Bioinformatics Introduction

Background and much more material is well presented in Prof Girke's class GEN240B, in particular Sequence Alignments

Running Analysis

- How to run BLAST on command line
- How to setup data files and process
- Development of workflows

Sequence search tools - BLAST

- BLAST is by far the most taught tool in Bioinformatics. I am not going to rehash this completely in this class.
- See NCBI's Introduction to BLAST
- many Millions of pages by Googling "blast introduction tutorial"

BLAST on Biocluster

There are multiple flavors of BLAST (implementations). Focus on the latest version from NCBI.

We will make links to two files which are ORFs from two yeast species. Try this in UNIX

```
# setup some files to do some searches
mkdir BLAST_demo
cd BLAST_demo
ln -s /bigdata/gen220/shared/data/C_glabrata_ORFs.* .
ln -s /bigdata/gen220/shared/data/S_cerevisiae_ORFs.* .
ln -s /bigdata/gen220/shared/data/Yeast_chr2_ORFs.fa .
```

Now we have some files, set them up for running BLAST. Our question is, what ORFs are similar at the DNA level between these two species.

```
module load ncbi-blast # load the module on the biocluster
makeblastdb -dbtype nucl -in C_glabrata_ORFs.fasta
ls
# C_glabrata_ORFs.fasta      C_glabrata_ORFs.fasta.nhr
# C_glabrata_ORFs.fasta.nin  C_glabrata_ORFs.fasta.nsq
# Yeast_chr2_ORFs.fa
head -n 7 Yeast_chr2_ORFs.fa  > YBL001C.cds # get 1st seq for an example

# we do this because I checked and that sequence takes up the first 7 lines
# of the file
blastn -query YBL001C.cds -db C_glabrata_ORFs.fasta
```

BLAST Running

Change the output format to tab delimited with `-outfmt 6` or `-outfmt 7`

```
blastn -query YBL001C.cds -db C_glabrata_ORFs.fasta -evalue 0.001 -outfmt 7 -out YBL001C-vs-Cglabrata.BLA
```

This will query the 1 sequence and produce a tab delimited file.

If you provide a multi-FASTA format file with many sequences, each one will be queried and all the results concated together.

```
blastn -query Yeast_chr2_ORFs.fa -db C_glabrata_ORFs.fasta -evalue 0.001 -outfmt 7 -out yeast_chr2-vs-Cgl
```

BLAST output formats (`-outfmt`)

The default BLAST report is written for people to read. Once you have more than a handful of queries you want a **table** that you can process with UNIX tools (`cut`, `sort`, `awk`), Python, or R. The `-outfmt` option controls this.

The examples in this section were run with BLAST+ 2.17 using the *S. cerevisiae* ORF `YBL001C` as the query against all *Candida glabrata* coding sequences from NCBI. To make the same files yourself:

```
module load ncbi-blast
curl -LO https://github.com/biodataprog/GEN220_data/raw/main/genome/S_cerevisiae.ORFs.fasta.gz
curl -L -o C_glabrata_CDS.fasta.gz https://ftp.ncbi.nlm.nih.gov/genomes/all/GCF/000/002/545/GCF_000002545
gunzip S_cerevisiae.ORFs.fasta.gz C_glabrata_CDS.fasta.gz
makeblastdb -dbtype nucl -in C_glabrata_CDS.fasta
# pull out one sequence: print lines from the YBL001C header until the next header
awk '/^>/ {keep = ($1 == ">YBL001C")} keep' S_cerevisiae.ORFs.fasta > YBL001C.cds
```

First: pick the right `-task`

`blastn` runs `-task megablast` by default. Megablast is very fast but is designed for *nearly identical* sequences (e.g. the same gene from the same species). Between two different species it misses most real matches:

```
blastn -query YBL001C.cds -db C_glabrata_CDS.fasta -evalue 1e-5 -outfmt 7
```

```
# BLASTN 2.17.0+
# Query: YBL001C ECM15 SGDID:S000000097, Chr II from 237467-237153, ...
# Database: C_glabrata_CDS.fasta
# 0 hits found
# BLAST processed 1 queries
```

Use `-task dc-megablast` (discontiguous megablast, for cross-species comparisons) or `-task blastn` (most sensitive, slowest). Running the first 200 yeast ORFs against *C. glabrata*, megablast found hits for **14** queries and dc-megablast for **119**. The rest of this section uses `-task dc-megablast`.

