Sequence Conversion and Annotation with BLAST+ (seqCAB) User Guide  
Table of Contents

[1. Introduction 2](#_Toc529095542)

[1.1. Outline 2](#_Toc529095543)

[2. Software installation 2](#_Toc529095544)

[2.1. seqCAB.py 2](#_Toc529095545)

[2.2. seqCAB\_lift.py 3](#_Toc529095546)

[2.3. seqCAB\_annotation.py 3](#_Toc529095547)

[3. Running seqCAB 4](#_Toc529095548)

[3.1. Running seqCAB for BLAST searches 4](#_Toc529095549)

[3.2. Running seqCAB\_lift.py for rescue 7](#_Toc529095550)

[3.3. seqCAB\_annotation.py for FASTA annotation, BLAST filtering and taxonomic assignment. 7](#_Toc529095551)

[**3.2.0.** **seqCAB\_annotation.py input and output files** 7](#_Toc529095552)

[**3.2.1.** **seqCAB annotation without parameter options** 9](#_Toc529095553)

[**3.2.2.** **seqCAB annotation parameter (flag) options** 9](#_Toc529095554)

{**Sequence Conversion and Annotation with BLAST+ (seqCAB)} (NRL-SF-009) SOFTWARE**

**W. Judson Hervey, IV, Naval Research Laboratory; Cheryl Ames, National Research Council Postdoctoral Fellow at the Naval Research Laboratory; Amina Jackson, Naval Research Laboratory (Former NRL Employee)**

**DISTRIBUTION STATEMENT**

**DISTRIBUTION A. Approved for public release: distribution unlimited.**

**Redistributions of source and binary forms, with or without modification, are permitted if redistributions retain the above distribution statement and the following disclaimer.**

**DISCLAIMER**

THE SOFTWARE IS SUPPLIED “AS IS” WITHOUT WARRANTY OF ANY KIND.

AS THE OWNER OF THE SOFTWARE, THE UNITED STATES, THE UNITED STATES DEPARTMENT OF DEFENSE, AND THEIR EMPLOYEES: (1) DISCLAIM ANY WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, TITLE OR NON-INFRINGEMENT, (2) DO NOT ASSUME ANY LEGAL LIABILITY OR RESPONSIBILITY FOR THE ACCURACY, COMPLETENESS, OR USEFULNESS OF THE SOFTWARE, (3) DO NOT REPRESENT THAT USE OF THE SOFTWARE WOULD NOT INFRINGE PRIVATELY OWNED RIGHTS, (4) DO NOT WARRANT THAT THE SOFTWARE WILL FUNCTION UNINTERRUPTED, THAT IT IS ERROR-FREE OR THAT ANY ERRORS WILL BE CORRECTED.

PORTIONS OF THE SOFTWARE RESULTED FROM WORK DEVELOPED BY OR FOR THE U.S. GOVERNMENT AND ARE SUBJECT TO THE FOLLOWING LICENSE: THE GOVERNMENT IS GRANTED FOR ITSELF AND OTHERS ACTING ON ITS BEHALF A PAID-UP, NONEXCLUSIVE, IRREVOCABLE WORLDWIDE LICENSE IN THIS COMPUTER SOFTWARE TO REPRODUCE, PREPARE DERIVATIVE WORKS, TO PERFORM OR DISPLAY ANY PORTION OF THAT WORK, AND TO PERMIT OTHERS TO DO SO FOR GOVERNMENT PURPOSES.

# Introduction

The intended purpose of this document is to serve as a user guide for ‘seqCAB’ BLAST+ utility users. The core design of seqCAB combines simplicity with efficiency for users seeking quick and qualitive assessment of high throughput sequence datasets, irrespective of organism (Eukaryote or Prokaryote) or molecule type (protein or nucleic acid), in the current era of exponential ‘omics’ data growth. This guide explains how to use seqCAB, a BLAST+ utility that optimizes performance by 90% to 110% on High Performance Computing (HPC) platforms without altering the functionality or results of NCBI BLAST+. Additionally, this document covers the use of a tool developed to annotate seqCAB BLAST results – called ‘seqCAB\_annotation’. Combined features of seqCAB include FASTA format conversion from over 25 commonly used data formats, rapid BLAST execution against local database (ncbi nt or nr, or user custom databse), FASTA file annotation with BLAST top hits, annotation report generation (TSV/CSV) for quick manipulation and interpretation of BLAST results, full taxonomy assignment of top hits based on user-designated thresholds (% identity, occurrence and sequence length) for downstream data visualization. This document can be found at: https://github.com/aminakjackson/seqCAB/blob/master/seqCAB\_User\_Guide

