

Public Resources for Bioinformatics

- **Databases**
- **Analysis Tools**

Observe: List of databases and service at NCBI, EBI, KEGG, and Ensembl.

Public Resources (II) – Analysis tools

- ❖ **Web-based analysis tools – easy to use, but often with less customization options.**
- ❖ **Stand-alone analysis tools – requires installation and configuration, but provides more customization options.**
- ❖ **Commercial analysis tools**
- ❖ **Scripting for bioinformatics projects**

Stand-alone tools 1.

Rules of the thumb:

- ❖ Make a folder for each program.**
- ❖ Make a sub-folder for input/output if necessary.**
- ❖ Read the instructions before installation.**

Stand-alone tools 3.

Command line applications:

- ❖ **Accounts for a large number of high-quality, sophisticated programs.**

**Practice – (install and) run standalone blast
in your own computer**

Pet Projects:

**Searching for potential ortholog of IL6 in
the *Drosophila* genome**

Practice – Install the blast program (1)

- 1. Read the manuals, download the appropriate BLAST executable file.**
- 2. Run the installation. Inspect the folder following installation.**
- 3. Inspect the contents of the doc and bin subfolders following the installation.**
- 4. Verify the installation by executing the “makeblastdb” in a terminal.**

Practice – Install the blast program (2)

- 5. Make a /blast folder in your home/GMS6014 folder.**
- 6. Add three more folders to your /blast directory, “/query”, “/dbs”, and “/out”.**
- 7. Start a terminal at the blast folder.**

Practice -- BLAST search in your own computer

1. Download data file from the course web page, or Ensemble. Save in the blast\dbs folder.
2. Start a CMD window (or Terminal in Mac), navigate to the C:\GMS6014\blast folder.
3. At the prompt “C:\GMS6014\blast\bin>” type the command “**makeblastdb -in ../dbs/Dm6.44 -dbtype nucl**”* -- format the dataset for the program.
4. Compose the query sequence save as “3LL6.txt” in the “blast\query\” folder.
5. Initiated the search by typing “**tblastn -db ../dbs/ Dm6.44 -query ../query/3IL6.txt -out ../out/myblast.html -html**”

*Command syntax may be different depending on platform and setup

What's in a command?

makeblastdb

Program – format
database for search.

–in

../dbs/Dm6.44

Path to file

–dbtype nucl

Feed me the
input file name

Tell me whether
it is a protein or
nucleotide
sequence file?

For more info, refer to the “user manual” file in the blast\doc folder.

Advantages of Running BLAST at Your Own Machine

- Do it at any time, no waiting on the line.
- Search for multiple sequences at once.
- Search a defined data set.
- Automate Blast analysis.
- Combine Blast with other analysis.
-

BLAST is a program implemented in C/C++

```
void BlastTickProc(Int4 sequence_number, BlastThrInfoPtr thr_info)
{
    if(thr_info->tick_callback &&
        (sequence_number > (thr_info->last_db_seq + thr_info->db_incr))) {
        NlmMutexLockEx(&thr_info->callback_mutex);
        thr_info->last_db_seq += thr_info->db_incr;
        thr_info->tick_callback(sequence_number, thr_info->number_of_pos_hits);
    }
```

Should I care ?

```
return;
```

```
}
```

```
/*
```

Sends out a message every PERIOD (i.e., 60 secs.) for the index.

This function runs as a separate thread and only runs on a threaded platform.

If you care:

1.) Data structure and Algorithm

SEQ {
 char: name
 char: sequence
 int: seq_length

Identify the best alignment
for two sequences (p69-73)

Seq1 : MA-DSV-WC . .

Seq2 : MALD-IHWS . .

Programming language comparison

Translation : C

```
/* TRANSLATION: 3 or 6 frame translate cDNA sequences
*/
//-----
#include "translation.hpp"

int main(int argc, char **argv)
{   int num_seq=0;
    char string[MAXLINE];
    DSEQ * dseq;

    infile.getline (string,MAXLINE);
    if (string[0]!='>') strcpy (dbname,string,MAXLINE);
    while (!infile.eof())
    {   dseq=Get_Lib_Seq ();
        if (dseq->reverse==0)
            Translation (&dseq->name[1], dseq->seq);
        else
            Translation (&dseq->name[1], dseq->r_seq);
        num_seq++;
        if (num_seq%1000==0)
        {   cout<<num_seq<<endl;
            cout<<dseq->name<<endl;
        }
        delete dseq;
    }

    infile.close();
    outfile.close();
    cout<<num_seq<<" translated"<<endl;
    getch();
    return 0;
}

DSEQ* Get_Lib_Seq()
{   int i,n;
    char str[MAXLINE];
    DSEQ* dseq;
    n = 0;
    dseq=new DSEQ;
    strcpy (dseq->name, dbname);
    while (n<MAXLINE)
    {   if (n%1000==0)
        {   cout<<n<<endl;
            cout<<str<<endl;
        }
        dseq->seq[n]=str[n];
        n++;
    }
    return dseq;
}
```

Translation : Python

```
#!/usr/bin/perl -- read from fasta DNA file and translate into
three frames

#

import string
from Bio import Fasta
from Bio.Tools import Translate
from Bio.Alphabet import IUPAC
from Bio.Seq import Seq
ifile = "S:\\Seq\\test.fasta"
parser = Fasta.RecordParser()
file =open (ifile)
iterator = Fasta.Iterator (file, parser)
cur_rec = iterator.next()
cur_seq = Seq (cur_rec.sequence,IUPACUnambiguousDNA())
translator = Translate.unambiguous_dna_by_id[1]
translator.translate (cur_seq)
```

Programming languages and platforms

Efficiency, Power

C/C++

Java -
Biojava

Simplicity, Fast Dev.

Perl - Bioperl

Python -
Biopython