The pairwise report: `-outfmt 0` (the default)

Good for looking at one alignment by eye; hard to use in a program.

```
blastn -task dc-megablast -query YBL001C.cds -db C_glabrata_CDS.fasta -evalue 1e-5
```

```
Sequences producing significant alignments:                          (Bits)  Value

lcl|NC_006026.1_cds_XP_445365.1_587 [locus_tag=CAGL0C04367g] [db_...  190     9e-49

>lcl|NC_006026.1_cds_XP_445365.1_587 [locus_tag=CAGL0C04367g]
...
Length=321

 Score = 190 bits (210),  Expect = 9e-49
 Identities = 225/305 (74%), Gaps = 0/305 (0%)
 Strand=Plus/Plus

Query  1    ATGCCCAAGATCTTTTGTTTAGCGGACGTGTGTATGGTCCCTATTGGCACCGACTCTGCT  60
            ||||| ||| | || ||| |||| || || || ||||| || ||||| || |  |||||
Sbjct  1    ATGCCAAAGGTTTTCTGTCTAGCTGATGTCTGCATGGTTCCAATTGGTACTGGTTCTGCC  60
...
```

Tables: `-outfmt 6` and `-outfmt 7`

`-outfmt 6` writes one line per alignment (called an HSP, *high-scoring segment pair*) with 12 tab-separated columns and **no header**:

```
blastn -task dc-megablast -query YBL001C.cds -db C_glabrata_CDS.fasta -evalue 1e-5 -outfmt 6
```

```
YBL001C lcl|NC_006026.1_cds_XP_445365.1_587 73.770  305 80  0   1   305 1   305 9.29e-49    190
```

`-outfmt 7` is the same table plus `#` comment lines that name the columns and report queries with **0 hits**. That's useful for checking your work; when processing the file skip the comments with `grep -v '^#'`.

```
# BLASTN 2.17.0+
# Query: YBL001C ECM15 SGDID:S000000097, Chr II from 237467-237153, ...
# Database: C_glabrata_CDS.fasta
# Fields: query acc.ver, subject acc.ver, % identity, alignment length, mismatches, gap opens, q. start,
# 1 hits found
YBL001C lcl|NC_006026.1_cds_XP_445365.1_587 73.770  305 80  0   1   305 1   305 9.29e-49    190
# BLAST processed 1 queries
```

The 12 standard columns are worth memorizing - you will use them in homework and projects:

Column	Name	Meaning
1	`qseqid`	query sequence ID
2	`sseqid`	subject (database hit) sequence ID
3	`pident`	percent identity of the aligned region
4	`length`	alignment length
5	`mismatch`	number of mismatches
6	`gapopen`	number of gap openings
7	`qstart`	start of the alignment in the query
8	`qend`	end of the alignment in the query
9	`sstart`	start of the alignment in the subject
10	`send`	end of the alignment in the subject
11	`evalue`	E-value (lower is better)
12	`bitscore`	bit score (higher is better)

Things to know when reading the table:

- **E-value** is the number of hits with a score this good you would expect *by chance* in a database this size. `1e-5` means 0.00001 chance hits. It depends on database size, so the same alignment gets a larger E-value in a bigger database.
- **Bit score** measures the alignment quality independent of database size; use it to compare hits and pick the best one.
- **Percent identity is only over the aligned region.** 100% identity over 30 bases of a 3000 base gene is not a good match. Compare `length` to the query length (see `qlen` and `qcovs` below).
- One query can have **several lines**: multiple database hits, and sometimes several HSPs against the same hit (e.g. two exons, or a repeated domain).
- If `sstart` is larger than `send`, the query matched the **reverse complement** strand of the subject.
- Hits for each query are listed **best first** (by E-value, then bit score).