## Outline

This section provides an overview of how ‘seqCAB’ BLAST job performance by utilizing all allocated resources for job distribution. seqCAB.py implementation with Message Passing Interface (MPI) uses MPI collectives scatter, gather and broadcast to distribute queries on multiple cores and create multiple instances of BLAST reference database that are searched in parallel. At the core level, seqCAB uses python multiprocessing object pool for further parallelization, thereby increasing the number of queries searched at one time. Using the seqCAB.py python script with python supported module MPI4py, seqCAB utilizes seqCAB MPI implementation for data format conversion and BLAST search. The Annotation tool seqCAB\_annotation.py uses Python data advanced modules Pandas to extract pertinent annotation and taxonomic information from the BLAST output file through python multiprocessing for parallelization.

# Software installation

This section covers basic seqCAB software installation. More detailed instructions for software compilation is described in README.md: <https://github.com/aminakjackson/seqCAB/blob/master/README.md>

## seqCAB.py

Running the seqCAB.py script for data conversion and BLAST search requires downloading and installing the following software.

NCBI BLAST+

Download either ncbi-blast-2.2.31+-src.tar.gz or ncbi-blast-2.2.30+-src.tar.gz. Both versions were tested successfully on our SGI system compiling with c++ compilers. The latest versions of BLAST+ could only compile with gcc, and therefore does not work with MPI.

<ftp://ftp.ncbi.nlm.nih.gov/blast/executables/blast+/2.2.31/ncbi-blast-2.2.31+-src.tar.gz>

* Python 3

Download python 3 at <https://www.python.org/downloads/source/> . Although python 2 would work, we recommend python 3.

* Python Modules: Follow the installation in the link below or use the installation instructions in our readme.
  + Download mpi4y version 2 for MPI python MPI module. <https://mpi4py.readthedocs.io/en/stable/install.html#using-pip-or-easy-install> or <https://pypi.python.org/pypi/mpi4py> ; install mpi4py by executing

$ pip install mpi4py

* + Download pandas module using this link <https://pandas.pydata.org> ; install pandas by executing

$ pip install pandas

* + Install biopython by following instructions in this link <https://biopython.org> or use conda

$ conda install biopython

Note: The quickest way to install python and the required modules is by installing Anaconda3: <https://repo.anaconda.com/archive/Anaconda3-5.2.0-Linux-x86_64.sh> , and then using conda to install pandas and mpi4py modules.

## seqCAB\_lift.py

seqCAB\_lift.py uses Biopython and pandas modules.

## seqCAB\_annotation.py

Running the seqCAB\_annotation.py script only requires the pandas module (see above).

# Running seqCAB

This section describes how to run seqCAB for data format conversion, optimized BLAST job distribution, and FASTA annotation and taxonomy assignment.

## Running seqCAB for BLAST searches

First, we will make the following assumptions:

* You are running seqCAB on HPC cluster (SGI).
* You have compiled a running version of BLAST c++ compliers (i.e. icc, icpc, mpiicc and mpiicpc). For information on how to compile BLAST on HPC follow readme.md instructions: <https://github.com/aminakjackson/seqCAB/blob/master/README.md>.
* You are familiar with the functionality of BLAST. For information on how BLAST works go to NCBI website: <https://www.ncbi.nlm.nih.gov/books/NBK279668/> and/or <https://www.ncbi.nlm.nih.gov/books/NBK279690/>

Depending on your HPC environment, you may need to execute seqCAB by requesting an interactive session or using a job scheduler (qsub) system, e.g., PBS. Keep in mind that the method for requesting batch interactive sessions differs based on HPC system configuration and resources. Please consult your system administrator for clarification of queue types, and walltime and node allocations. Below is an example of an interactive session request on our SGI Thunder system:

*$ qsub -l walltime=5:00:00 -q standard -A <myacc> -l ncpus=72 -I*

* -l walltime=<5:00:00> amount of time you will using the computing nodes
* -l ncpus= <72>: number of cores
* -q <standard>: name of the queue you be running. Others may include debug, PHI, etc.

