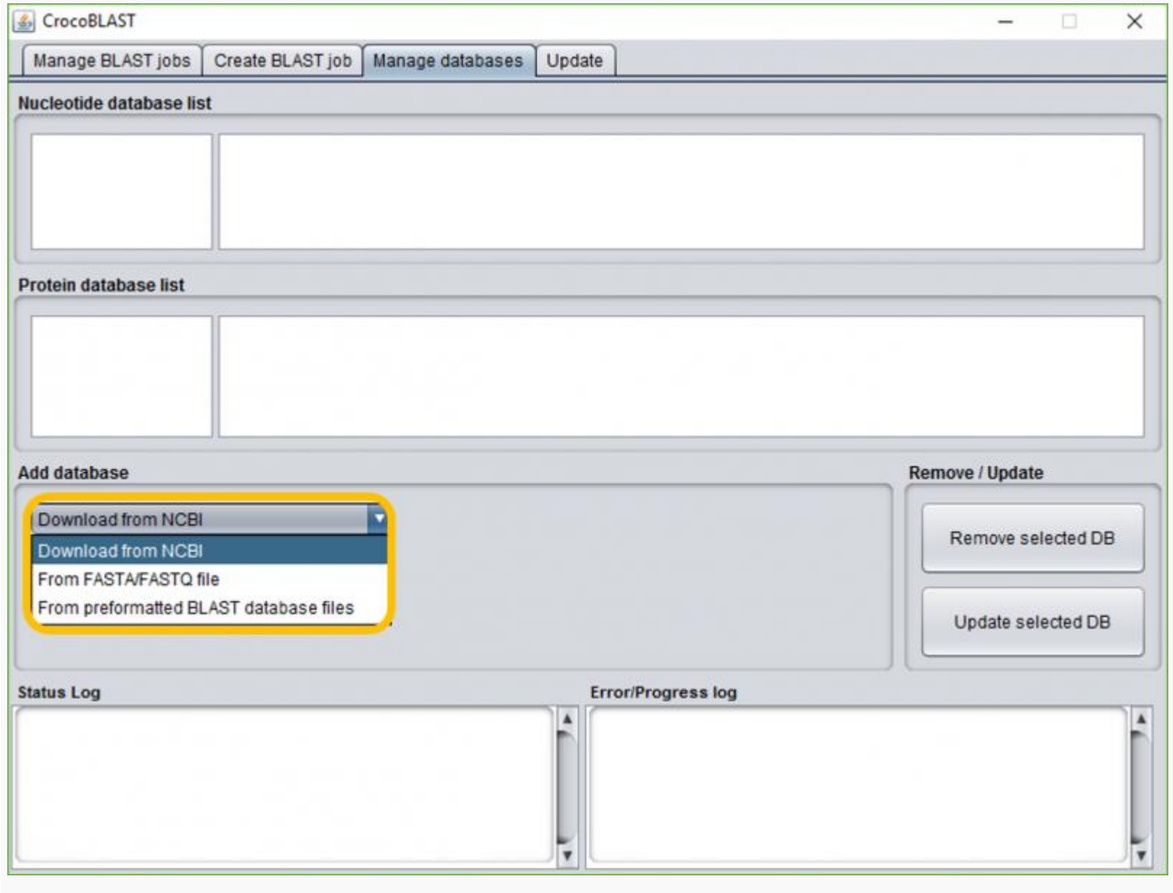
2.1. Retrieving and adding a database from the NCBI servers

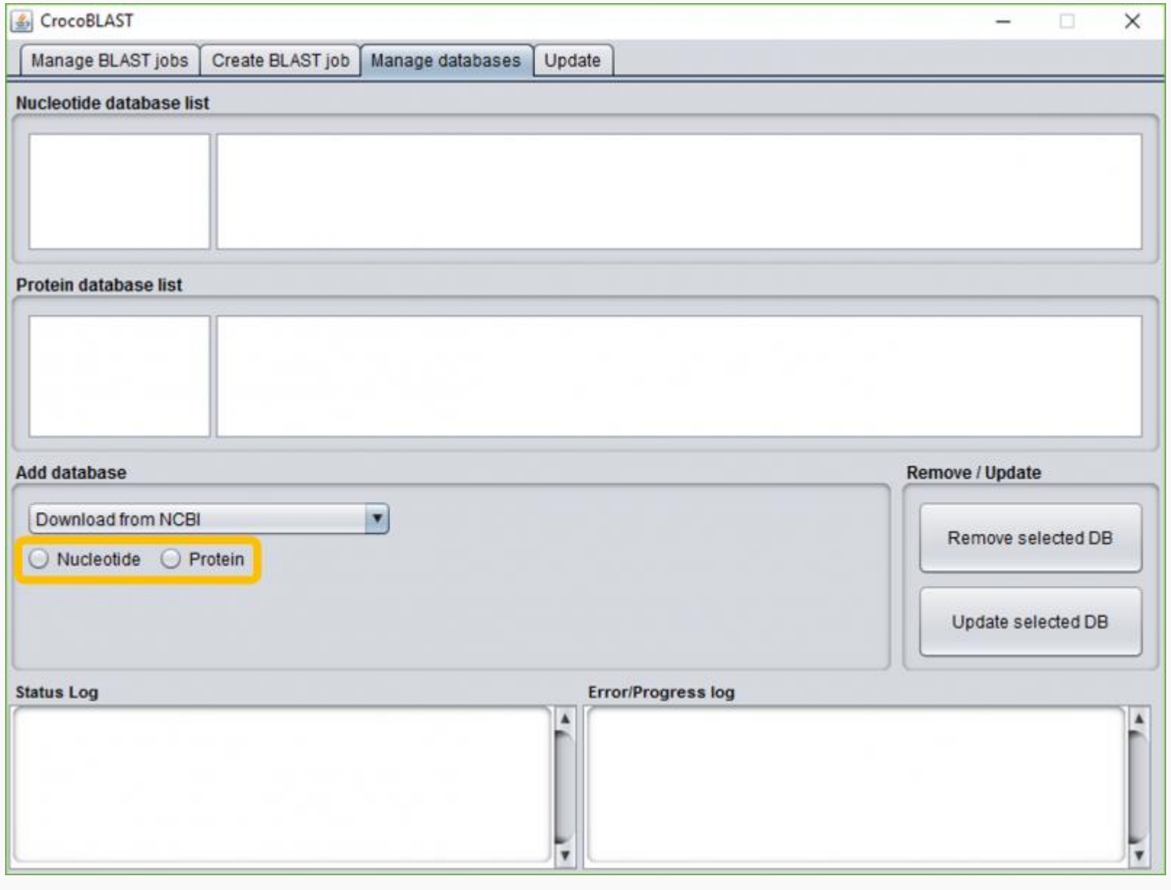
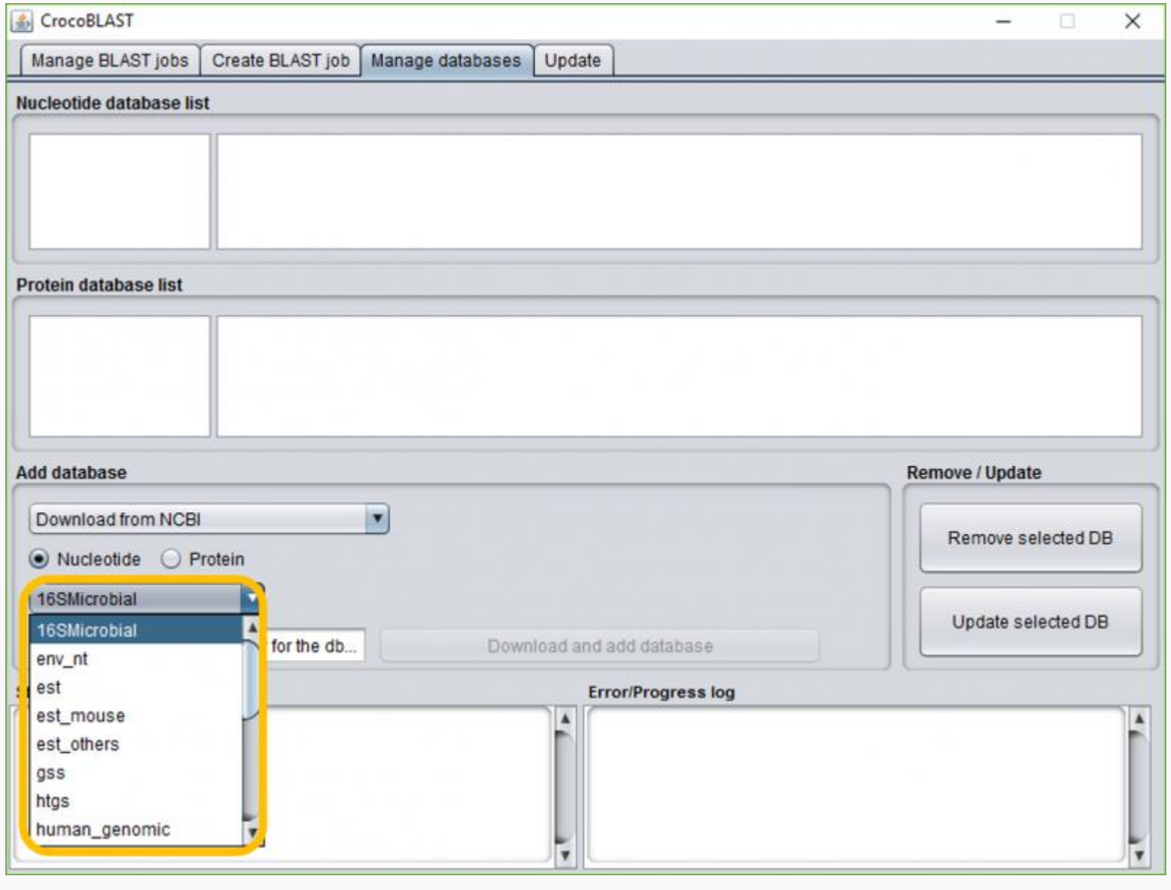
In the most typical scenario, you will use the established reference sequence databases maintained by NCBI. CrocoBLAST allows you to specify the name of such a database, and will download or update the database for you:

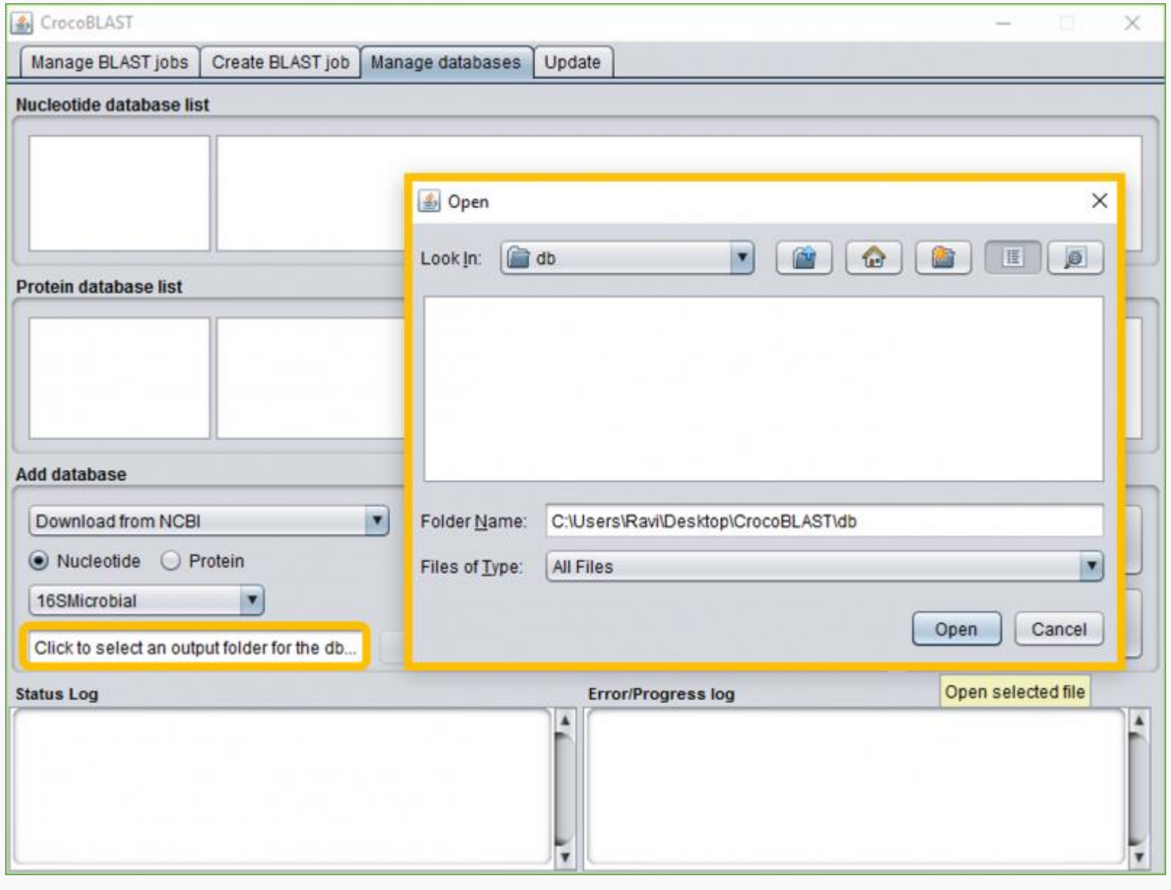
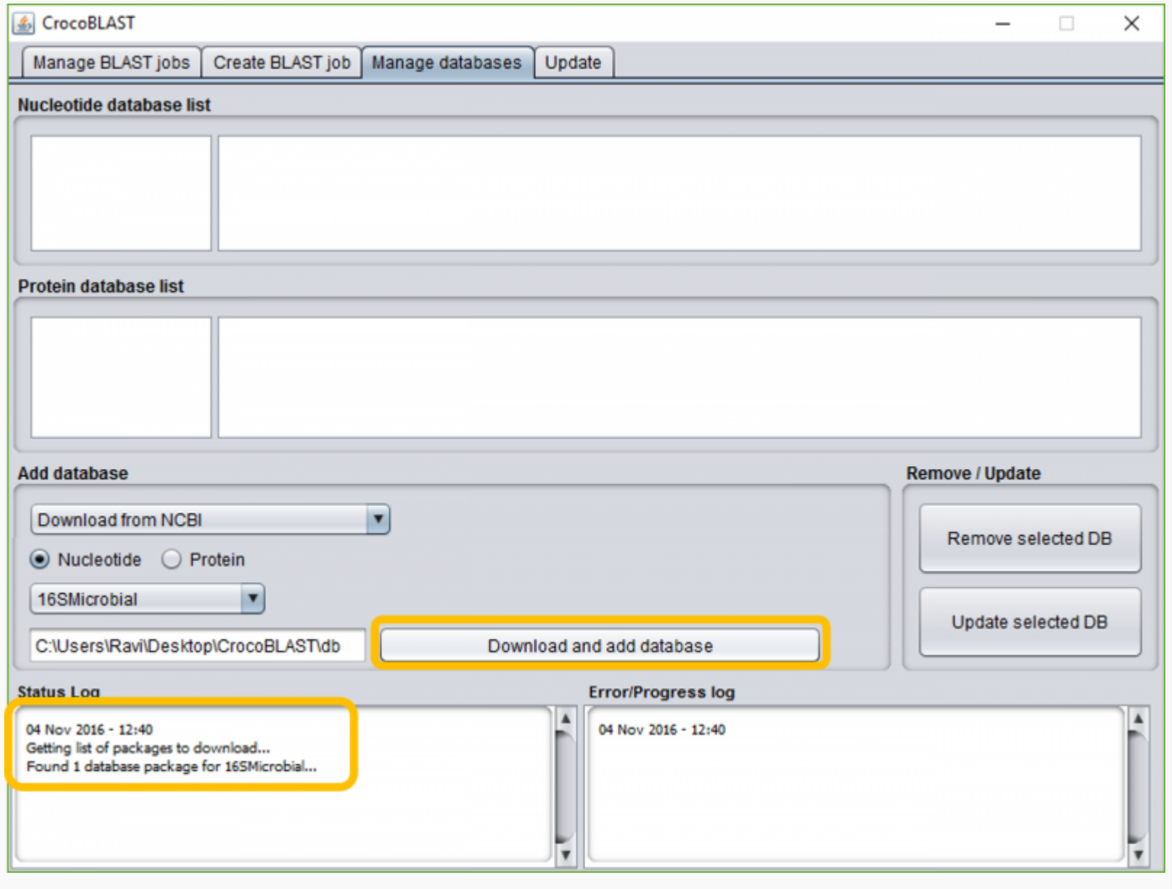
a) Select the “Manage databases” tab.