Choosing your own columns

After `6`, `7`, or `10` you can list exactly the columns you want, in quotes. Useful extras are `qlen` and `slen` (sequence lengths), `qcovs` (percent of the query covered by all HSPs to that subject), and `stitle` (the full description line of the hit):

```
blastn -task dc-megablast -query YBL001C.cds -db C_glabrata_CDS.fasta -evalue 1e-5 \
  -outfmt "6 qseqid sseqid pident length qlen slen qcovs evalue bitscore"
```

```
YBL001C lcl|NC_006026.1_cds_XP_445365.1_587 73.770  305 315 321 97  9.29e-49    190
```

The alignment covers 97% of the 315 base query - a convincing ortholog candidate. The full list of column names is in `blastn -help` under `-outfmt`.

Other formats you may run into:

- `-outfmt 10` - the same as 6 but comma separated (CSV): `YBL001C,lcl|NC_006026.1_cds_XP_445365.1_587,73.770,305`
- `-outfmt 5` - XML, and `-outfmt 15` - JSON; complete but verbose, for reading with a program (Biopython's `Bio.Blast` reads XML)
- `-outfmt 11` - BLAST archive format. Save this if a search is slow; `blast_formatter -archive file.asn -outfmt 6` re-formats it later without re-running the search.

Working with the table on the command line

Here is a table of the first 200 yeast ORFs against *C. glabrata*:

```
# the first 200 sequences: count headers and print until we pass 200
awk '/^>/ {n++} n <= 200' S_cerevisiae.ORFs.fasta > first200.fa
blastn -task dc-megablast -query first200.fa -db C_glabrata_CDS.fasta -evalue 1e-5 \
  -outfmt 6 -out first200-vs-Cglabrata.tsv
head -n 5 first200-vs-Cglabrata.tsv
```

```
YAL001C lcl|NC_005967.2_cds_XP_444805.1_27  83.607  61  10  0   2810    2870    2819    2879    3.16e-10
YAL003W lcl|NC_006029.1_cds_XP_446340.1_1556    82.137  627 103 3   1   621 1   624 3.42e-178   621
YAL004W lcl|NC_006030.1_cds_XP_446529.1_1744    85.781  647 92  0   2   648 671 25  0.0 753
YAL004W lcl|NC_006030.1_cds_XP_446506.1_1721    78.341  651 136 3   2   648 674 25  1.73e-150   530
YAL004W lcl|NC_006034.2_cds_XP_448432.1_3658    74.924  654 153 5   2   648 689 40  4.36e-120   428
```

Some questions we can answer with the tools from the UNIX lectures:

```
# How many queries had at least one hit? (column 1, unique values)
cut -f1 first200-vs-Cglabrata.tsv | sort -u | wc -l
# 119

# Which queries have the most hits? (gene families)
cut -f1 first200-vs-Cglabrata.tsv | sort | uniq -c | sort -nr | head -n 5
#   13 YAL019W
#   10 YAL005C
#    8 YAL004W
#    7 YBL047C
#    7 YAR035W

# Keep only the best (first) hit for each query:
# awk prints a line only the first time it sees the value in column 1
awk '!seen[$1]++' first200-vs-Cglabrata.tsv > first200-besthits.tsv
wc -l < first200-besthits.tsv
# 119

# How many alignments have at least 80% identity? (column 3)
awk '$3 >= 80' first200-vs-Cglabrata.tsv | wc -l
# 26

# The three highest scoring alignments: sort numerically (-n), reversed (-r), on column 12
sort -t$'\t' -k12,12nr first200-vs-Cglabrata.tsv | head -n 3
```

```
YAL005C lcl|NC_006030.1_cds_XP_446529.1_1744    88.336  1929    219 2   1   1929    1   1923    0.0 2461
YAL026C lcl|NC_006030.1_cds_XP_446634.1_1849    75.093  3505    858 9   442 3943    391 3883    0.0 2359
YAL038W lcl|NC_006036.2_cds_XP_449865.1_5112    91.345  1502    126 2   4   1503    7   1506    0.0 2118
```