After securing an appropriate session or queue, verify that the required modules use the same compilers as used to compile BLAST by executing:

$ module list

$ module load <mpi-module> (in the event the module is not yet loaded).

Set the environment variables to point to the right compliers by executing the following (depending on your shell).

$ setenv CC mpiicc or setenv CC icc

$ setenv CXX mpicpc or setenv CXX icpc

or

$ export CC=mpiicc or export CC=icc

$ export CXX=mpiicpc or export CXX=icpc

$ set CONDA\_PATH /home/myhome/anaconda3/bin

You are now ready to run your BLAST search by executing the following, interactively or within a job script (see below):

$ time mpirun -np 72 python seqCAB.py myfasta.fasta "blastp -db myblast\_database\_path/nr -num\_threads 2 -evalue 1e-5 -max\_target\_seqs 10" BLAST.out

While the Batch interactive may perform as fast as Batch submission, for jobs greater than 10,000 nucleotide or 1000 amino acid sequences, it may be essential to use batch submission to ensure sufficient allocated resources to complete the run. Here is an example job script required for Batch submission, created for Portable Batch System (PBS).

#!/bin/sh

#Start with required PBS directives

*# required for your projects account id*

#PBS -A *Project\_ID*

*# Required allocation of wall time clock for the job*

#PBS -l walltime=15:00:00

*# Required allocation for desired number of cores.*

#PBS -l ncpus=72

*# Required request for the queue to run in*

#PBS -q standard

*# job name optional.*

#PBS -N *myjob*

cd $WORKINGDIR

*#Load modules that have the C++ compilers used to compile BLAST*

module load intel\_mpi

module load intel-7

#*Set to Anaconda that has python modules needed for seqCAB*

set CONDA\_PATH /home/myhome/anaconda3/bin

For more information on using PBS visit this link <http://www.arc.ox.ac.uk/content/pbs-job-scheduler>

The following points are worth noting:

* Normally when running BLAST, -query is used to provide the query file and -out to designate an output file. However, when writing the job submission script to run BLAST with seqCAB.py, the usual “-query” flag is replaced by “-q” (fallowed by the input file, e.g., FASTQ, FASTA, etc), the “-b” flag indicates BLAST commands will follow next inside closed quotations, after which the usual BLAST “-out” flag is replaced with “-p”.
* With the exception on the query file name and the output file name (noted above), all other BLAST parameters must be within closed quotations. Follow BLAST guidelines in the link provided earlier for a full list of BLAST parameters available.
* You do not necessarily need to export the icc & icpc compilers for each run, but it is good practice to echo them out and make sure they are always within the BLAST path.
* If your query file is not in FASTA format, the specific format extension must be designated after a “.” to allow seqCAB to convert it into a FASTA file, using FASTA-reader, the default input format for BLAST+. For example, when running the example command to call seqCAB.py and BLAST above, designating the input sequence file as myinputfile.fastq will allow seqCAB to convert your sequence input file from FASTQ format to FASTA format.

Once the job starts running, you will immediately see a generated FASTA file in your working directory. Subsequently, two directories will immediately be created in your working directory.

* myinputfile\_BLAST\_Input\_DIR: directory with the divided input files spread across the core; number of files equivalent to number of user-assigned cores
* myinputfile\_BLAST\_Output\_DIR: directory with the output file results from BLAST searches; number of files equivalent to number of user-assigned cores (corresponding file numbers in myinputfile\_BLAST\_Input\_DIR directory)

Both the input and output files are indexed within their respective directories (explanation above). The input index file has a corresponding index for its BLAST output results in the output file, for consistency and easy tracking, or if you wish to view partial results before the entire job is completed.

