Reciprocal Best Hit BLAST Version 2

Nicolas Schmelling

Abstract

This protocol describes the basic steps for a reciprocal best hit BLAST. It will show you how to download sequences from NCBI, creating BLAST databases, and how to blast.

Citation: Nicolas Schmelling Reciprocal Best Hit BLAST. **protocols.io**
dx.doi.org/10.17504/protocols.io.q3rdym6
Published: 18 Jun 2018

Guidelines

I recommend running this protocol in a docker container, so you don't have to worry about installation.

Use this docker container: https://hub.docker.com/r/biodckr/blast/

Protocol

Download sequences from NCBI

Step 1.

The first thing you need to do is to download the sequences of choice from the NCBI server. You need to decide whether you need the nucleotide or protein sequences. I recommend only downloading sequences from genomes that are labeled as *Complete* or *Chromosome* level to ensure a certain quality of the genome assembly.

The following commands will download all protein sequences from genome assemblies labeled as *Complete* or *Chromosome*, futher merge them into a single sequence FASTA file, and move them into the newly created *db* directory.

cmd COMMAND

```
mkdir PROJECT_NAME
cd PROJECT_NAME

wget ftp://ftp.ncbi.nih.gov/genomes/refseq/assembly_summary_refseq.txt

awk -
F '\t' '{if($12=="Complete Genome") print $20}' assembly_summary_refseq.txt > assembly_sum
mary_complete_genomes.txt
awk -
F '\t' '{if($12=="Chromosome") print $20}' assembly_summary_refseq.txt > assembly_summary_
chromosome.txt

mkdir RefSeqCompleteGenomes
mkdir RefSeqCompleteReports
mkdir RefSeqChromosomeGenomes
```

```
mkdir RefSeqChromosomeReports
mkdir db

for next in $(cat assembly_summary_complete_genomes.txt);
do
wget -P RefSeqCompleteGenomes "$next"/*protein.faa.gz;
wget -P RefSeqCompleteReports "$next"/*assembly_report.txt;
done

for next in $(cat assembly_summary_chromosome.txt);
do
wget -P RefSeqChromosomeGenomes "$next"/*protein.faa.gz;
wget -P RefSeqChromosomeReports "$next"/*assembly_report.txt;
done

gunzip RefSeqCompleteGenomes/*.gz
gunzip RefSeqChromosomeGenomes/*.gz

python change_fasta_header.py RefSeqCompleteGenomes
python change_fasta_header.py RefSeqChromosomeGenomes

cat RefSeqCompleteGenomes/*.fasta > db/all_complete_genomes.fasta
cat RefSeqChromosomeGenomes/*.fasta > db/all_chromosomes.fasta

cat db/all_complete_genome.fasta db/all_chromosome.fasta > db/all_genomes.fasta
```

Create database from sequences

Step 2.

After downloading the sequences of choice from NCBI, you need to create a database. Therefore you change the directory into the *db* directory where all the sequences are stored. Next you run the second command. This will create a protein database with all the downloaded sequences. If you downloaded nucleotide sequence make sure you change the *dbtype* option to *nucleotide*.

Parameters

in: FASTA file including all sequences for you database
out: Database name
dbtype: Database type usually 'prot' for proteins or 'nucleotide' for nucleotides

cmd COMMAND

```
cd db
makeblastdb -in FILE_NAME.fasta -dbtype 'TYPE' -out DATABASE_NAME
```

Select sequences as queries to search for homologs in database

Step 3.

After downloading the sequences and creating the database you want to start with your first BLAST run. To do that you need to place your sequences of choice in the directory *seq*. Use a single FASTA file for each

query sequence. The following commands will loop over the query sequences and serach for homologs in your database. The program used here is BLASTP, which is used for protein sequences. Other BLAST programs can be found in the documentation.

Parameters
query: Input sequence
db: BLAST database
out: Name of the output file
outfmt: Output format
evalue: Arbitrary cut-off of sequence similarity. The e-value depends on your database size, so the large your database the smaller your e-value can be. I recommend something between 10^{-5} and 10^{-20}.
word_size: Number of nucleotides/amino acids, which resembles the smallest unit of your query
num_alignments: Maximum number of alignment partner in the database for a single query sequence

cmd COMMAND

```
mkdir seq
# Add sequence FASTA files in seq directory

for f in seq/*.fasta
do
blastp -query "$f" -db DATABASE_NAME -out "${f%.fasta}_blast.xml" -outfmt 5 -
evalue NUMBER -word_size NUMBER -num_alignments NUMBER
done
```

Create new FASTA files for BLAST hits

Step 4.
You now created your first BLAST results. For a reciprocal BLAST you need to align these hits back to the original genome from your query sequence. For this, you need to create FASTA files from your BLAST results. You can find the code to run this analysis on my GitHub code, function *parse_hits* (https://github.com/schmelling/reciprocal_BLAST/blob/master/notebooks/1_KaiABC_BLAST_Data_Collection_and_Perprocessing.ipynb). I'll try to create a universally working script soon.

Align hits against original genome

Step 5.
Now that you have FASTA files for the hits you can start the second BLAST run and align the hits back to the original genome from the query sequence. You need to create a database from the genomic sequences first and then start your BLAST run. This time you set the *num_alignments* parameter to 1, because you only want to record the best alignment partner in the genome. You don't have to worry about the *evalue* since the best hit will most likely be 0 or in close proximity.

cmd COMMAND

```
makeblastdb -in GENOME_FILE.fasta -dbtype 'TYPE' -out DATABASE_NAME
```

```
for f in seq/*_matches.fasta
do
blastp -query "$f" -db DATABASE_NAME -out "${f%_matches.fasta}_back_blast.xml" -outfmt 5 -
word_size NUMBER  -num_alignments 1
done
```

Filter hits and create CSV files for each query sequence

Step 6.

In the last step, you need to filter those hits that aligned back to the original query sequence. These are the only valid hits following the standard procedure for a reciprocal best hit BLAST. You can find the code to run this analysis on my GitHub code, function *filter_hits* (https://github.com/schmelling/reciprocal_BLAST/blob/master/notebooks/1_KaiABC_BLAST_Data_Collection_and_Perprocessing.ipynb). I'll try to create a universally working script soon.