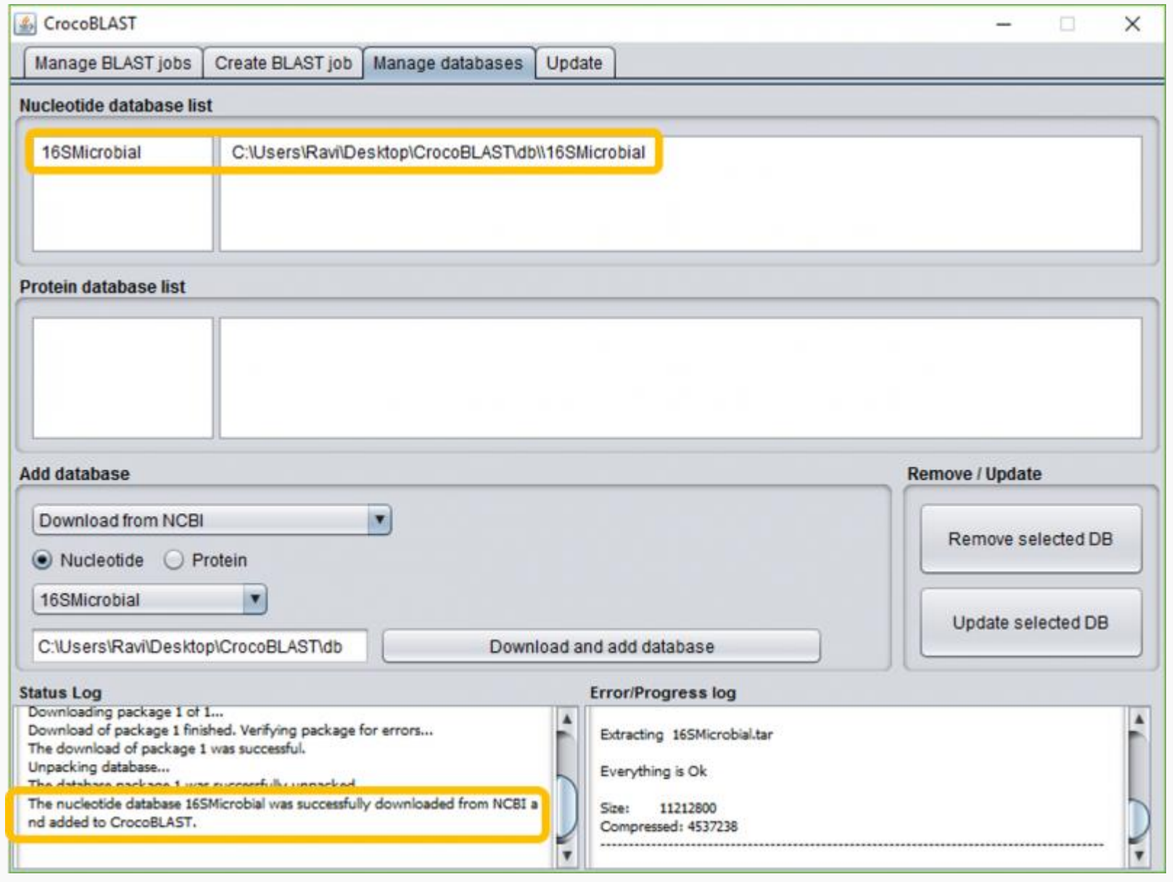
b) Select “Download from NCBI” as a source for the database



c) Select the type of database that you want to add	**d)** Select the name of database that you want to add
	

e) Choose the output folder in which the database will be set up	**f)** Click the “Download and add database” button (note the log message below)
	

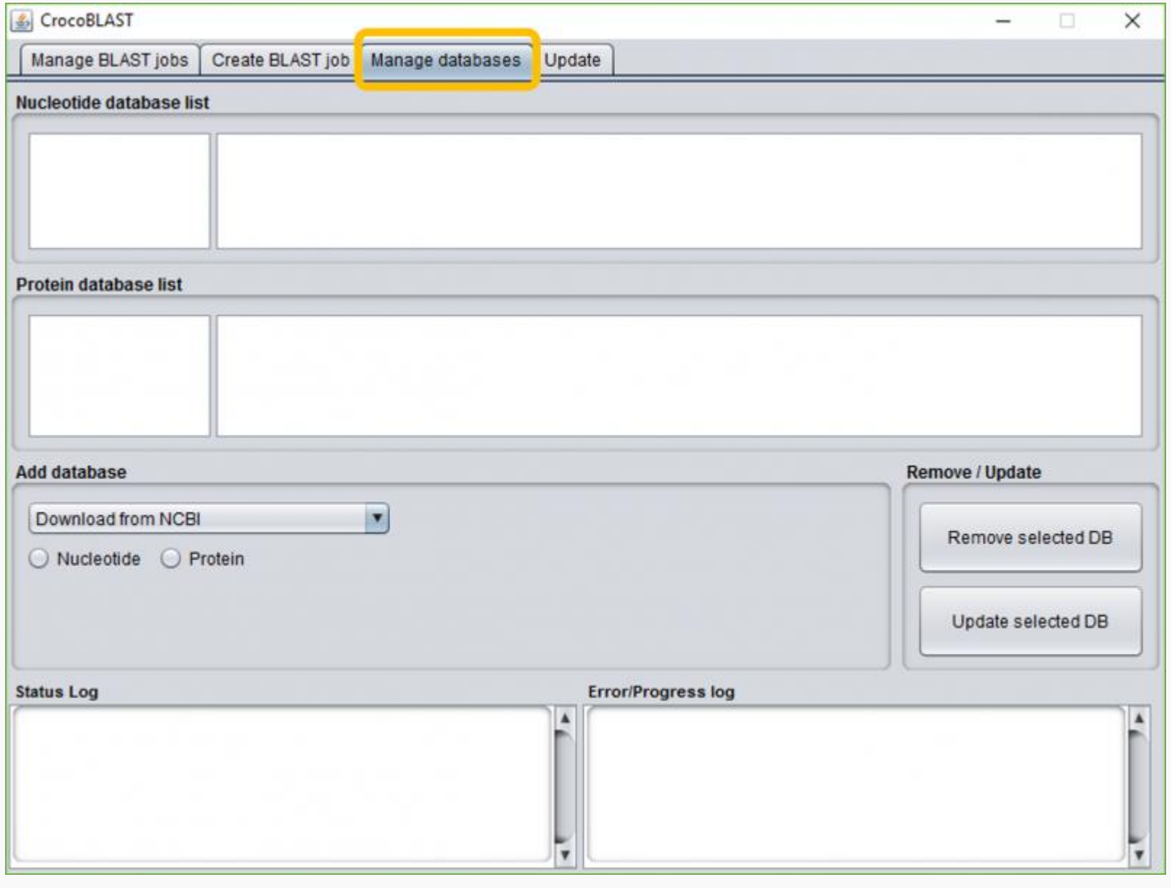
g) The log may get messy but do not worry. All you need is the success message!



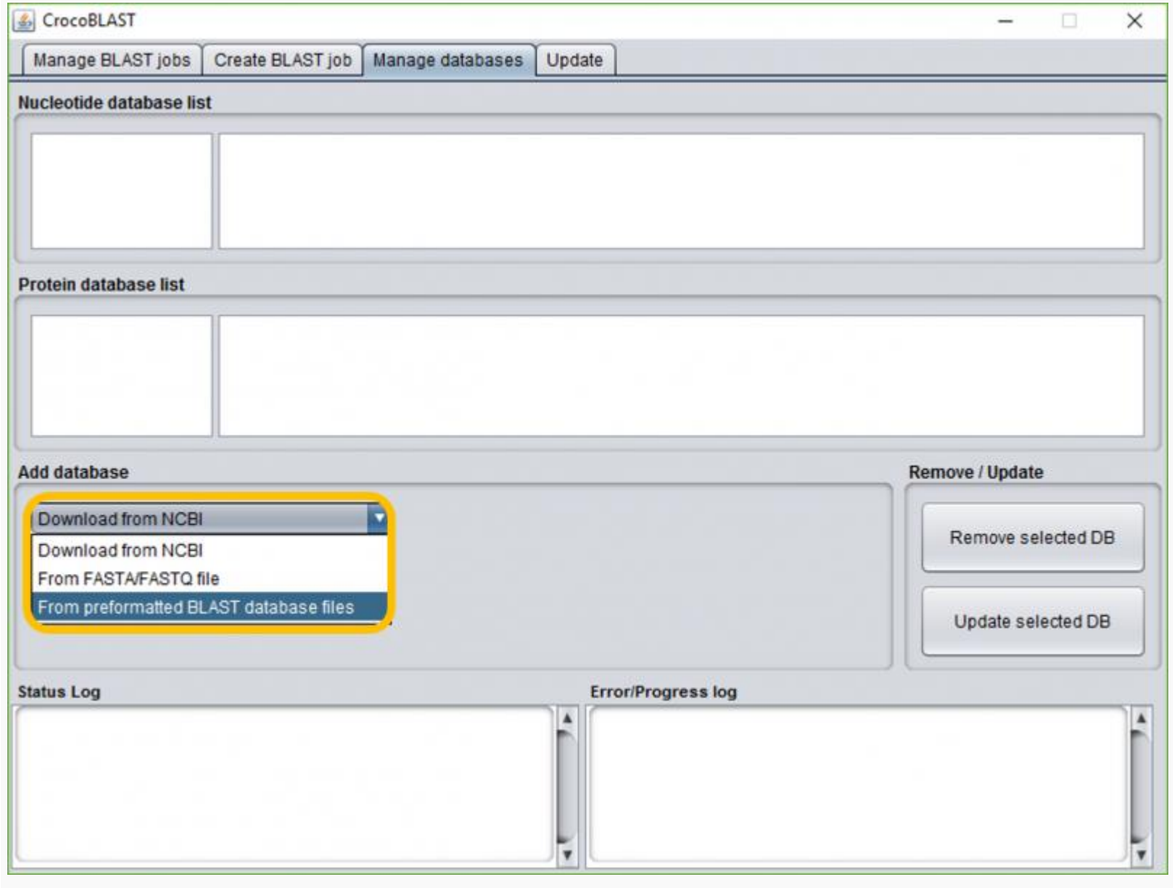
2.2. Adding a pre-formatted database from your computer

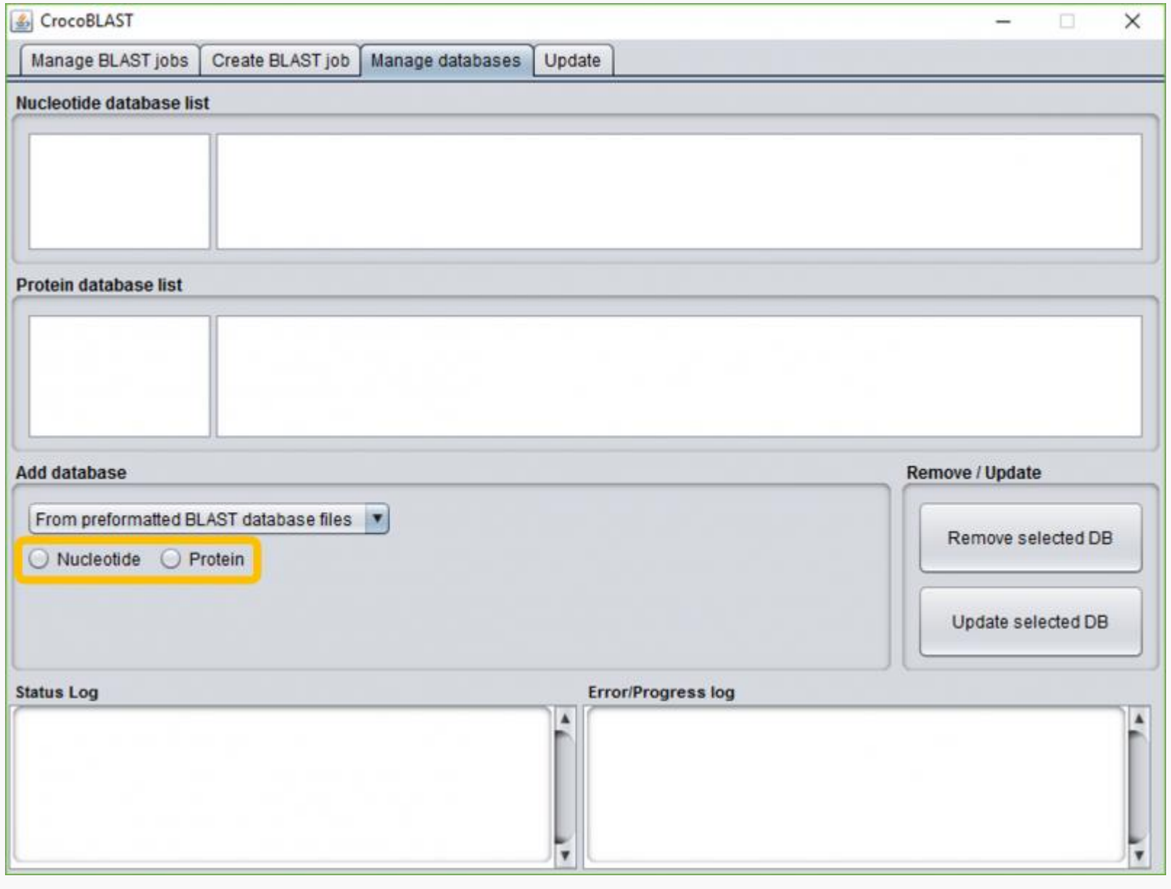
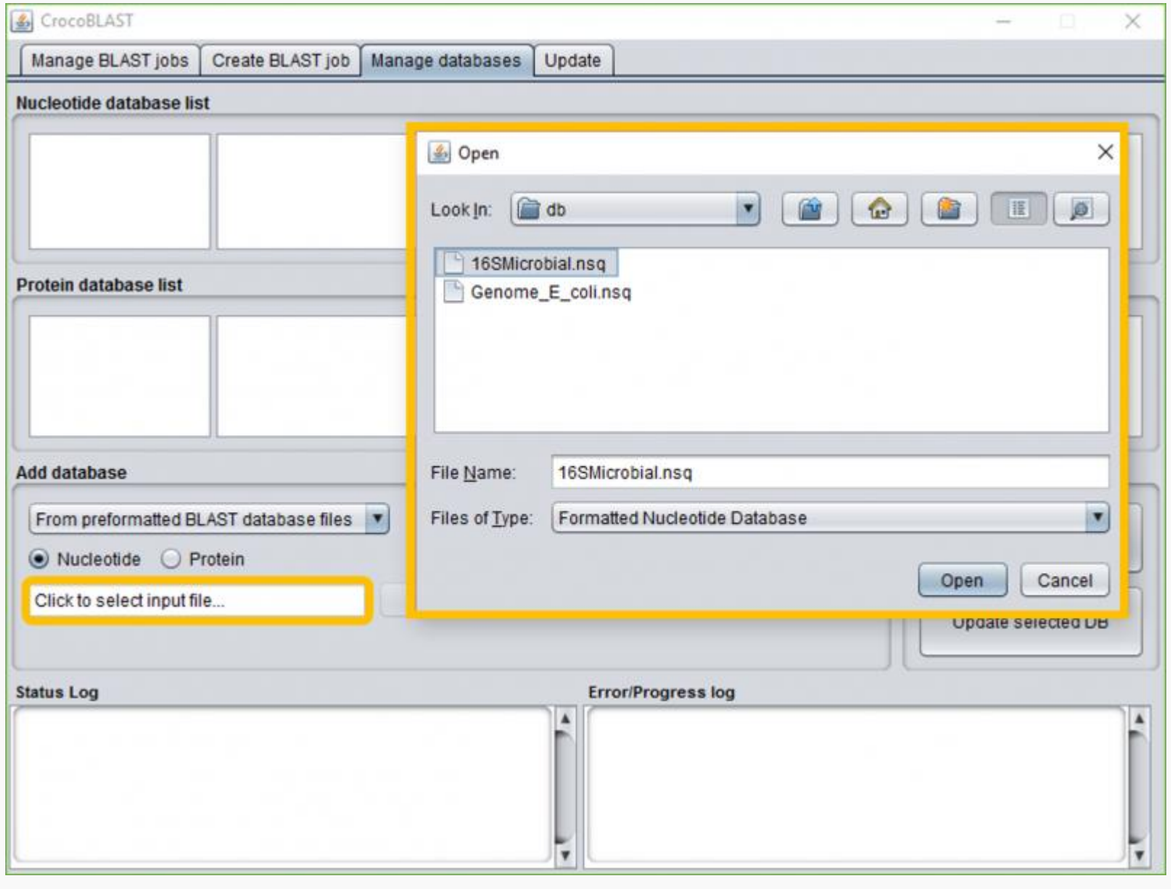
If you have already downloaded BLAST databases, or if you do not have internet connection, you may add to the CrocoBLAST index database files stored on your computer. The following command can be used when you have pre-formatted database files (e.g., psq or nsq):

a) Select the “Manage databases” tab.