## Running seqCAB\_lift.py for rescue

In the event of premature termination of a seqCAB homology search, due to unforeseen circumstances, e.g., HPC system outage or wall-time exceeded, the custom script ‘seqCAB\_lift.py’ permits the user to pick up the BLAST job where it left off, rather than starting the run from the beginning, generating a single final output directory of all BLAST hits.

As a preventative measure, when submitting the initial job script for seqCAB homology search and annotation (-A) assignment, users can pre-emptively call upon seqCAB\_lift.py by assigning the -j flag, followed by -walltime <i> (actual anticipated walltime (mins) minus 5) and the name of the job script to be resubmitted in the event of a crash. Running the rescue script seqCAB\_lift.py is semi-automated. If activated (i.e., your BLAST job is not completed with 5 mins remaining of the assigned wall-time), ‘seqCAB\_lift.py’ generates a FASTA file of all unprocessed sequences (i.e., sequences not yet processed by BLAST) that permits the user to resubmit the seqCAB job on just the unprocessed\_lift.FASTA, rather than rerunning BLAST from the start on the entire dataset.

## seqCAB\_annotation.py for FASTA annotation, BLAST filtering and taxonomic assignment.

Both ‘seqCAB\_annotation.py’ (standalone version) and ‘seqCAB\_annotation1.py’ (automated version) use the Python Pandas-Data Frame module (Pandas) to extract pertinent information from top hits in the BLAST output file that is used to populate Annotation report(s) (TSV/CSV format).

### **seqCAB\_annotation.py input and output files**

**Input files**

The following input files are required:

1. BLAST output directory consisting of your BLAST search results, from seqCAB.py step described above (myqueryfile \_BLAST\_Output\_DIR). Note: You can also use a single BLAST output file (generated from seqCAB.py or generic BLAST execution). Be sure to name files accordingly, as seqCAB\_annotation.py looks for the “DIR” extension at the end of the input directory or file name.
2. The original FASTA file corresponding to your BLAST results (make sure prefixes match with the BLAST output directory or file).
3. If taxonomic assignments (Super Kingdom down to species) are desired, download the taxonomy database from NCBI (optional). A script is provided (taxonmy\_conversion.py) to convert the NCBI taxonomy database into a TSV/CSV format utilized by seqCAB\_annotation.py.

We recommend keeping all input files in the same directory in which you are running the script.

Things to note:

* The pandas module must be installed. If you are using Anaconda, following the execution of seqCAB.py (section 3.1 above), pandas will already be installed. See readme.md: <https://github.com/aminakjackson/seqCAB/blob/master/README.md>
* Python 3 is required to run seqCAB\_annotaion.py

**Output files**

seqCAB\_annotation.py always generates an output directory (BLAST\_Output\_DIRmyfasta\_DIRBLAST\_annotation\_DIR) containing files that are indexed according to the originally indexed files in the seqCAB.py output directory. These files are then concatenated into several corresponding file types. Your output file count and type will vary depending on user-assigned parameter (flag) options (i.e., taxonomy, % identity, length and occurrence).

The six default output files, generated without assigning additional flags (section 3.2.1) are:

* myfasata\_allblastered\_hits.tsv
* myfasta\_no\_Blast\_hits.txt
* myfasta\_filtered.tsv
* myfasta\_Species\_unfiltered\_occ.tsv
* myfasta\_Species\_filtered\_occ.tsv
* myfasta\_blast\_annotated.fasta

When the taxonomy flag (section 3.2.2) is used, two additional files are generated:

* myfasta\_lineage.tsv
* myfasta\_lineage.tsv\_complet\_taxonomy.tsv

### **seqCAB annotation without parameter options**

Example script to run seqCAB annotation without any parameter options:

$ python seqCAB\_annotation.py myBLASToutput\_DIR myfasta.fasta

This will result in the default files below:

* myfasta\_annotated.fasta: annotated FASTA file in which the query sequence header is appended with the identity of the top BLAST hit, e-value and identity score
* myfasata\_allblastered\_hits.tsv: TSV/CSV file of all BLAST hits including their index from top to bottom based on e-value
* myfasta\_no\_Blast\_hits.txt: text file of query ids with no BLAST hits
* myfasta\_unfiltered\_occ.tsv: TSV/CSV file containing the species name and how many times it occurred as a top hit over all your BLAST results.