Notice that `YAL004W` and `YAL005C` have the same best hit (`XP_446529.1`), and that the `YAL004W` hit is on the reverse strand (`sstart` 671 > `send` 25). `YAL005C` is *SSA1*, an Hsp70 chaperone. `YAL004W` is a "dubious ORF" that lies entirely inside *SSA1* on the opposite strand, so it matches the same *C. glabrata* gene in reverse. Always look

at what your sequences are before interpreting hits. Best-hit tables also don't give you orthologs automatically; see the Orthology lecture.

A warning about `-max_target_seqs`: it is commonly used to "keep the top N hits", but it limits the hits kept *during* the search, and it doesn't guarantee you get the N best hits in the database. Filter the table afterwards (as above) instead.

BLAST: what are the tools

- `makeblastdb` - index a database (required to do once before searching)
- `blastn` - DNA/RNA to DNA/RNA search
- `blastp` - protein to protein search
- `blastx` - translated query (DNA/RNA) against protein database
- `tblastn` - protein query against translated (DNA/RNA) database
- `tblastx` - translated query and database (both in DNA/RNA but search in protein space)
- `blastdbcmd` - retrieve a sequence from a blast formatted DB

BLAST: what are the cmdline options?

All the tools have documented command line options. Use -h or -help to get detailed info. Sometimes with no arguments will print documentation, other times will not.

```
makeblastdb
```

```
USAGE
makeblastdb [-h] [-help] [-in input_file] [-input_type type]
-dbtype molecule_type [-title database_title] [-parse_seqids]
[-hash_index] [-mask_data mask_data_files] [-mask_id mask_algo_ids]
[-mask_desc mask_algo_descriptions] [-gi_mask]
[-gi_mask_name gi_based_mask_names] [-out database_name]
[-max_file_sz number_of_bytes] [-logfile File_Name] [-taxid TaxID]
[-taxid_map TaxIDMapFile] [-version]

DESCRIPTION
Application to create BLAST databases, version 2.2.30+

Use '-help' to print detailed descriptions of command line arguments
==========================================================================
```

BLAST: what are the cmdline options?

```
blastn -h
```

```
USAGE
blastn [-h] [-help] [-import_search_strategy filename]
[-export_search_strategy filename] [-task task_name] [-db database_name]
[-dbsize num_letters] [-gilist filename] [-seqidlist filename]
[-negative_gilist filename] [-entrez_query entrez_query]
[-db_soft_mask filtering_algorithm] [-db_hard_mask filtering_algorithm]
[-subject subject_input_file] [-subject_loc range] [-query input_file]
[-out output_file] [-evalue evalue] [-word_size int_value]
[-gapopen open_penalty] [-gapextend extend_penalty]
[-perc_identity float_value] [-qcov_hsp_perc float_value]
[-xdrop_ungap float_value] [-xdrop_gap float_value]
[-xdrop_gap_final float_value] [-searchsp int_value] [-max_hsps int_value]
[-sum_stats bool_value] [-penalty penalty] [-reward reward] [-no_greedy]
[-min_raw_gapped_score int_value] [-template_type type]
[-template_length int_value] [-dust DUST_options]
```

```
[-filtering_db filtering_database]
[-window_masker_taxid window_masker_taxid]
[-window_masker_db window_masker_db] [-soft_masking soft_masking]
[-ungapped] [-culling_limit int_value] [-best_hit_overhang float_value]
[-best_hit_score_edge float_value] [-window_size int_value]
[-off_diagonal_range int_value] [-use_index boolean] [-index_name string]
[-lcase_masking] [-query_loc range] [-strand strand] [-parse_deflines]
[-outfmt format] [-show_gis] [-num_descriptions int_value]
[-num_alignments int_value] [-line_length line_length] [-html]
[-max_target_seqs num_sequences] [-num_threads int_value] [-remote]
[-version]

DESCRIPTION
Nucleotide-Nucleotide BLAST 2.2.30+

Use '-help' to print detailed descriptions of command line arguments
```