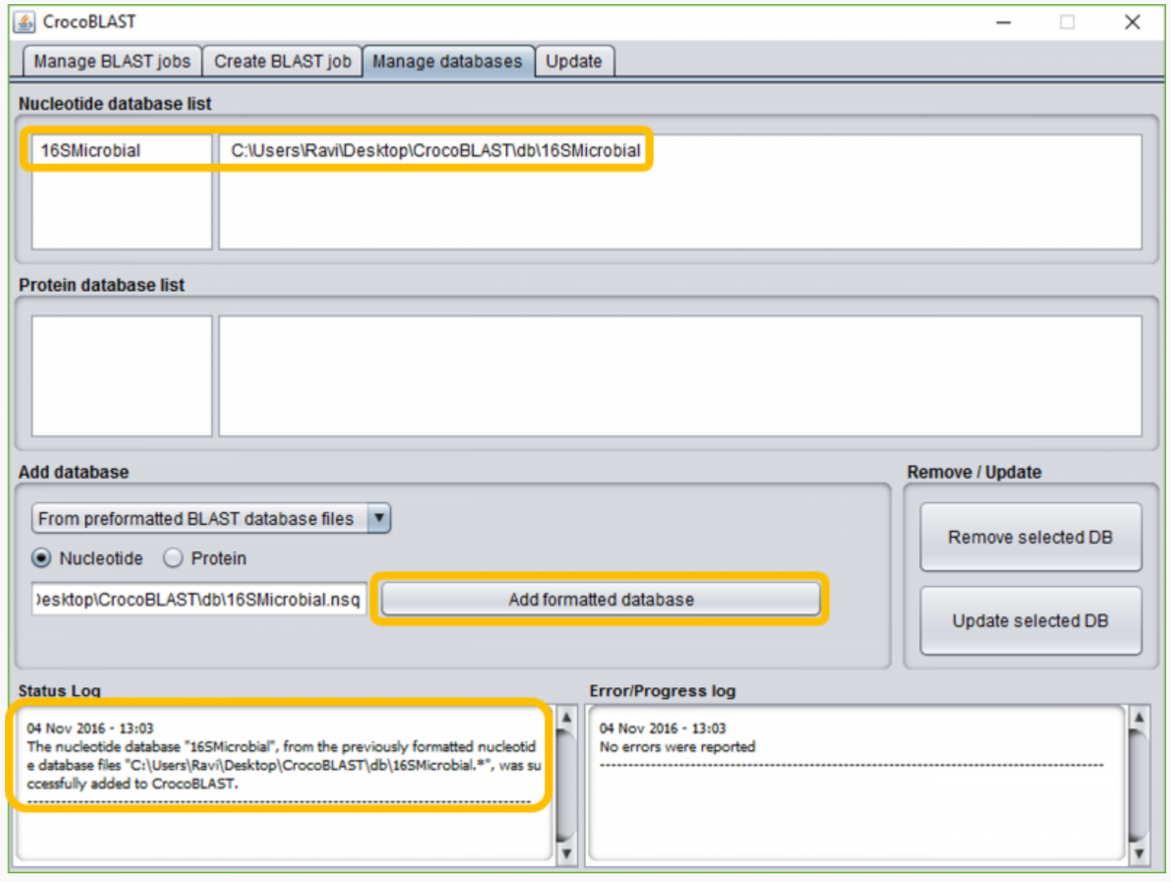


b) Select “From formatted BLAST database files” as a source for the database



c) Select the type of database that you want to add	**d)** Choose the formatted database file containing the desired database
	

e) Click the “Add formatted database” button (note the log message below). That is all!

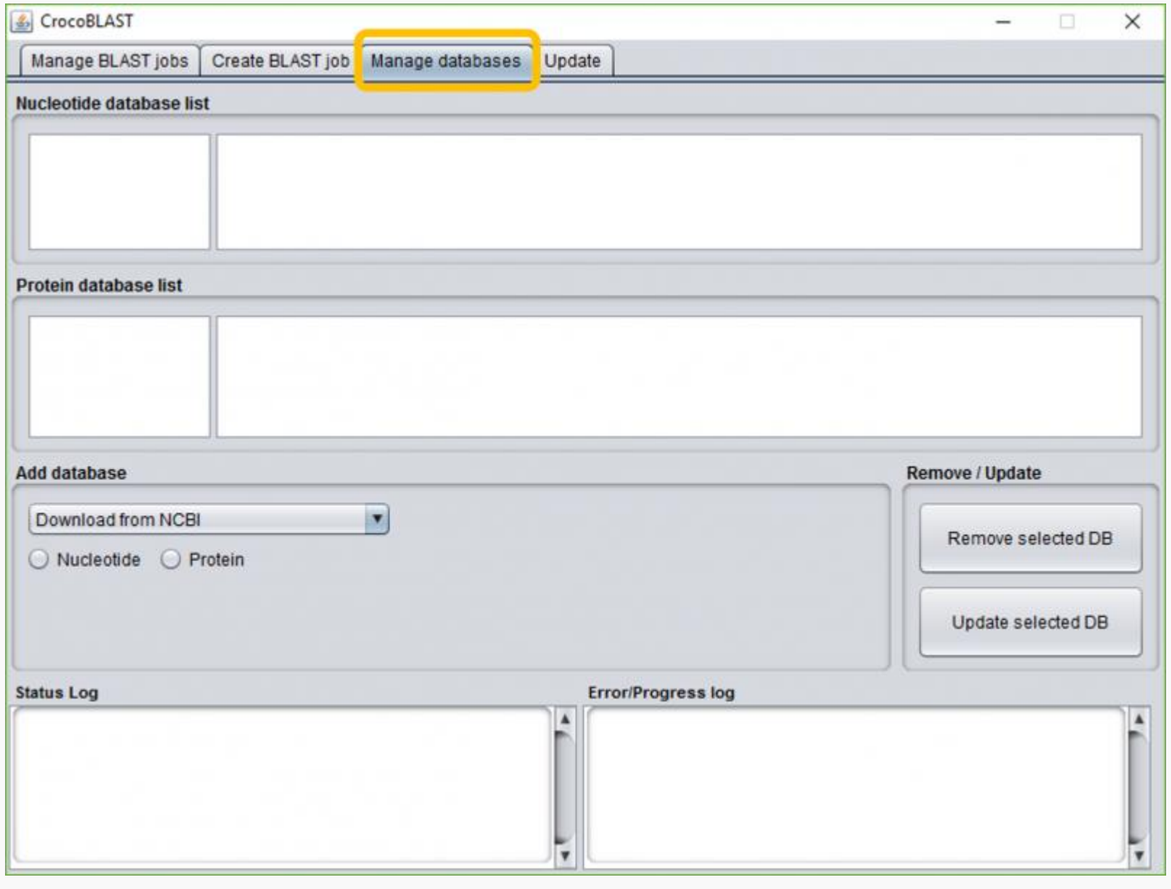


Note that CrocoBLAST will assume that all database files related to the given ".nsq" file are contained in the same folder (which is most likely the case).

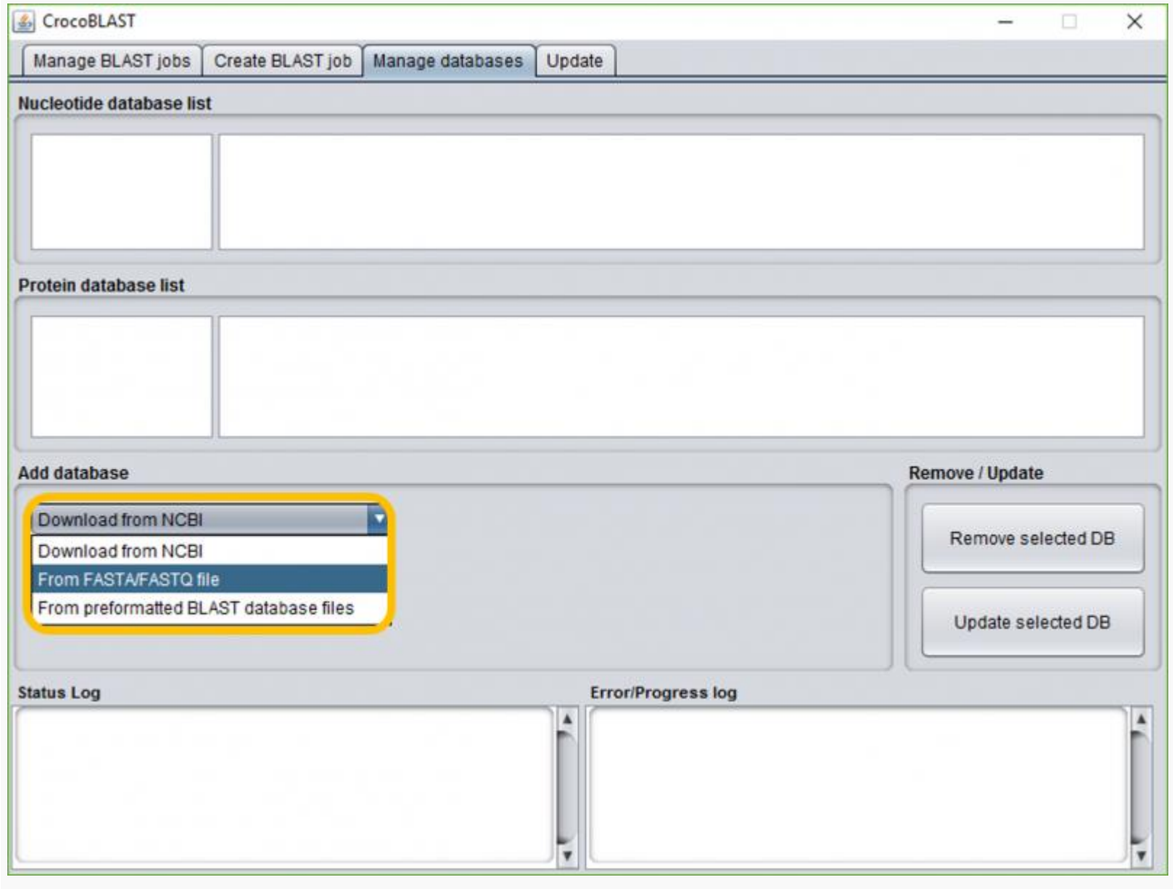
2.3. Adding a database from a FASTA/FASTQ sequence file in your computer

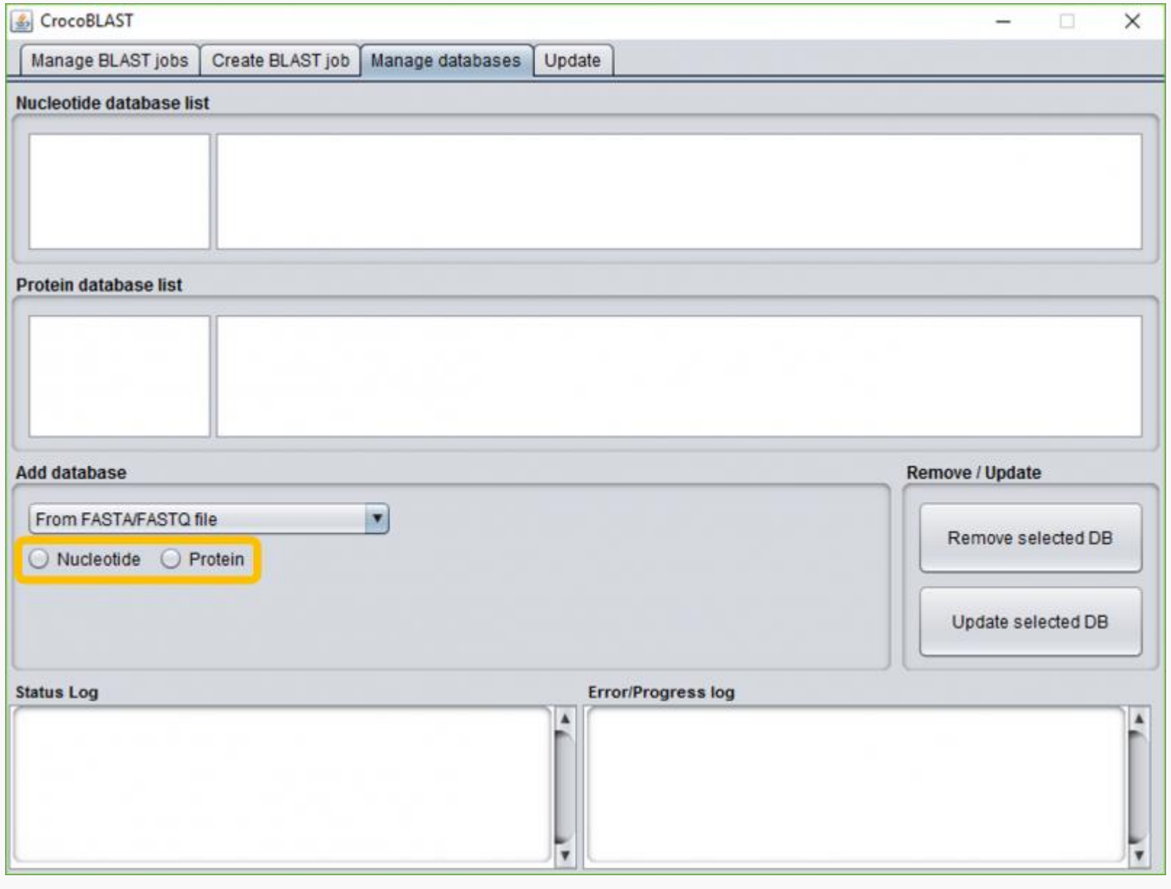
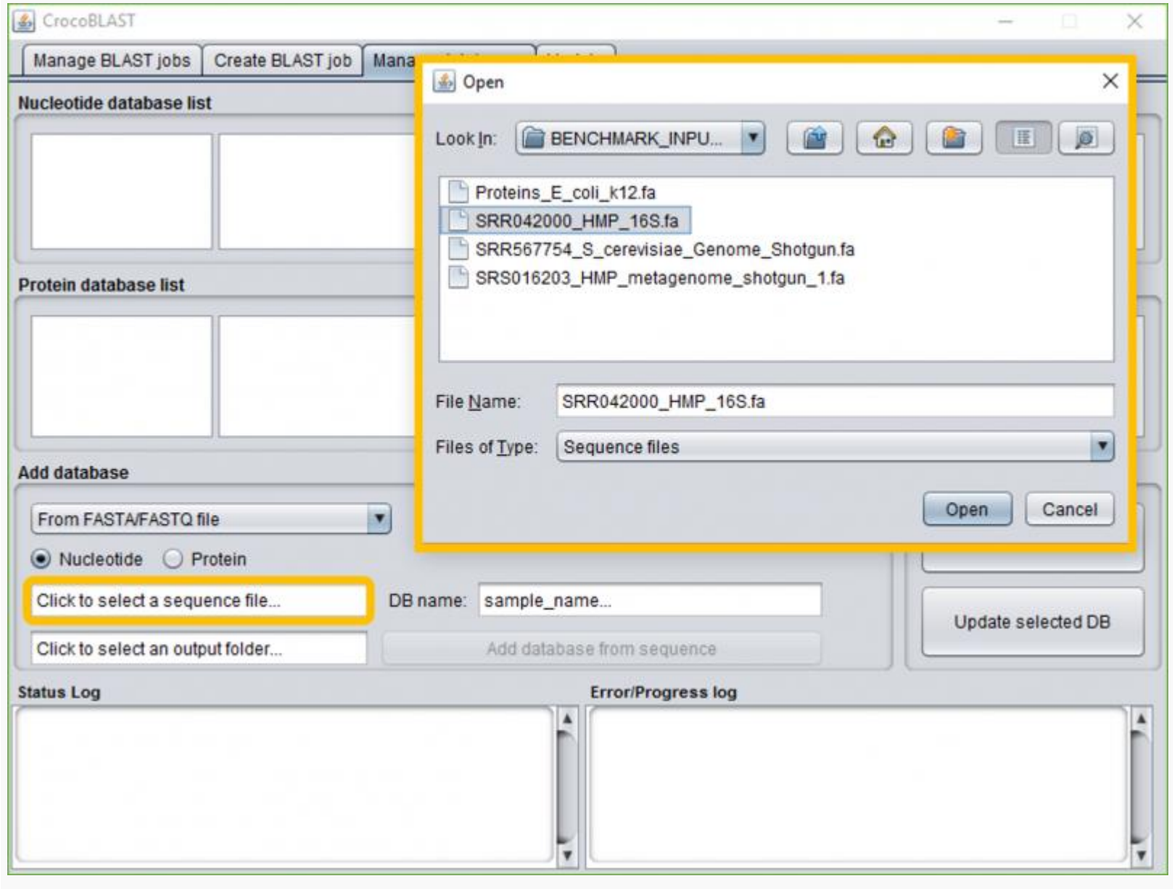
Remember to provide a unique and representative name for each database you add, so that it is easy to refer to the databases later. When your database is in FASTA or FASTQ format, you will need to tell CrocoBLAST the **type of sequence** it will find in the file:

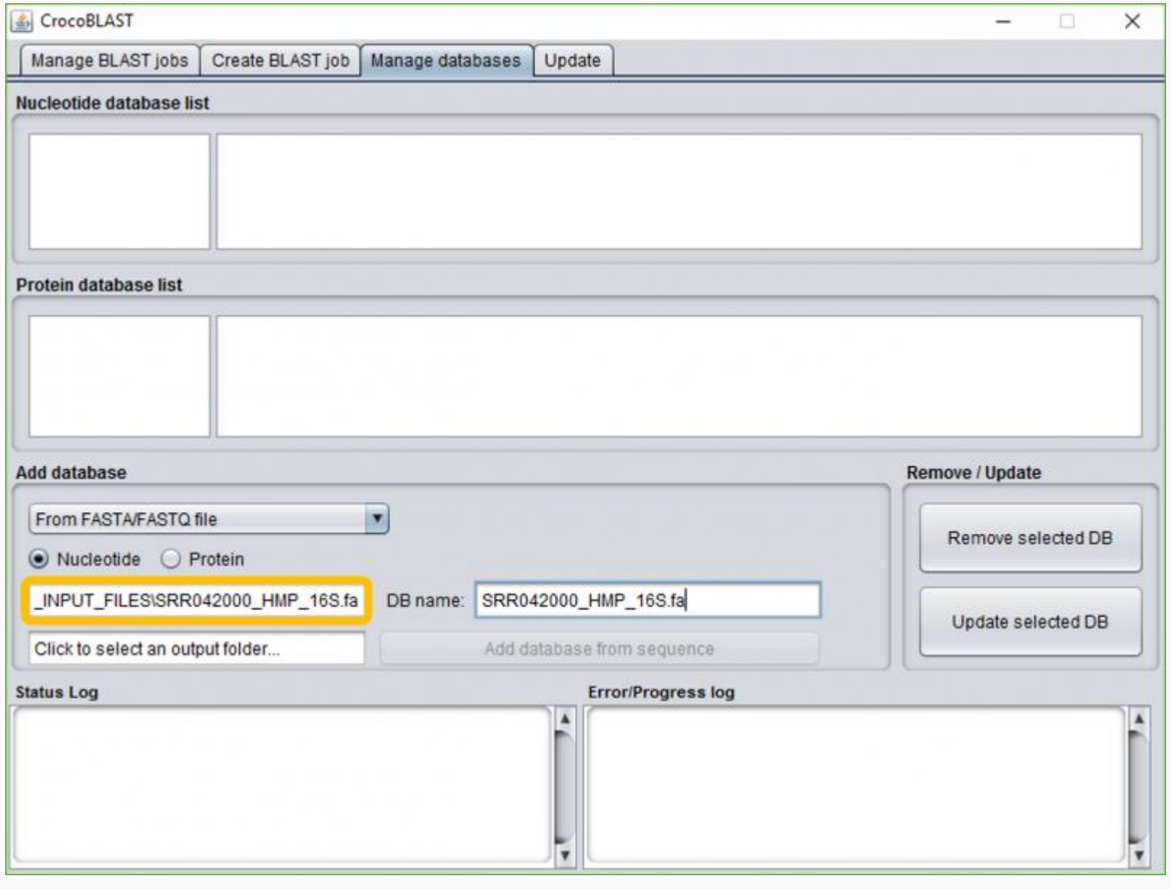
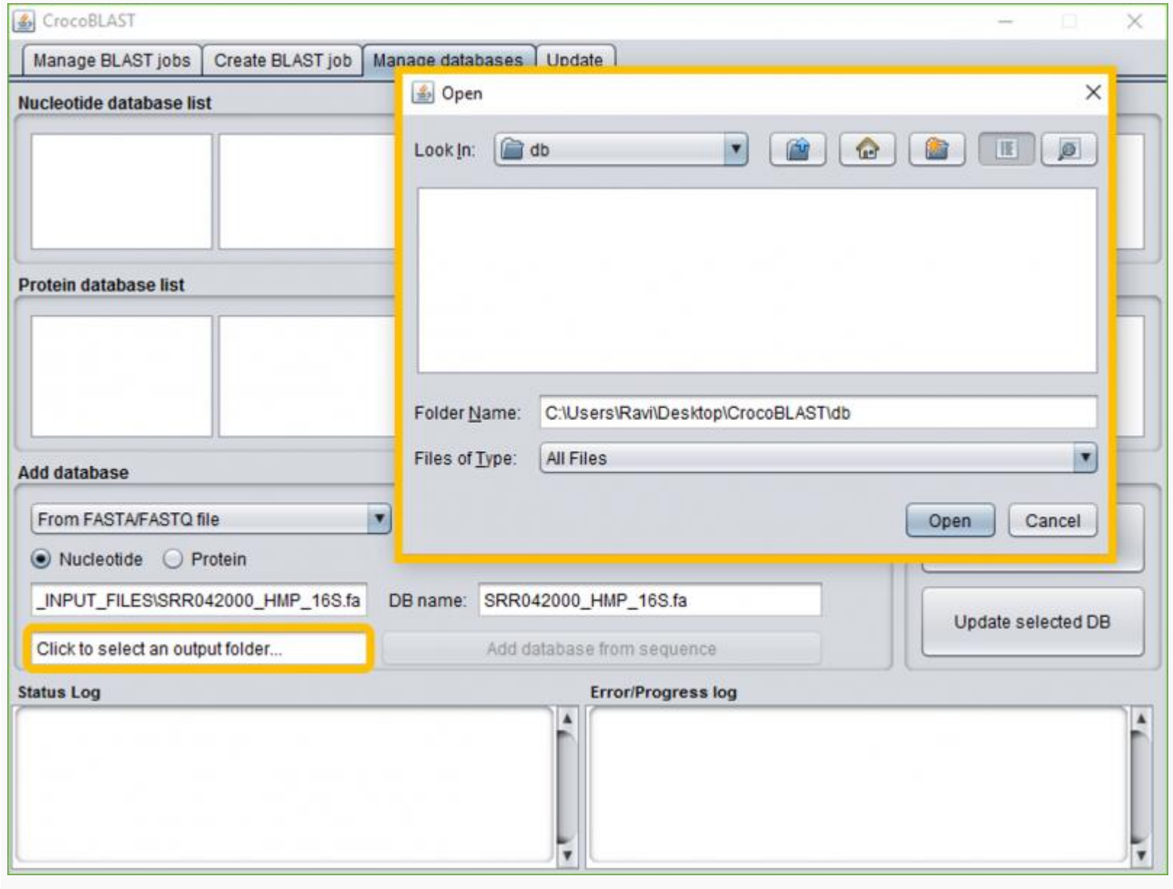
a) Select the "Manage databases" tab.



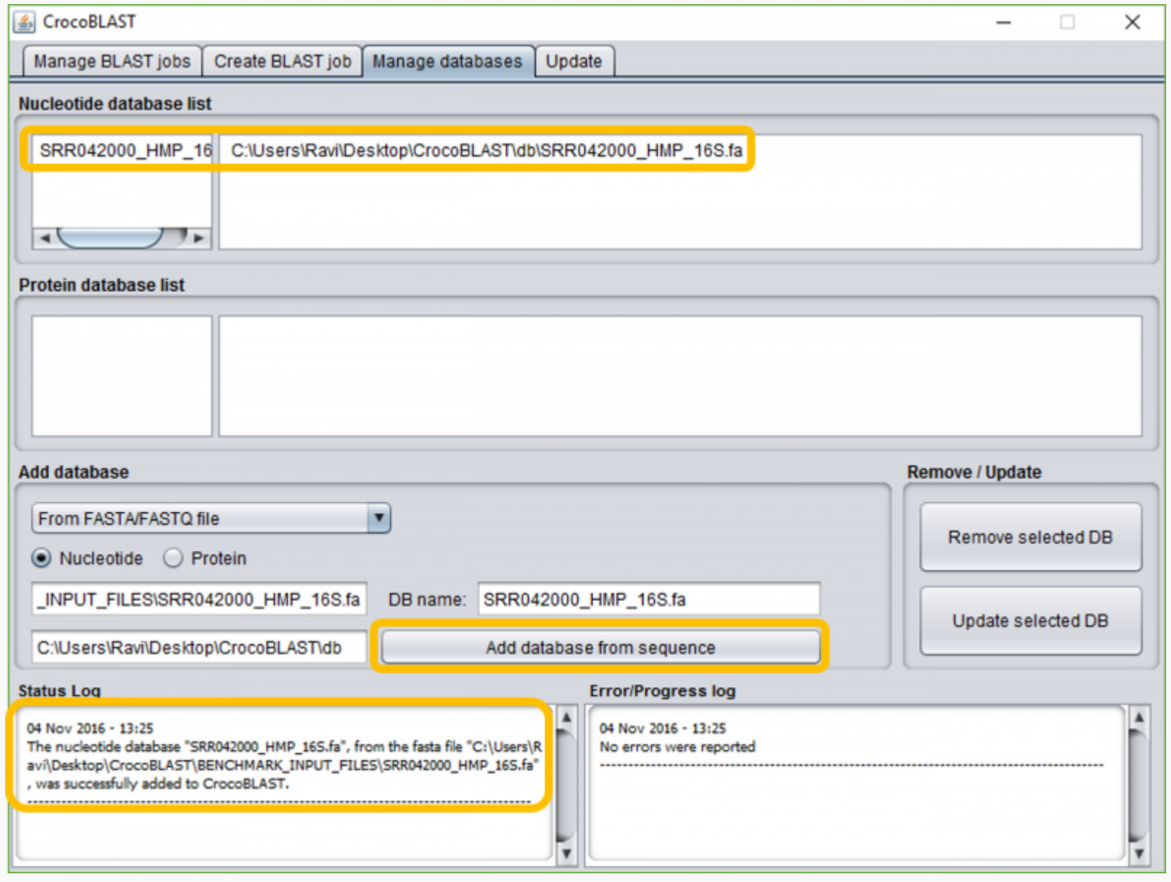
b) Select "From FASTA/FASTQ file" as a source for the database



c) Select the type of database that you want to add	**d)** Choose the sequence file to be converted into a BLAST database
	

e) Change the database name if you want (it uses the file name as default)	**f)** Choose the output folder in which the database will be set up
	

g) Click the “Add database from sequence” button (note the log message below). That is all!

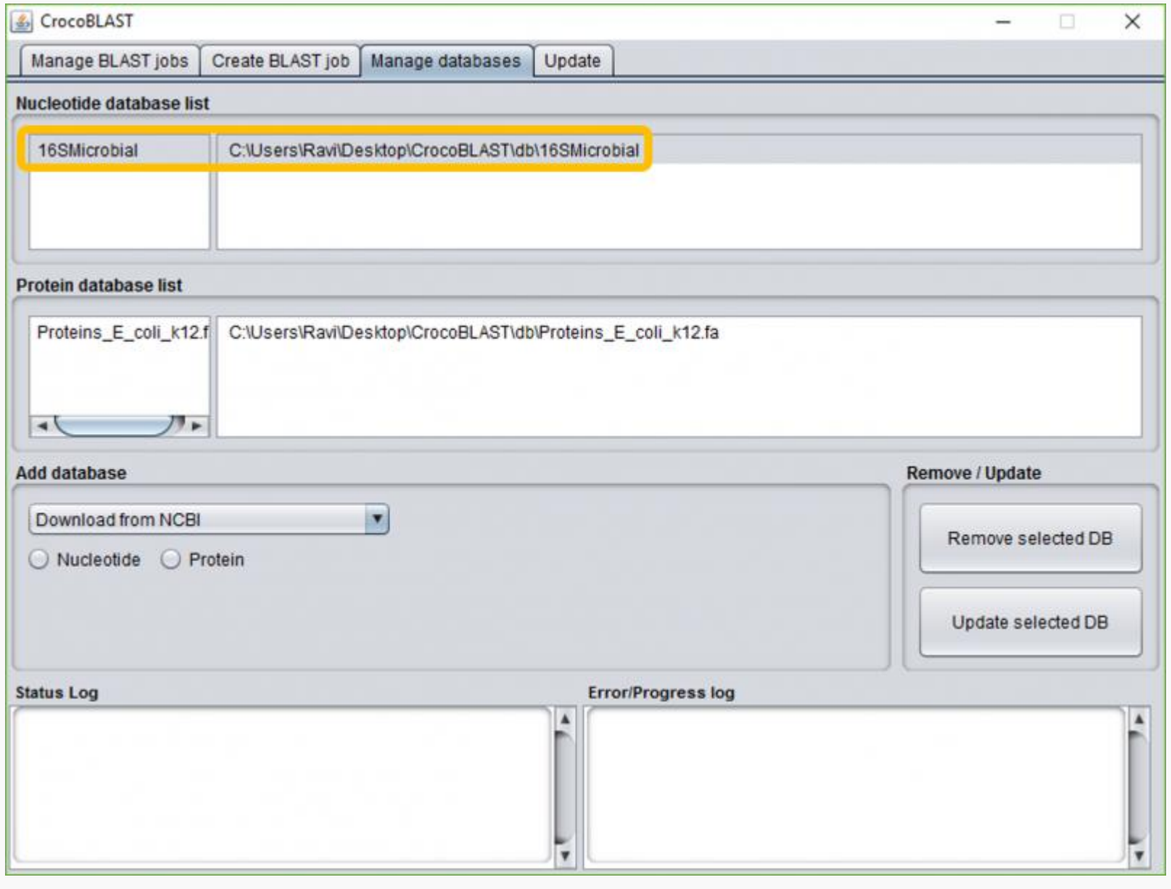


Formatted BLAST database files will be generated in the specified output folder.