### **seqCAB annotation parameter (flag) options**

* -n <integer value>: number of threads
* -t <file csv formatted value>: taxonomy
* -i <integer value>: identity score cutoff based on BLAST top hit results
* -o <integer value>: species occurrence cutoff based on top hit BLAST results
* -l <integer value>: sequence length cutoff based on original FASTA file

When -n flag is used, python multiprocessing module will create n (user designated) number of threads for faster processing and job parallelization. Note: This is more efficient if the dataset has greater than1000 sequences. Avoid assigning too many processes, as this may create issues related to memory over-assignment.

An example script for running seqCAB annotation with -n option:  
$ python seqCAB\_annotation.py myBLASToutput\_DIR myfasta.fasta -n 12

12 is the number of threads/processes that will be used by the program.

When the -t flag is used, seqCAB\_annotation.py matches BLAST annotations with corresponding taxonomic assignments in the NCBI taxonomy database (taxonomy\_db.csv), which we downloaded and converted to CSV format using the ‘taxonomy\_conversion.py’ script. Both the script to convert the NCBI taxonomy database to CSV format and the readily available taxonomy\_db.csv (latest version) files are available in the seCAB github repository.

An example script for running seqCAB annotation with -t option:

python seqCAB\_annotation\_mpi\_v1.py myBLASToutput\_DIR myfasta.fasta -t taxonomy\_db.csv

Selection of the -t parameter will result in the following output files, in addition to the default files (listed above):

* myfasta\_lineage.tsv: TSV/CSV file containing unique species retrieved from BLAST results with corresponding full taxonomic assignment from super kingdom to species-level.
* myfasta\_lineage.tsv\_complet\_taxonomy.tsv: TSV/CSV file containing complete taxonomic assignment, from species to super kingdom, together with fasta sequence information such as query ids and sequence length.
* myfasta\_taxonomy\_annotated.fasta: FASTA file in which the query sequence header is appended with the identity of the top BLAST hit, e-value and % identity score from top hits

With the flags -i (% identity), -l (query sequence length), -o (occurrence of species-annotated sequence), users can specify the threshold above which sequences will be retained for taxonomic assignment, following seqCAB annotation. BLAST annotation results that are at or below these user-defined cut-offs will not be included in all the outputted filtered files (see above), but will remain among sequences in the unfiltered files.

An example script for running seqCAB annotation with -t option, and additional flags:

$ python seqCAB\_annotation\_mpi\_v1.py myBLASToutput\_DIR myfasta.fasta -i 90

$ python seqCAB\_annotation\_mpi\_v1.py myBLASToutput\_DIR myfasta.fasta -l 550

$ $ python seqCAB\_annotation\_mpi\_v1.py myBLASToutput\_DIR myfasta.fasta -o 10

Users can run the options separately, or together in various combinations.

An example script for running seqCAB annotation with -t option, and additional flags:

$ python seqCAB\_annotation\_mpi\_v1.py myBLASToutput\_DIR myfasta.fasta -t taxonomy\_db.csv -i 90 -l 550 -o 10

Or

$ python seqCAB\_annotation\_mpi\_v1.py myBLASToutput\_DIR myfasta.fasta -t taxonomy\_db.csv -l 550

Though the order of the input file and directory are fixed, the order in which flags are designated with respect to the taxonomy database does not matter.

**Automated version of the Annotation feature**

To save additional time, the Annotation and Taxonomy steps can be automated, i.e., run simultaneously with the BLAST homology search (seqCAB.py), by assigning the -L flag, followed by the aforementioned user-assigned flags, within the seqCAB.py job submission script, to calls the automated version ‘seqCAB\_annotation1.py’.