BLAST: some key arguments

- -query - query file name (required)
- -db - database file name (required)
- -evalue - set the evalue cutoff
- -max_target_seqs - max number of hit seqs to show
- -num_alignments - max number of alignments to show
- -num_threads - number of threads (parallel processing to run, 8 will be faster than 2)
- -outfmt - specify a simpler format than the text format, try '-outfmt 6' for tabular format
- -subject - instead of doing a DB search, search for alignments between query sequence and 1 to many subject sequences. Useful when want to just see the alignment of 2 sequences already picked out from other analyses

BLAST: Putting it all together

This is a script. e.g. `run_blast.sh`

```
#!/bin/bash -l
#SBATCH -p short -N 1 -n 1 -c 4 --mem 2G --time 1:00:00
#SBATCH -J BLASTN
#SBATCH -o logs/%x.%j.log

module load ncbi-blast

set -euo pipefail

CPU=${SLURM_CPUS_PER_TASK:-1}
if [ ! -f C_glabrata_ORFs.fasta.nhr ]; then
  makeblastdb -in C_glabrata_ORFs.fasta -dbtype nucl
fi
blastn -query Yeast_chr2_ORFs.fa -db C_glabrata_ORFs.fasta \
-evalue 1e-5 -outfmt 6 -out yeastORF-vs-CglabrataORF.BLASTN.tab -num_threads $CPU
```

Now submit this script (the `logs` folder must exist before the job starts)

```
mkdir -p logs
sbatch run_blast.sh
squeue -u $USER # check on your submitted job
```

See UNIX IV for how to choose `-c`, `--mem` and `--time`.

Other types of search tools

- **HMMER**
  - Identify conserved domains in a protein
  - Sensitive searches for distant homologs
  - phmmer can be of comparable speed to BLASTP
  - HMMs are a way to not just match a single sequence but match a pattern
- **FASTA**
  - Another tool like BLAST
  - Doesn't require formatting the database
  - FASTA/SSEARCH are more full length optimal alignments instead of individual scoring pairs, a single best alignment generated
  - Global alignment also with ggsearch

Other seq search tools

- **Exonerate**
  - Another aligner useful for cDNA to genome alignment and protein to genome alignment
  - splice-site aware
  - output harder to parse but there is a GFF-flavor output and parsers in some toolkits
- **USEARCH / VSEARCH**
  - fast, near-exact search tool
  - useful in microbiome short-read
- **DIAMOND**
  - fast, near-exact short read search tool
  - translated BLASTX search option to search proteins against a short read database

Retrieving Sequences from databases

Some of this will be mentioned later in Genome Assembly lecture. But here are some details about different ways to retrieve sequences are located here.

Already downloaded data

`/bigdata/gen220/shared/data/Afum` has some already downloaded datasets.

Remote databases

The International Nucleotide Sequence Database Collaboration (INSDC) are the joint databases for sequence and related biomedical data deposition. These are critical central tools for archive of sequence data for the scientific community. Often when we say "deposited in GenBank" we mean this central repository, but there are data for Sequence Reads, Assemblies, annotated genomes, Gene Expression, and Individual sequence records.