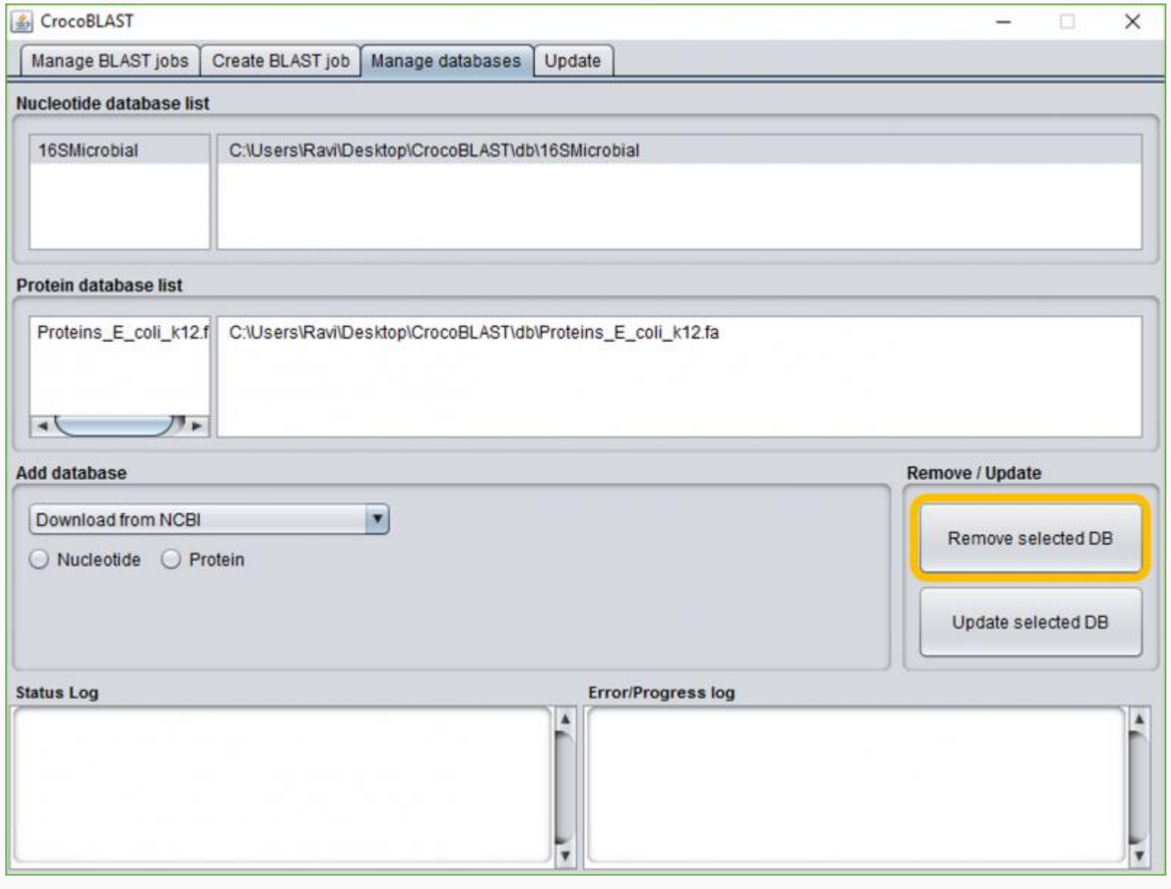
2.4 Removing a database

In case there is a missing entry in a database set for CrocoBLAST, or you have a new version of some database which name is already in use, it might be useful to remove a database from CrocoBLAST. This action does not delete the actual database files.

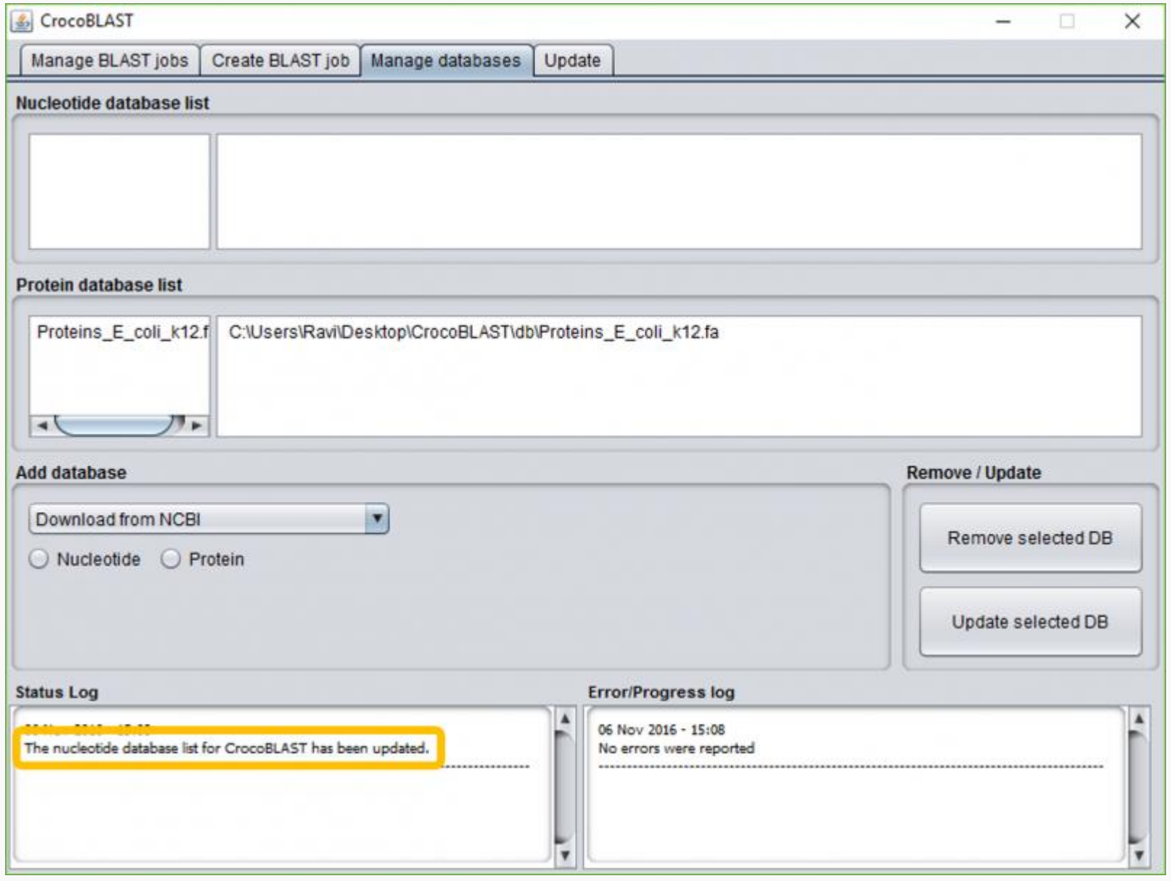
a) Select the database you wish to remove.



b) Click the “Remove selected DB” button.



c) Verify that the database list has been updated. That is all!



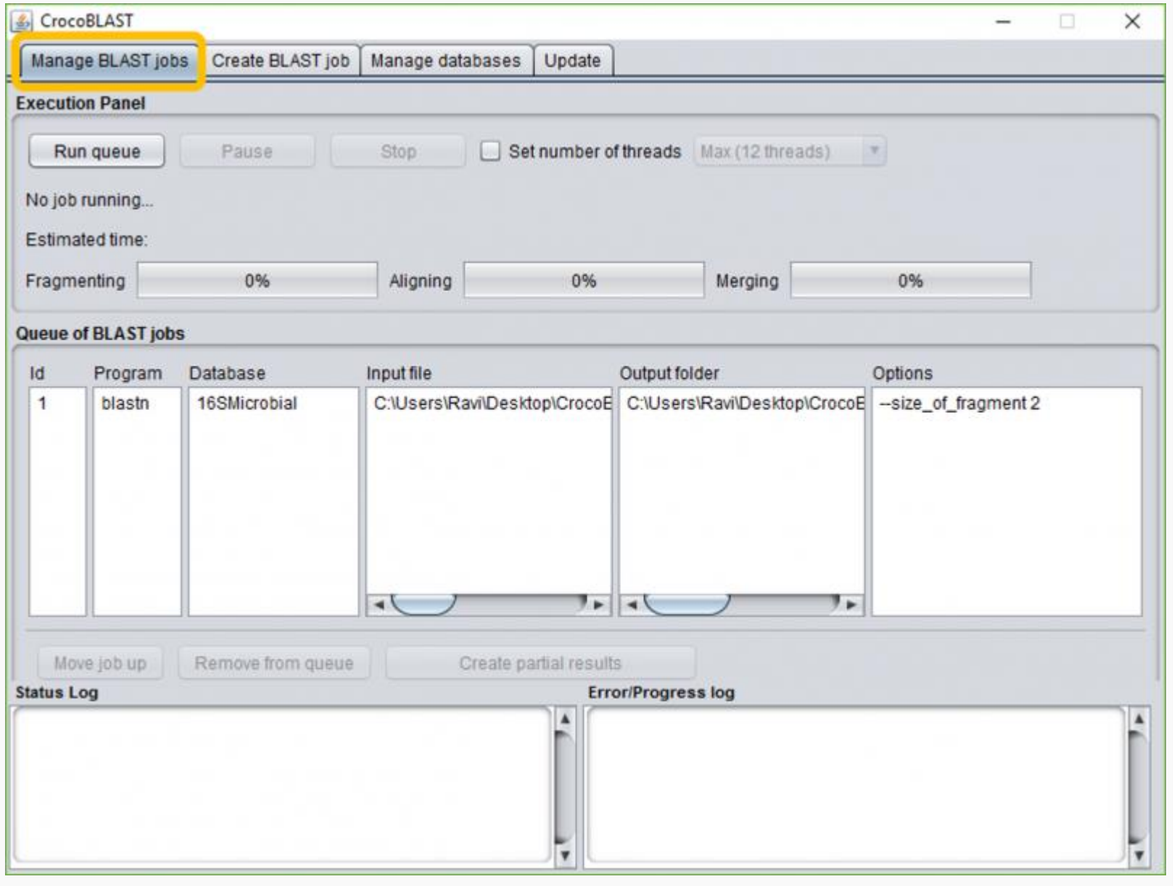
In case you accidentally removed a database by mistake, do not worry: as CrocoBLAST does not automatically delete the database files, you can re-add your database to CrocoBLAST with the commands in example 2.2.

