Recipe 11: Running BLAST

You have five basic ways to run BLAST, four of which are as follows: (1) locally from the shell command line, (2) locally from a Python script or interactive Python session, (3) locally using Biopython, and (4) through the NCBI web server using Biopython locally. The fifth way, using your browser and the BLAST web page, will not be described here. The other four methods have the advantage of being automatized easily (e.g., if you want to run BLAST for a whole set of query sequences).

What are the advantages of running BLAST locally? First of all, you can search a query sequence in a customized database, e.g., in a newly sequenced genome you are studying or a set of protein sequences of your interest (e.g., only protein kinases). Second, you may want to insert the program in a pipeline, e.g., to search a large number of query sequences instead of a single one. For example, you might wish for each of them to retain and parse the BLAST output only under certain conditions (e.g., if you have at least a match with >50% sequence identity to the query). Finally, only by running BLAST locally will you have full control over the sequence database and, by that, reproducibility of your search.

To run BLAST locally (either from the shell or from a script), you have to download and install the BLAST+ package (http://blast.ncbi.nlm.nih.gov/Blast.cgi?CMD=Web&PAGE_TYPE=BlastDocs&DOC_TYPE=Download). BLAST+ is a new suite of BLAST tools that utilizes the NCBI C++ Toolkit. You can find instructions on how to download and install the package at http://www.ncbi.nlm.nih.gov/books/NBK1762/.

Once the downloaded files are unpacked and the package is installed, you will have to set up two environment variables in order to tell your system where to look for the installed BLAST programs and inform the

BLAST programs in which directory to search for the databases: PATH and BLASTDB. As for the former, you will have to add the path of the BLAST bin directory to the PATH environment variable of your computer (see Appendix D, Section D.3.5). Otherwise, you have to change to the BLAST bin directory on the terminal shell and run BLAST from there.

Q & A: How Can I Know Which Is the Path of the BLAST bin Directory?

If you install the BLAST program from source, you will have to place the downloaded package under a desired directory, e.g., /home/john. When you unpack the package, a BLAST directory will appear in /home/john (e.g., /home/john/ncbi-blast-2.2.23+). You have to add to the PATH environment variable the bin directory under this BLAST directory. In order to do this, under the `bash` shell you can use the command line:

```
PATH:/home/john/blast-2.2.23+/bin
export PATH
```

Under the `tcsh` shell:

```
setenv PATH ${PATH}:/home/john/ncbi-blast-2.2.23+/bin
```

If you want to permanently modify the PATH variable, you can add this (these) line(s) to the `.bash_profile` or `.cshrc` sturtup files in your home directory, respectively, and restart your computer.
Notice that when you use the dmg disk to install BLAST+ on Mac OS X (10.4 or higher), all BLAST+ programs will be installed under /usr/local/ncbi/blast/bin.

Then you have to modify the BLASTDB environment variable. In order to do this, you have to create the BLAST database directory /blast/db in your home directory:

```
mkdir /home/john/blast/db
```

This is the directory where you will put all the databases (either downloaded from the BLAST website or your custom ones) that you will use with BLAST.

Create a `.ncbirc` text file in your home directory having the following path specification:

```
; Start the section for BLAST configuration
[BLAST]
; Specifies the path where BLAST databases are installed
BLASTDB=/home/john/blast/db
```

The semicolon at the beginning of the first and third lines indicates a comment.

Save and exit the file. Save at least a database in `/home/john/blast/db`. If you want to download a database from NCBI, go to ftp://ftp.ncbi.nlm.nih.gov/blast/db. Now you should be ready to run BLAST.

RUNNING BLAST LOCALLY FROM THE SHELL COMMAND LINE

You have to choose the alignment program from the BLAST package you want to run. Depending on the type of query and target sequences (nucleotide or protein) and type of search (one sequence against several sequences, or pairwise), you will have different programs and/or options from which to choose. You can find a detailed user manual at http://www.ncbi.nlm.nih.gov/books/NBK1763/. Unless you use a preformatted database downloaded from the NCBI ftp site, you will need to format your custom sequence file. To this aim, the BLAST package provides the `makeblastdb` application, which produces BLAST databases from FASTA files:

```
makeblastdb -in mygenome.fasta -parse_seqids -dbtype prot
```

`-in` is the option for the input file, `-parse_seqids` enables parsing of sequence ids, and `-dbtype` specifies the type of input molecules (`nucl` or `prot`). Notice that `–parse_seqids` will be able to parse the sequence identifiers if they start immediately after the ">" and follow the format described in http://www.ncbi.nlm.nih.go v/books/NBK7183/?rendertype=table&id=ch_demo.T5.