Downloading FASTA databases from NCBI, Uniprot

On HPCC there are already databases installed and indexed for BLAST searches.

```
#!/bin/bash -l
#SBATCH -p epyc -N 1 -n 1 -c 16 --mem 32G --time 2-00:00:00
#SBATCH -J blastp_nr
#SBATCH -o logs/%x.%j.log
module load db-ncbi
module load ncbi-blast
# loads the current ncbi folder as env variables
# $BLASTDB and $NCBI_DB
# after loading this you can run blast without specifying
# the location of the databases
```

```
set -euo pipefail
CPU=${SLURM_CPUS_PER_TASK:-1}
blastp -db nr -query seqs.fasta -out seqs-nr.blastp -num_threads $CPU -evalue 1e-5
```

FTP / Web downloads

NCBI

The NCBI databases include several useful resources. Note these are now giant databases in some places so care is needed in whether you can download these to your own folder and if it already exists on cluster you should try to use those.

- swissprot
- nr - non-redundant protein seqs, very large ...
- nt - non-redundant nucleotide seqs, very large ...
- env_nr - environmental protein seqs
- env_nt - environmental nucl seqs

Another resource that is helpful is Refseq which are *somewhat verified* sequences from genomes.

For example, to get all fungal refseq proteins use the lftp tool. Here is an interactive session:

```
lftp ftp://ftp.ncbi.nih.gov/refseq/release
lftp> cd fungi
lftp> mget fungi.*.protein.faa.gz
lftp> exit
pigz -dc *.faa.gz > refseq_fungi.faa
```

Uniprot

To Download uniprot_swissprot database via ftp protocol. See https://www.uniprot.org/downloads for more download files available including uniref which is a set of clustered sequences at 100%, 90%, and 50% identity which can reduce the size of the total protein database but still leave representatives.

- uniprot_swissprot
- uniref50

The UniRef50 database for this as it isn't too big but useful for some relatively fast searching and more comprehensive than swissprot for taxonomic representation.

Downloading from SRA

The easiest way to download from SRA is using the parallel fastq-dump tool. This is already installed on the cluster `module load parallel-fastq-dump`. It uses the NCBI SRA tool fastq-dump but speeds it up by running parts of the SRA to fastq conversion in parallel.

You can use this for a single SRA accession. Here's an example one https://www.ncbi.nlm.nih.gov/sra/?term=SRR649944.

```
#!/bin/bash -l
#SBATCH -p short -N 1 -n 1 -c 8 --mem 16gb --time 2:00:00
#SBATCH -J fastqdump
#SBATCH -o logs/%x.%j.log
module load parallel-fastq-dump

set -euo pipefail
CPU=${SLURM_CPUS_PER_TASK:-1}
OUTDIR=data/sra
mkdir -p $OUTDIR
SRARUN=SRR649944 # this is a small example accession
```

```
parallel-fastq-dump --tmpdir "${SCRATCH:-/tmp}" --gzip  --sra-id $SRARUN --threads $CPU -O $OUTDIR/$SRARU
```

If you wanted to run this on a file with multiple accessions

```
#!/bin/bash -l
#SBATCH -p epyc -N 1 -n 1 -c 8 --mem 16gb --time 12:00:00
#SBATCH -J fastqdump
#SBATCH -o logs/%x.%j.log
module load parallel-fastq-dump

set -euo pipefail
CPU=${SLURM_CPUS_PER_TASK:-1}
OUTDIR=data/sra
SAMPLEFILE=sra.txt
mkdir -p $OUTDIR
while read SRARUN; do
  parallel-fastq-dump --tmpdir "${SCRATCH:-/tmp}" --gzip  --sra-id $SRARUN --threads $CPU -O $OUTDIR/$SRA
done < $SAMPLEFILE
```

Run `mkdir -p logs` before submitting either script. With many accessions it is faster to run one accession per task of a job array, as in UNIX IV.