3. Managing CrocoBLAST execution

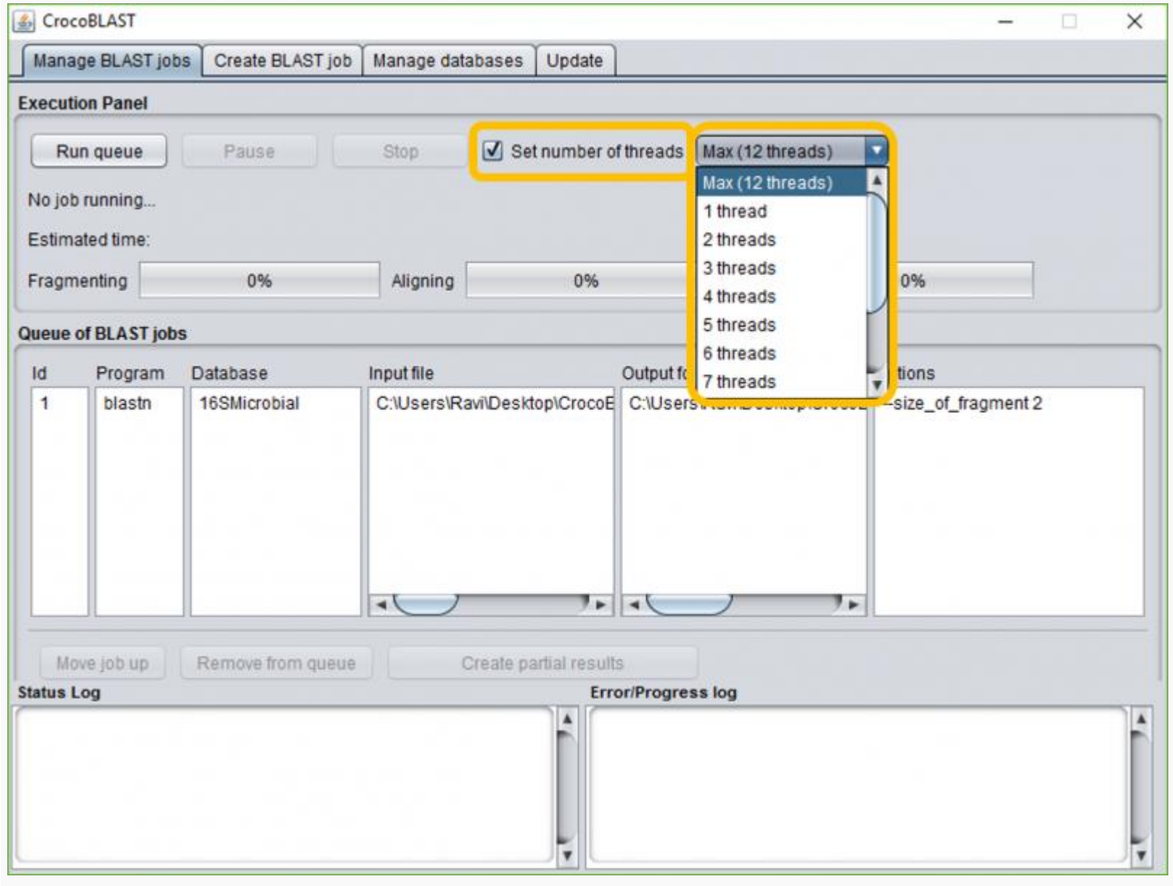
3.1. Running

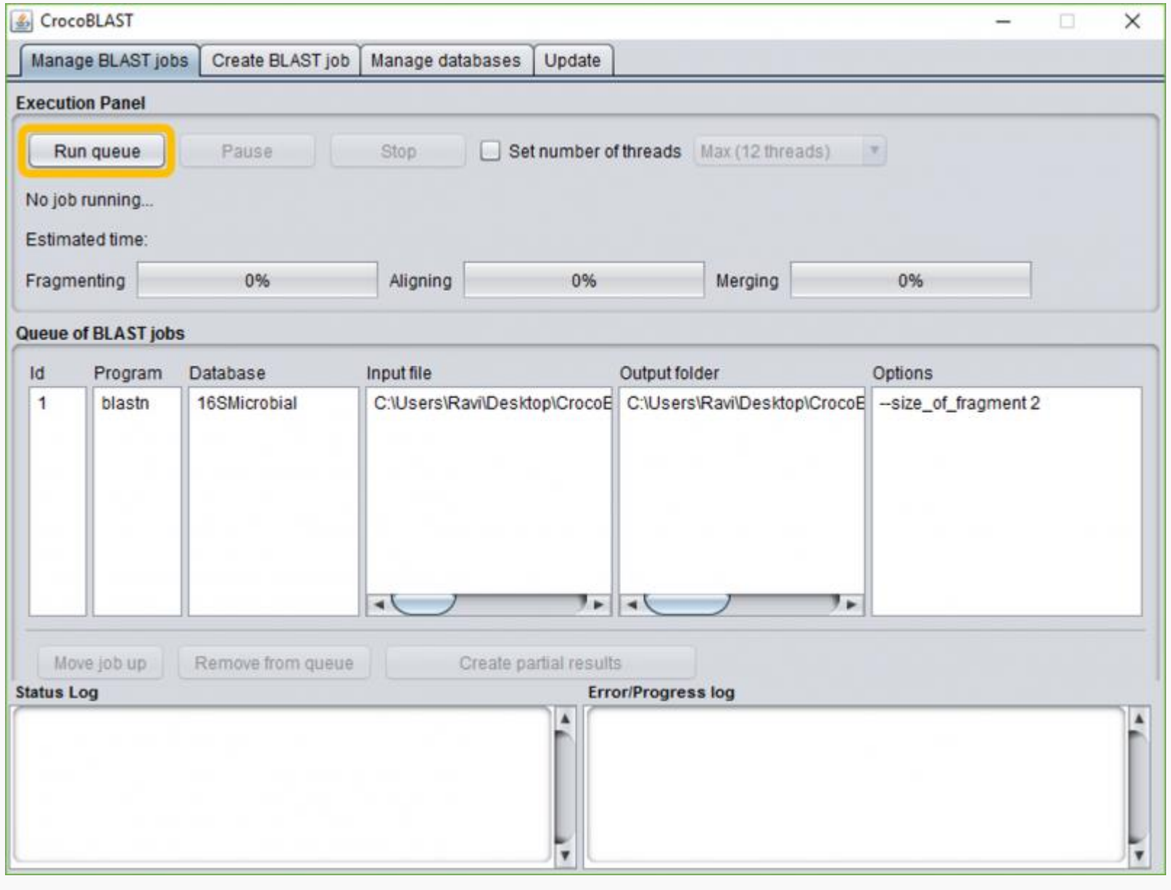
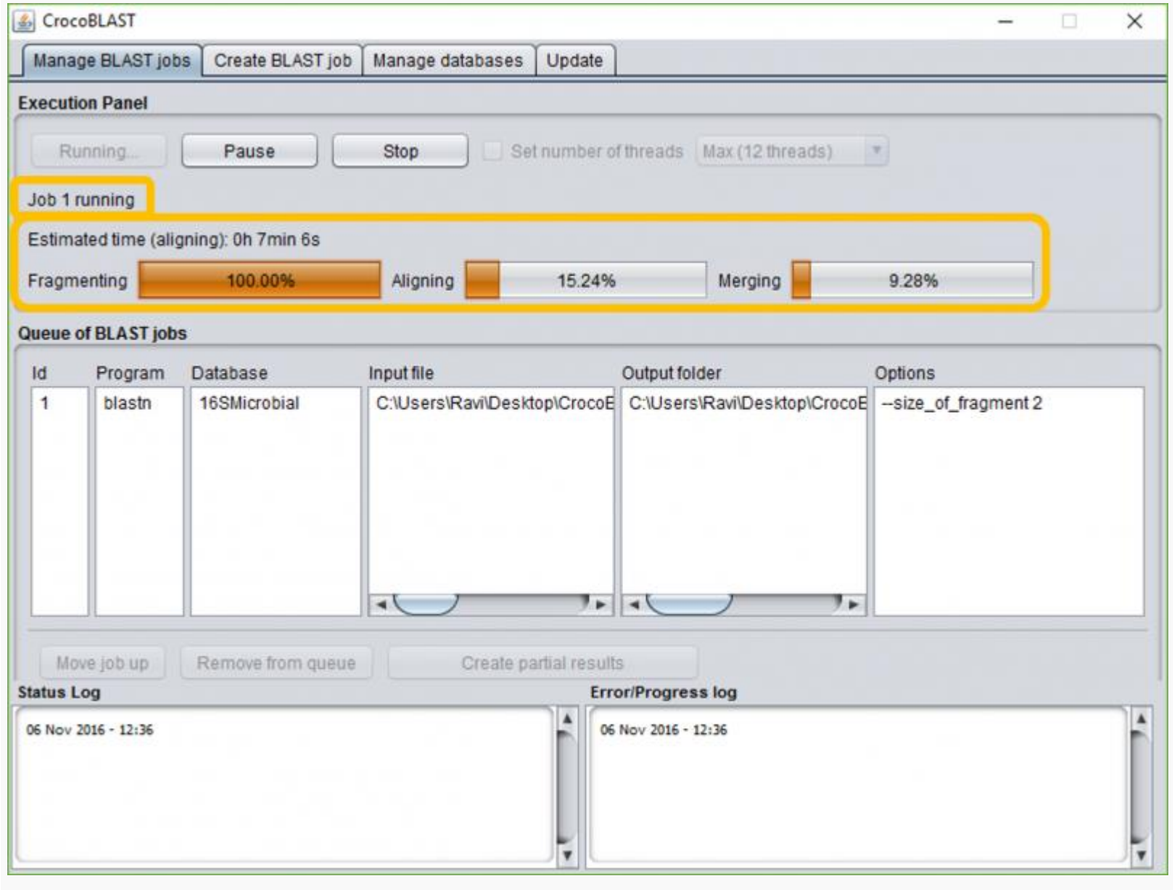
Say you have *created one or more BLAST jobs* and are ready to BLAST them. All you need to do is:

a) Select the “Manage BLAST jobs” tab.



b) (Optional) Select the number of threads to be used.



c) Click on the “Run queue” button.	**d)** You can now inspect the progress of the Jobs on your queue until they finish.
	

In case you do not set the number of threads, CrocoBLAST will detect how many threads your computer has and assign all of them to CrocoBLAST. That is a good option if you want to dedicated all the computer resources for achieving maximum performance, and if no other intense calculations or programs will be running concurrently to CrocoBLAST on the assigned computer.

Assigning a specific number of threads is useful if you are using a shared computer that concurrently runs different calculations or programs, has several users, or if you were allowed by the manager of a server to use only a certain number of threads in that computer.

The "Run queue" commands tells CrocoBLAST to sequentially run **all jobs in queue** by in small fragments, assigned to all available cores and reassigned as soon as core becomes free until all fragments of all jobs are finished.

The execution process is divided into three steps:

```
"Fragmenting", when the input file is fragmented into smaller parts;
"Aligning", when the input file is being aligned using BLAST against the specified database;
"Assembling", when the output files corresponding to all fragments are merged together in a final output identical to that of BLAST.
```

Each step has its own progress bar and time estimation.

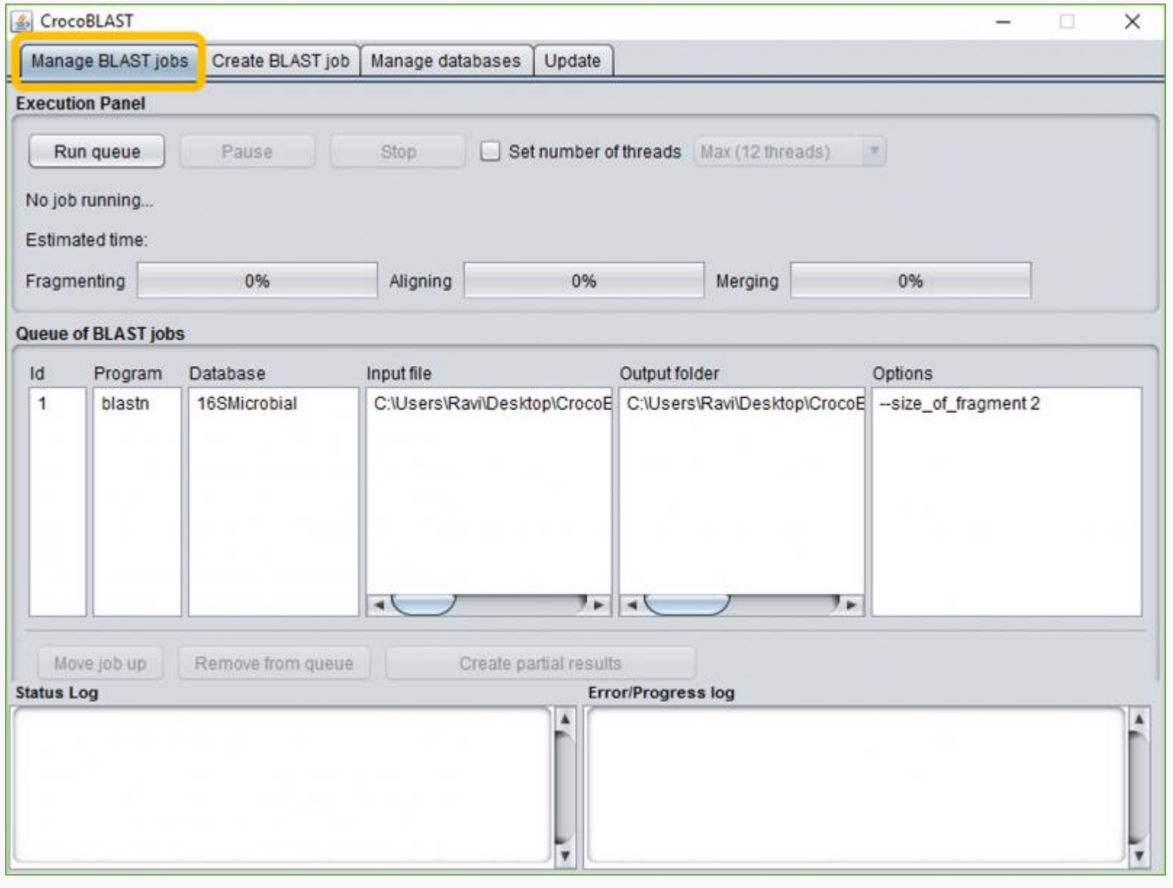
If CrocoBLAST was assigned a single thread, each step runs sequentially after the previous one is finished. However, if you are running CrocoBLAST with several threads, the Alignment will start as soon as a certain number of fragments have been generated, which usually happens in a few seconds, and both stages run concurrently. Similarly, given that more than 1 thread was assigned to CrocoBLAST, the assembling stage will start as soon as the fragmenting stage finishes, running concurrently with the alignment stage.

3.2 Pausing, stopping, and resuming

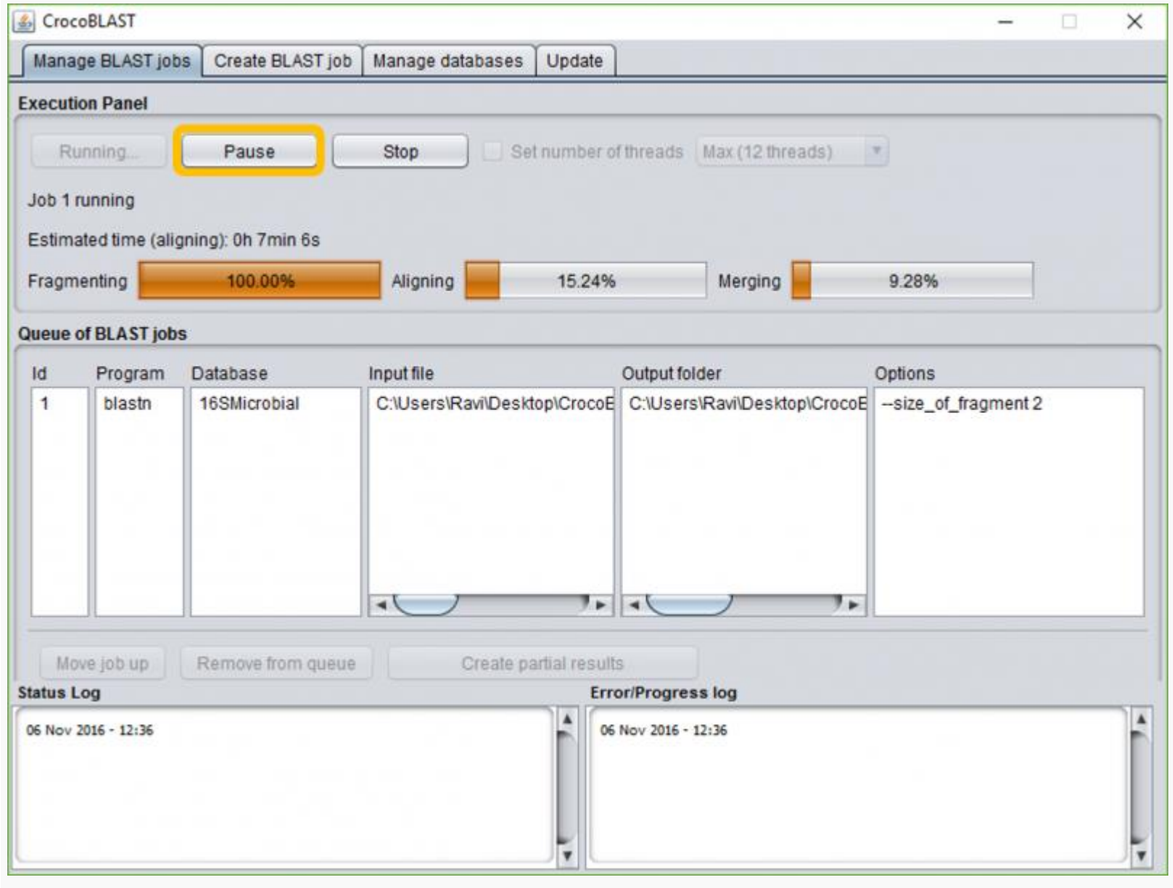
You can easily stop or pause the execution at any time. The difference between *pause* and *stop* rests with how long you are willing to wait before your computational resources become available, and how much partial output you need.

To interrupt the execution in a controlled way, waiting for all currently active fragments to finish, you should use the "pause" command. In about 5 seconds (in case you did not manually set a fragment size; otherwise it can take more, or less time), your CrocoBLAST processes will all be interrupted and all your progress, from the finished fragments and from the ones that were active when the "pause" command was sent, will be saved.

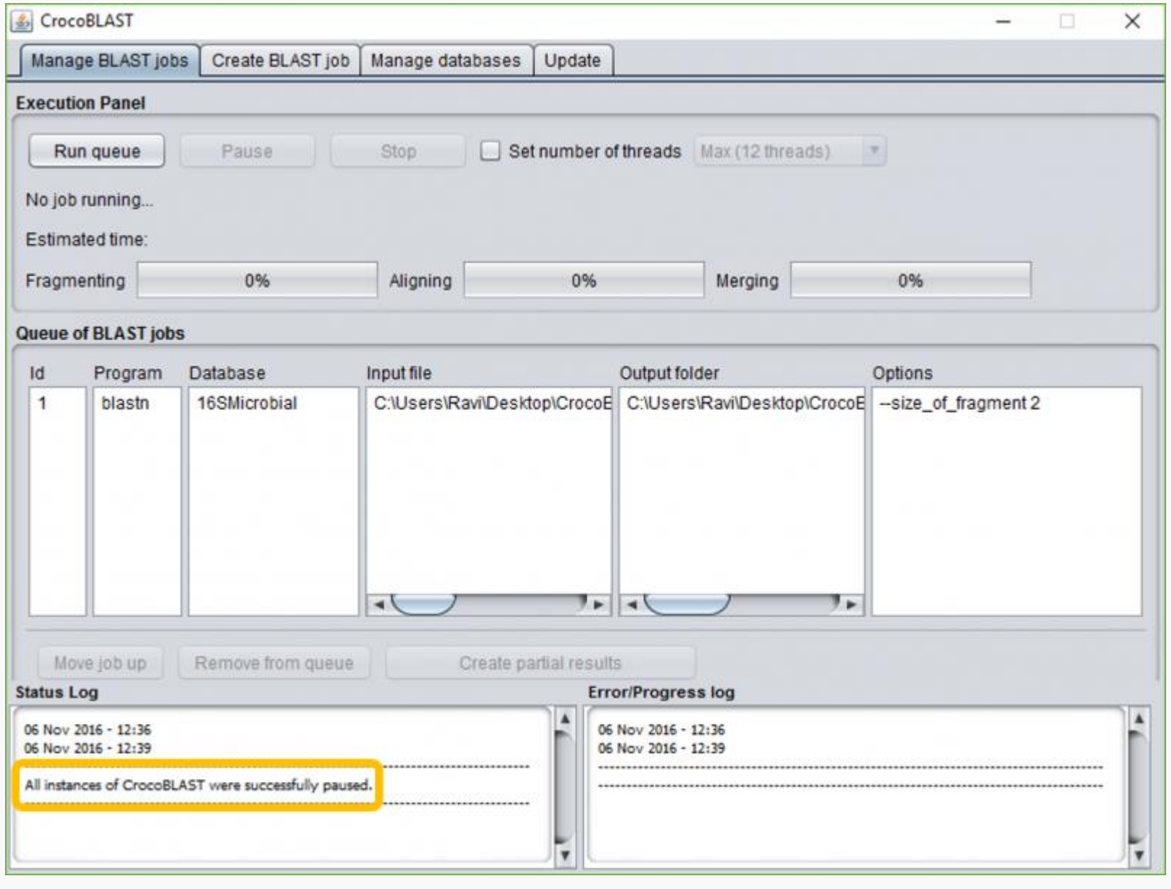
a) Select the “Manage BLAST jobs” tab.



b) Press the “Pause” button.