The query sequence can be in FASTA format and this is the structure of the command line:

```
blastProgram -query InputSeq.fasta -db Database -out OutFile
```

For example,

```
blastp -query P05480.fasta -db nr -out blast_output
```

`blastp` aligns protein sequences, `nr` is the name of the BLAST-formatted database, `blast_output` is the name you have chosen for the BLAST output file, and `P05480.fasta` is a file that contains your query sequence in FASTA format. There are several other options you can add to this command line (e.g., if you want to set a threshold on the BLAST e-value). You can obtain a list of the available options by simply typing the program name followed by `–h` (usage and description) or `–help` (usage, description, and description of arguments):

```
blastp -help
```

RUNNING BLAST LOCALLY FROM A PYTHON SCRIPT OR INTERACTIVE SESSION

To run BLAST from a Python script, you can translate the previous command to Python code. In this case, you just have to pass the BLAST shell command line as string argument of the `system` method of the `os` module, after having imported it:

```
import os
cmd = "blastp -query P05480.fasta -db nr.00 -out blast_output"
os.system(cmd)
```

In this example, the `nr.00` database downloaded from the NCBI FTP site is used. To work more flexibly with BLAST via the `os.system()` method, you can concatenate the command string using variables for the filenames or for other parameters.

RUNNING BLAST LOCALLY USING BIOPYTHON

You can achieve the same results with Biopython. The only difference is that Biopython provides functions to create suitable command lines. Then you have to use `os.system()` anyway:

```
from Bio.Blast.Applications import NcbiblastpCommandline
import os
comm_line = NcbiblastpCommandline(query = \
    "P05480.fasta", db = "nr.00", out = "Blast.out")
print comm_line
os.system(str(comm_line))
```

Source: Adapted from code published by A.Via/K.Rother under the Python License.

The advantage of this method is that the keyword parameters in `NcbiblastpCommandline` are a little more explicit than in the concise command previously used (this may make it easier to find out if you misspelled an argument). For example, if you want the output in XML format you can define `comm_line` as follows:

```
comm_line = NcbiblastpCommandline(query= \
    "P05480.fasta", db="nr.00", out="Blast.xml", \
    outfmt=5)
```

The actual choice may depend on your personal preference, which is why we explain both versions here.

Notice that, the object to import in the case you want to run BLAST+ with nucleotide sequences is `NcbiblastnCommandline`. For example:

```
cline = NcbiblastnCommandline(query="my_gene.fasta", \
    db="nt", strand="plus", evalue=0.001, \
    out="Blast.xml", outfmt=5)
```

RUNNING BLAST THROUGH THE WEB USING BIOPYTHON

```
from Bio.Blast import NCBIWWW
BlastResult_handle = NCBIWWW.qblast("blastp","nr"," P05480")
BlastOut = open("P05480_blastp.out", "w")
BlastOut.write(BlastResult_handle.read())
BlastOut.close()
BlastResult_handle.close()
```

Source: Adapted from code published by A.Via/K.Rother under the Python License.

When running BLAST through the web, Biopython sends a query to the NCBI server and retrieves the result. The `qblast()` method of the `NCBIWWW` module of `Bio.Blast` runs the BLAST program by taking BLAST parameters, the input sequence, and the target data set as arguments. It returns a "handle" (`BlastResult_handle`) that can be read like a file using the method `read()` and finally written to an output file specifically opened in "w" mode. The output file is in XML format, which can be parsed using the BLAST output parser (see Recipe 9). Please note that the maintainers of the BLAST server expect you to use their service reasonably. If you are planning to run hundreds of queries, please switch to the local version in order to avoid blocking the server (in which case, the server may end up blocking your queries).