

## PERL Programs

Program Name	Function	Input file(s)	PERL module requirements
blastparser_csr1.0.pl	Parse the output of a standalone BLAST search	BLAST output	BioPerl <sup>a</sup>
discard_shortseqs1.0.pl	Discard sequences below a specified length	FASTA	BioPerl <sup>a</sup>
fasta_length1.0.pl	SequenceID and length (tab delimited output)	FASTA	BioPerl <sup>a</sup>
gc_percent1.0.pl	SequenceID and GC% (tab delimited output)	FASTA	BioPerl <sup>a</sup>
retrieve_sequence_subset1.0.pl	Retrieve desired sequences	SeqID list, FASTA file	BioPerl <sup>a</sup>
tag_fasta_header1.0.pl	Add a tag to the fasta header line	FASTA	BioPerl <sup>a</sup>
two_list_comparison1.0.pl	Union, intersection and unique items from two lists	Two lists	List::Compare <sup>b</sup>

<sup>a</sup>Instructions for downloading and installing BioPerl are available at [http://www.bioperl.org/wiki/Getting\\_BioPerl](http://www.bioperl.org/wiki/Getting_BioPerl).

<sup>b</sup>List::Compare is available from CPAN (<http://www.cpan.org>).

### What is perl?

- Perl is a popular computer programming language.
- You will need to open a terminal and type commands to use these programs.

### blastparser\_csr1.0.pl

Goal: The NCBI standalone BLAST program generates a text file. This script converts BLAST output into a table that can be imported into excel.

Usage: `blastparser_csr1.0.pl -i [BLAST output filename] -n [number of descriptions]`

Output: A text file with the extension “.parsed” will be created.

The output looks like this: seqID, GenBank acc., description, evaluate.

### discard\_shortseqs1.0.pl

Goal: Eliminate sequences below a length threshold.

Usage: `discard_shortseqs1.0.pl -i [input filename] -o [output filename] -c [length cutoff]`

Input: File containing sequences (nucleotide or amino acid) in FASTA format.

Output: File containing sequences longer than the specified length cutoff.

Note: “longer” not “longer than or equal to length cutoff”.

### fasta\_length1.0.pl

Goal: Determine the length of each sequence (number of nucleotides or amino acids).

Usage: `fasta_length1.0.pl [filename]`

Input: File containing sequences (nucleotide or amino acid) in FASTA format.

Output: Two columns 1) SequenceID and 2) sequence length.

### gc\_percent1.0.pl

Goal: GC% is a sequence characteristic used to infer horizontal gene transfer.

Usage: `gc_percent1.0.pl [filename]`

Input: File containing nucleotide sequences in FASTA format.

Output: Two columns 1) SequenceID and 2) GC%  $[(G+C/A+T+G+C)*100]$ .

### **retrieve\_sequence\_subset1.0.pl**

Goal: Obtain desired sequences from a fasta file.

Usage: retrieve\_sequence\_subset1.0.pl -f [fasta filename] -l [list filename]

Input: Two files 1) a fasta file and 2) a list of seqIDs to retrieve (one seqID per line).

Output: The desired sequences should be retrieved from the fasta file.

You should check to confirm that the output contains the expected number of sequences. SeqIDs in the list file that do not exist in the fasta file will be ignored.

### **tag\_fasta\_header1.0.pl**

Goal: It is often desirable to add text to the header line of a fasta file.

(e.g., before combining files for further analysis).

Usage: tag\_fasta\_header1.0.pl -i [input filename] -o [output filename] -t [tag to add]

Input: File containing sequences (nucleotide or amino acid) in FASTA format.

Output: The script will create a file with a tag inserted into the header line of each sequence, like this: >seqID {tag} description

Hint: Use underscores (i.e., “\_”) instead of spaces for multi-word tags.

### **two\_list\_comparison1.0.pl**

Goal: Compare two lists (e.g., accession numbers or annotations).

Usage: two\_list\_comparison1.0.pl -a [list1 filename] -b [list2 filename]

Input: Two lists (one item per line).

Output: Four output files will be created.

Intersection (items that appear at least once in both lists).

Union (items that appear at least once in either list).

Unique to list 1.

Unique to list 2.