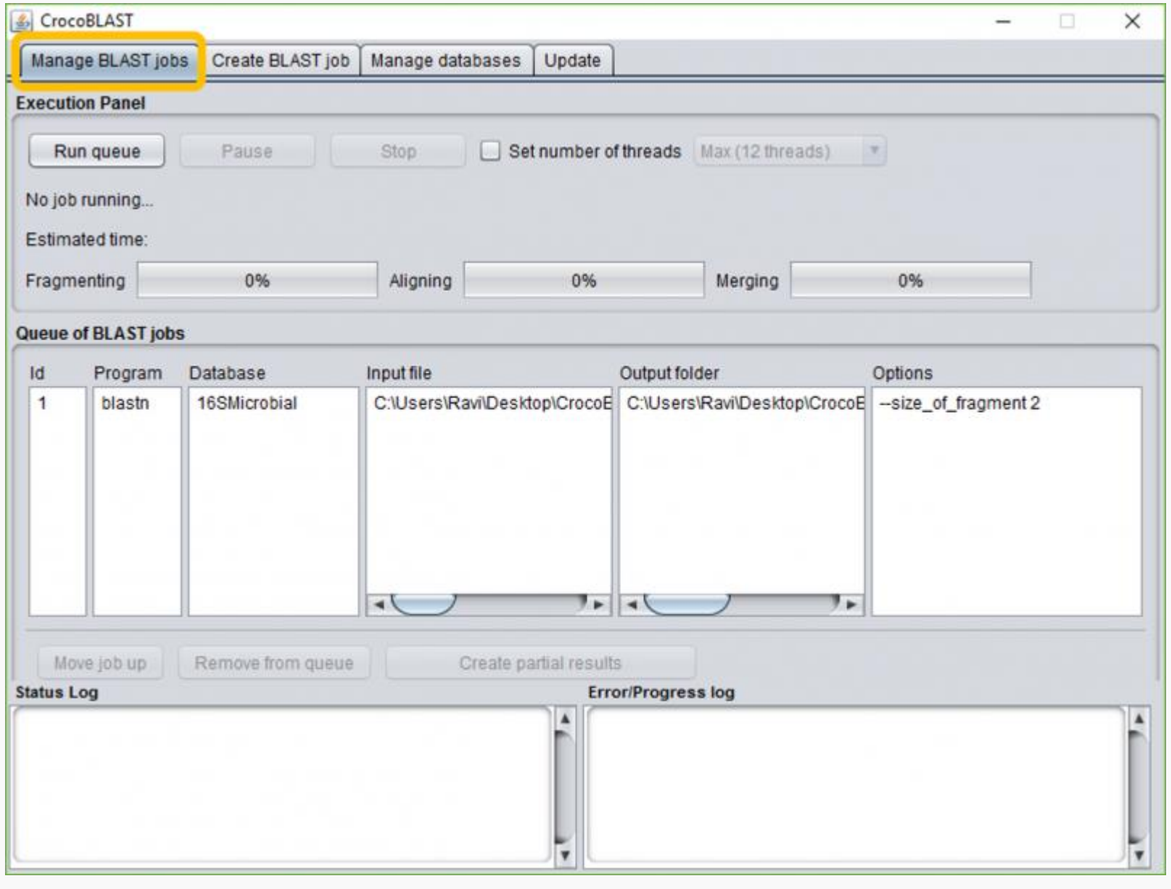


c) Wait a few seconds until you get the confirmation message. That is all!

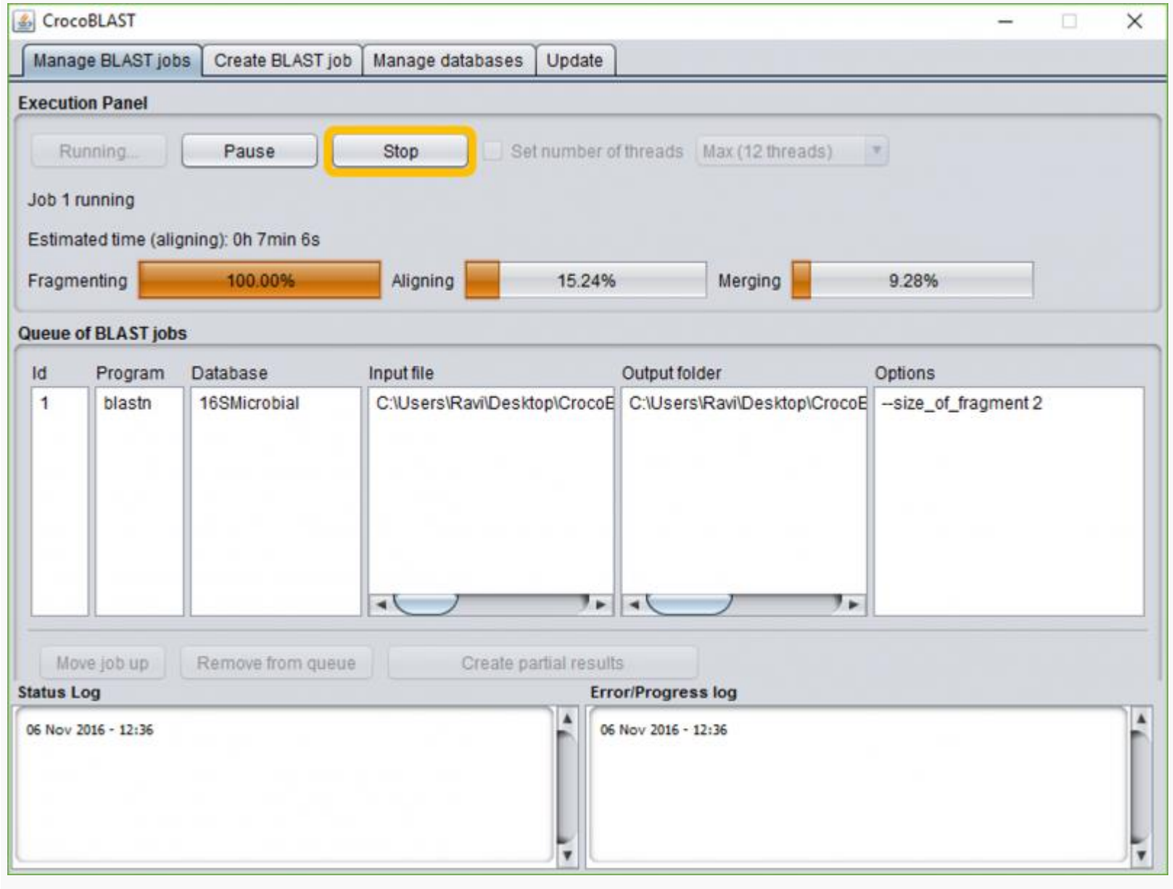


To immediately interrupt CrocoBLAST execution, freeing up the memory and cores you should use the command "stop". Although this option also saves your progress, performing the stop command will make you lose the progress of the fragments currently being aligned (which typically corresponds to 5 seconds of execution in case you did not manually set a fragment size).

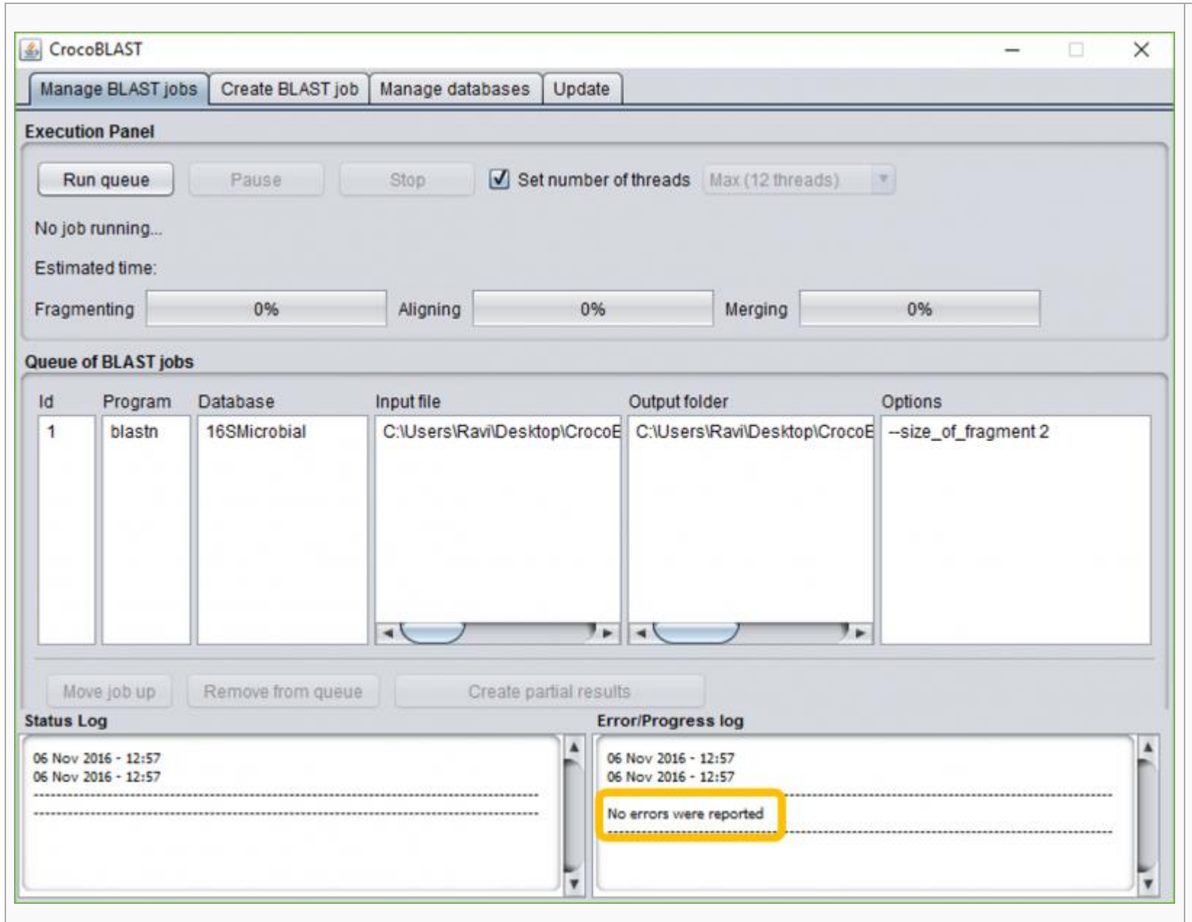
a) Select the “Manage BLAST jobs” tab.



b) Press the “Stop” button.



c) CrocoBLAST will be stopped immediately. You will receive a report in case any error happens.



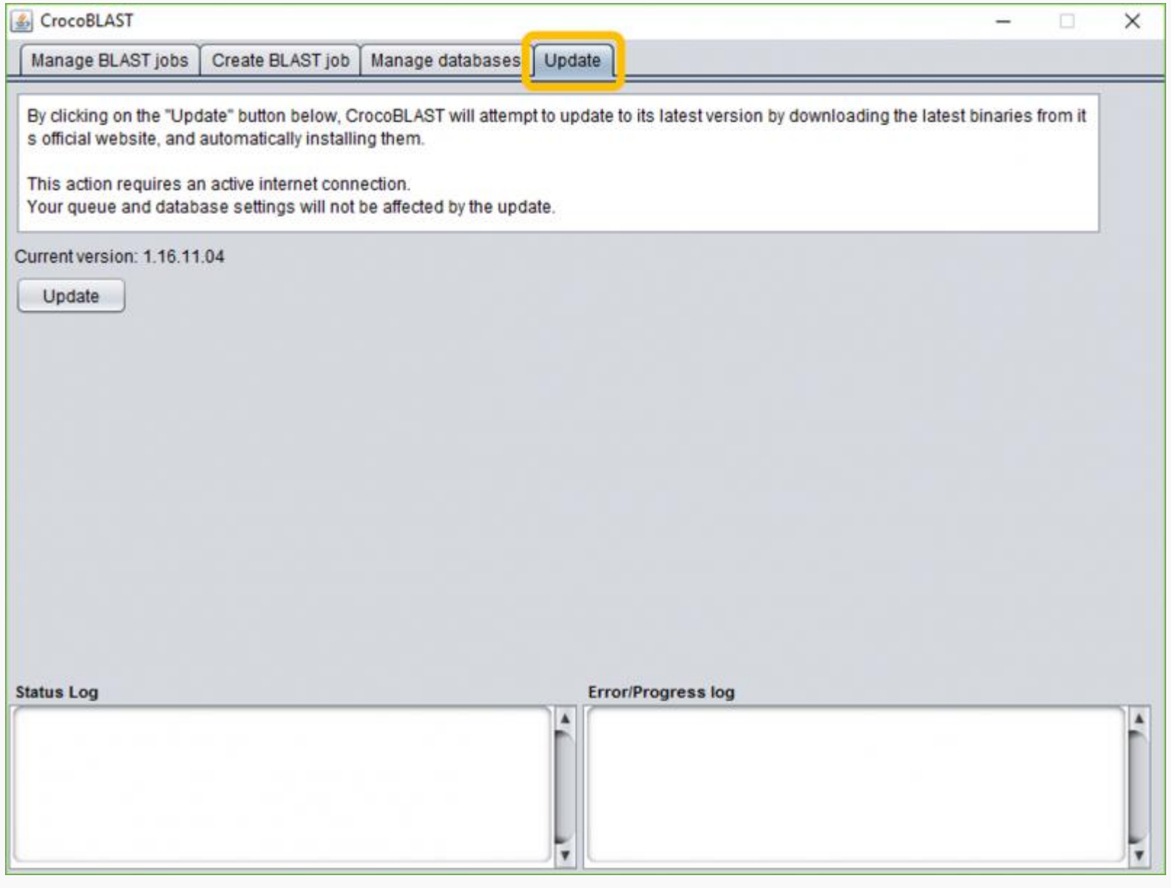
To resume the CrocoBLAST jobs all you need to do is to perform the "run" command as described in example "3.1".

CrocoBLAST will then automatically detect the current state of each job in the queue, and continue from where it left off, resuming the first job on the queue from whichever state it was.

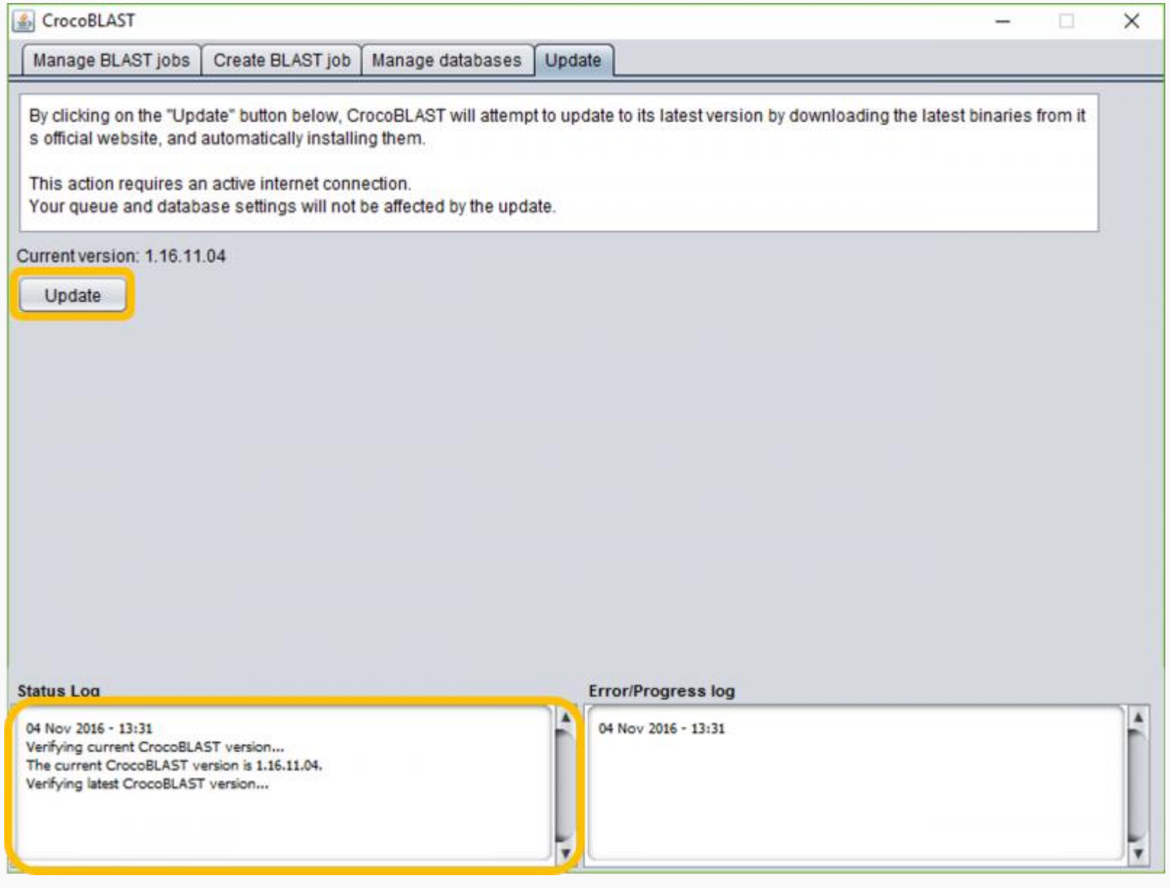
4. CrocoBLAST update

CrocoBLAST has a built-in update function that allows you to keep CrocoBLAST always up to date. Given that you have internet access, all you need to do is to type:

a) Select the “Update” tab



b) Click the “Update” button and follow the log messages. CrocoBLAST may restart during the update process



With that command, CrocoBLAST will check for newer versions automatically update if a newer version is found. CrocoBLAST updates will not affect your queue, indexed databases, or the progress of your BLAST jobs.

IV) Using CrocoBLAST (Command Line)

CrocoBLAST is built to help you plan your BLAST jobs and run them efficiently. CrocoBLAST operates with the concept of *queue*, which is basically a list of BLAST jobs scheduled to run. Thus, you can plan several BLAST jobs and let CrocoBLAST manage their execution for you.

All CrocoBLAST functionality is available via the command line utility and the graphical user interface. In fact, the graphical user interface does precisely what its name suggests: it provides an interface for the command line utility. In a nutshell, while you can interact with CrocoBLAST via simple commands, you may also use the interface to generate the commands or read the output of such commands.

The following guide contains step-by-step instructions for using CrocoBLAST in the **command line** mode.

1. Manage BLAST jobs

The efficiency of CrocoBLAST lies in its ability to parallelize the execution of your BLAST jobs. This is related to breaking each big calculation into smaller pieces, and then organizing the execution of the pieces. Depending on the job, CrocoBLAST may benefit from using smaller or larger smaller pieces. CrocoBLAST automatically detects the ideal size for breaking the input file and does it for you. Alternatively, the size of each input file fragment can be manually specified as described in examples "1.3" and "1.4."

1.1. Creating BLAST jobs

As already mentioned, BLAST takes an input file with unknown sequences and aligns each such sequence against a database of known sequences. To create a job, you must first specify the BLAST program you plan to use, which depends on the nature of the unknown sequences in your input file, and the nature of the sequences in the reference database. Then, you need to specify the name of the *database* listed in the CrocoBLAST index (more details on that below) that contains the reference sequences you wish to use. Finally, provide the input file and the location where you want CrocoBLAST to place the output files. Keep in mind that the output files may be quite large. Finally, if you want to change the default BLAST settings, you can do so by specifying the names and values of the BLAST options of interest.

1. `crocoblast -add_to_queue blast_program database input_file output_folder`
2. `crocoblast -add_to_queue blast_program database input_file output_folder --blast_options option1 value1 option2 value2 ...`
3. `crocoblast -add_to_queue blast_program database input_file output_folder --size_of_fragment size_in_kb`
4. `crocoblast -add_to_queue blast_program database input_file output_folder --size_of_fragment size_in_kb --options option1 value1 option2 value2 ...`

If no "--blast_options" are specified, CrocoBLAST will use all BLAST default values.

If no "--size_of_fragment" is specified, CrocoBLAST will take up to 5 seconds to auto-detect the ideal size for that job on your computer.

Note that, CrocoBLAST does not accept that the "--size_of_fragment" argument to be placed after "--blast options". If you intend to use both settings, please follow example number 1.4.

When you create a BLAST job, CrocoBLAST automatically assigns each BLAST job a unique job ID, and updates the CrocoBLAST queue.

1.2 Removing BLAST jobs

In case you want to remove a BLAST job from the queue, you can do so by referring to it using its job_id. The job_id is informed every time a job is added to the queue, but you can also learn it using the command "status" described in the example "1.4.1".

1. `crocoblast -remove_from_queue job_id`
2. `crocoblast -remove_from_queue job_id_1 job_id_2 ...`

The first command will remove only one CrocoBLAST job from the queue, while the second command can be used for removing several jobs at once.

1.3 Verifying the state of the queue

While CrocoBLAST operates with the concept of queue, it is important to note that only one job (the first on the queue) is active at any given time. You can check the current state of the CrocoBLAST queue using the command:

1. `crocoblast -status`

This will provide you with information regarding which jobs are on queue, with full details regarding the job ID and BLAST setup, as well as a description about the progress of each job. The progress of each job is described in three main directions: fragmentation of the input file, alignment, and assembly of results. In case CrocoBLAST is running when the "status" command is submitted, the estimated time for finishing the first job on the queue is also informed.

1.4 Moving a BLAST job to the top of the queue

As only one job (the first on queue) is active on CrocoBLAST at each time, moving another job to the top of the queue can be useful for changing the order in which the jobs will be executed. You can move a job to the top of the CrocoBLAST queue using the commands:

1. `crocoblast -move_top_queue job_id`
2. `crocoblast -move_top_queue job_id_pos_1 job_id_pos_2 ...`

2. Manage databases

2.1 Listing indexed databases

The first time you add a database into CrocoBLAST, it will be set as an internal entry with a given name, so that in the future it is easier for you to access this database. You can see which databases are already indexed in CrocoBLAST by typing:

1. `crocoblast -list_databases`

This command will separately list nucleotide and protein databases.

2.1. Retrieving and adding a database from the NCBI servers

In the most typical scenario, you will use the established reference sequence databases maintained by NCBI. CrocoBLAST allows you to specify the name of such a database, and will download or update the database for you:

1. `crocoblast -add_database --ncbi_download ncbi_database_name output_folder`
2. `crocoblast -update_ncbi_database ncbi_database_name output_folder`

Using this command will create a database entry named "ncbi_database_name" in CrocoBLAST.

When adding or updating a database in this manner, you need not worry about the format of the database, as NCBI provides pre-formatted database files.

2.2. Adding a pre-formatted database from your computer

If you have already downloaded the databases from NCBI, or if you do not have internet connection, you may add to the CrocoBLAST index database files stored on your computer. The following command can be used when you have pre-formatted database files (e.g., psq or nsq):

1. `crocoblast -add_database --formatted_db database_file.nsq`
2. `crocoblast -add_database --formatted_db database_file.psq`
3. `crocoblast -add_database --formatted_db database_file.00.nsq`
4. `crocoblast -add_database --formatted_db database_file.00.psq`

Using this command will create a database entry named "database_file" in CrocoBLAST.

Note that CrocoBLAST will assume that all database files related to the given ".nsq/.psq" file are contained in the same folder (which is most likely the case).

2.3. Adding a database from a FASTA/FASTQ sequence file in your computer

If you have already downloaded the databases from NCBI, or if you do not have internet connection, you may add to the CrocoBLAST index database files stored on your computer. Remember to provide a unique and representative name for each database you add, so that it is easy to refer to the databases later. When your database is in FASTA or FASTQ format, you will need to tell CrocoBLAST the **type of sequence** it will find in the file:

1. `crocoblast -add_database --sequence_file nucleotide fasta_file database_name output_folder`
2. `crocoblast -add_database --sequence_file protein fasta_file database_name output_folder`
3. `crocoblast -add_database --sequence_file nucleotide fastq_file database_name output_folder`
4. `crocoblast -add_database --sequence_file protein fastq_file database_name output_folder`

Using this command will create a database entry named "database_name" in CrocoBLAST.

Formatted BLAST database files will be generated in the specified output folder.

2.4 Removing a database

In case there is a missing entry in a database set for CrocoBLAST, or you have a new version of some database which name is already in use, it might be useful to remove a database from CrocoBLAST. However, you should not that this action will not delete the database files, but only remove them from CrocoBLAST managing system. For deleting the database files (which can save you a significant amount of hard disk space), you should check its location and manually delete the corresponding database files.

1. `crocoblast -remove_database nucleotide database_name`
2. `crocoblast -remove_database protein database_name`

In case you accidentally removed a database by mistake, do not worry: as CrocoBLAST does not automatically delete the database files, you can re-add your database to CrocoBLAST with the commands in example 2.2.

3. Managing CrocoBLAST execution

The efficiency of CrocoBLAST lies in its ability to parallelize the execution of your BLAST jobs. This is related to breaking each big calculation into smaller pieces, and then organizing the execution of the pieces. Depending on the job, CrocoBLAST may benefit from using smaller or larger smaller pieces. CrocoBLAST automatically detects the ideal size for breaking the input file and does it for you. Alternatively, the size of each input file fragment can be manually specified as described in examples "1.3" and "1.4."

3.1. Running

Say you have *created one or more BLAST jobs* and are ready to BLAST them. All you need to do is type one of the two commands:

1. `crocoblast -run`
2. `crocoblast -run --num_threads number_of_threads`

The first one will detect how many threads your computer has and assign all of them to CrocoBLAST. That is a good option if you want to dedicated all the computer resources for achieving maximum performance, and if no other intense calculations or programs will be running concurrently to CrocoBLAST on the assigned computer.

The second options lets you set how many threads CrocoBLAST will use. That is a good option if you are using a shared computer that concurrently runs different calculations or programs, has several users, or if you were allowed by the manager of a server to use only a certain number of threads in that computer.

These commands tell CrocoBLAST to sequentially run all jobs in queue by into small fragments, assigned to all available cores and reassigned as soon as core becomes free until all fragments of all jobs are finished.

The execution process is divided into "Fragmenting", when the input file is fragmented into smaller parts, "Aligning", when the input file is being aligned using BLAST against the specified database, and "Assembling", when the output files corresponding to all fragments are merged together in a final output identical to that of BLAST. Each step has its own progress bar and time estimation.

If CrocoBLAST was assigned a single thread, each step runs sequentially after the previous one is finished. However, if you are running CrocoBLAST with several threads, the Alignment will start as soon as a certain number of fragments have been generated, which usually happens in a few seconds, and both stages run concurrently. Similarly, given that more than 1 thread was assigned to CrocoBLAST, the assembling stage will start as soon as the fragmenting stage finishes, running concurrently with the alignment stage.

3.2 Pausing, stopping, and resuming

You can easily stop or pause the execution at any time. The difference between *pause* and *stop* rests with how long you are willing to wait before your computational resources become available, and how much partial output you need.

To interrupt the execution in a controlled way, waiting for all currently active fragments to finish, you should use the "pause" command. In about 5 seconds (in case you did not manually set a fragment size; otherwise it can take more, or less time), your CrocoBLAST processes will all be interrupted and all your progress, from the finished fragments and from the ones that were active when the "pause" command was sent, will be saved.

To immediately interrupt CrocoBLAST execution, freeing up the memory and cores you should use the command "stop". Although this option also saves your progress, performing the stop command will make you lose the progress of the fragments currently being aligned (which typically corresponds to 5 seconds of execution in case you did not manually set a fragment size). To pause or stop you just type the corresponding command as given below:

1. `crocoblast -pause`
2. `crocoblast -stop`

To resume the CrocoBLAST jobs all you need to do is to perform the "run" command as described in examples "3.1.1" and "3.1.2"

CrocoBLAST will then automatically detect the current state of each job in the queue, and continue from where it left off, resuming the first job on the queue from whichever state it was.

4. CrocoBLAST update

CrocoBLAST has a built-in update function that allows you to keep CrocoBLAST always up to date. Given that you have internet access, all you need to do is to type:

1. `crocoblast -update_crocoblast`

With that command, CrocoBLAST will check for newer versions automatically update if a newer version is found. CrocoBLAST updates will not affect your queue, indexed databases, or the progress of your BLAST jobs.

V) Technical details

CrocoBLAST is free to use within the conditions of its license, and has been available for download since July 2016 at http://webchem.ncbr.muni.cz/Platform/App/CrocoBLAST. There is no login requirement for downloading or running **CrocoBLAST**. For running the graphical user interface, simply double click it. For running the command line utility, please open a command line prompt (terminal in Linux, cmd in Windows).

1. Software requirements

CrocoBLAST runs on Windows and Linux, and all results are provided in the regular BLAST output format. Once you download the "zip" with all necessary **CrocoBLAST** files, all you need to do is unzip it to the desired location. As CrocoBLAST works as a wrapper for BLAST, you will need to have BLAST available on your computer before you can run CrocoBLAST. If you don't already have BLAST, please get the latest version from the NCBI website. No further requirements exist for running the command line utility of **CrocoBLAST**, or inspecting the results. In Windows, downloading files from the NCBI server requires PowerShell, which ships automatically with Windows 7 and newer, and can be installed on Windows XP and Vista. The graphical user interface requires Java 8, which is likely already installed on your computer. If not, please visit https://java.com/en/download/.

2. Hardware requirements

Because of the nature of the data being processed, it is better if your computer has at least 200 MB of free RAM per core. Nonetheless, it is possible to run CrocoBLAST on large files even if less memory is available, but you will need to manually specify a small fragment size during job submission. Furthermore, if you need to analyze NGS data, the input and output files involved in such calculations can be quite large, and therefore you will need to have sufficient space on your hard disk. NCBI databases range from 10 MB to 500 GB (whole genomes). Depending on the type of sequencing experiment you ran, your input files may range from a few kB to hundreds of GB. If you don't specify any BLAST options, the size of the output file may be up to 1500 times the size of the input file. Nevertheless, in the typical use case, requesting relevant BLAST options (e.g., provide only the first 20 hits) will greatly reduce the size of the output file.

3. Limitations

As mentioned above, all results are provided in one of three BLAST output formats (pairwise, tabular or comma-separated-values). CrocoBLAST does not currently offer facilities for graphical visualization and analysis of the BLAST results, partly due to the fact that it targets big data, and partly because there are other great free tools are already available for such purposes. We recommend that you obtain the alignments using CrocoBLAST, and then use some specialized software (e.g., MEGAN) to further extract relevant information from your data.

CrocoBLAST currently does not implement a parallelization of the BLAST calculation over the network. This aspect may be addressed in a future version of CrocoBLAST, once we have gathered sufficient information regarding the most common use case for network-distributed calculations. Your feedback is greatly valued.

Finally, while CrocoBLAST will run on most versions of Windows (XP or newer) and Linux, CrocoBLAST will not run on OS X. It is unlikely that this should change in the immediate future, but do check back with us just in case.

4. Troubleshooting

CrocoBLAST typically checks that the necessary files and permissions exist before starting the demanding BLAST calculation.

If you get into trouble while trying to run CrocoBLAST, please check the error messages, which are quite informative and should help you overcome the most common issues you are likely to encounter. If you experience further issues, please contact us and describe the problem in detail.