Downloading sequence records from GenBank

You can use several tools to download accessions from genbank. It does require certain versions of perl are installed or conda.

```
module load bioperl
bp_download_query_genbank.pl --query 'AY295118.1'
```

```
>AY295118 Parmelia ernstiae voucher MAF 9805 tubulin gene, partial cds.
GAGGACATTCCTCCATAATGTGATACGTAGCTCACAGCTTTCAAGGCTTCAAACAACAAA
TATGTTCCTCGTGCCGTACTCGTCGATCTCGAGCCTGGTACCATGGATGCTGTCCGCGCT
GGTCCTTTTGGCCAGCTTTTCCGACCCGATAACTTCGTATTTGGTCAATCTGGTGCTGGT
AATAATTGGGCTAAGGGTCATTACACCGAGGGTGCAGAATTGGTGGACCAAGTCCTCGAT
GTTGTGCGTCGAGAGGCTGAAGGATGCGACTGCCTCCAGGGCTTCCAGATCACGCACTCC
CTCGGTGGTGGAACTGGTGCTGGTATGGGTACGCTTTTGATCTCGAAAATCCGTGAGGAG
TTCCCAGATCGTATGATGGCTACATTCTCCGTGGTTCCTTCACCAAAGGTATCCGACACT
GTTGTGGAGCCATACAACGCTACTCTCTCCGTGCATCAATTGGTCGAGAACTCGGATGAG
ACCTTCTGTATCGATAATGAGGTTGGTCAAGTGCGATTTTTTCACAGAGGCGCAAGGACT
GATATGTCAATCTAGGCGCTCTATGACATTTGCATGCGCACCCTCAAGCTCTCCAACCCA
TCCTACGGGGATCTTAACCACCTTGTCTCCGCGGTCATGTCTGGTGTTACCACCTGCCTC
CGTTTCCCCGGTCAACTCAATTCCGACCTTCGAAAACTAGCCGTCAACATGGTCCCATTT
CCCCGTCTACATTTCTTCATGGTTGGCTTCGCACCTCTTACCAGCCGAGGTGCTAACTCA
TTCCGTGCGGTCAGCGTACCAGAATTGACCCAACAAATGTACGAC
```

If you want to retrieve a number of sequences at a time you can specify a query. Below are the options for running the tool. If you want to retrieve data from protein database you need to specify the database with `--db` option.

```
bp_download_query_genbank.pl --query "Neurospora[ORGN]" --db nucest -o Ncrassa_ESTs.fa --format fasta
```

```
Other options
 Provide ONE of:

  -q --query query string OR
  --queryfile profile file with query OR
  --gi --gis --gifile file with list of GIs to download
```

```
 Database type:

 -d --db database (nucleotide [default], nucest, protein, )

 -o --out --outfile output file (results are displayed on screen otherwise)
 -f --format sequence file output format (fasta by default)
 -v --verbose debugging output

Query options
 --maxids maximum number of IDs to retrieve in a set (100 at a time by default)
 --reldate
 --maxdate maxdate for a record
 --mindate minimum date for record
 --datetype edat or mdat (entered or modified)
```

Specialized fungal databases

FungiDB

The FungiDB project provides access to a set of Fungal genomes loaded into this system. The resources for downloads are available at this link which includes current and previous releases. Data sets are organized by Abbreviations of genus + species and strain name. For example the Genome, CDS, Protein, and Transcripts associated with the *Neurospora crassa* OR74A strain are available from this link.

JGI

The JGI Mycocosm provides one of the largest collection of fungal genomes through the sequencing and annotation project. There are more than 1000 genomes available through several interfaces hosted by the JGI. Some scripts for automation of downloads are needed to directly extract data from the site onto linux clusters. GLOBUS and other functionality do exist for dataset downloads as well.

Ensembl

Ensembl provides a nearly complete set of public deposited genomes into GenBank organized by major domains. The Ensembl Fungi is a portal with access to thousands of genomes.

Saccharomyces or Candida Genome Database

See SGD and CGD for main site for genome browsers, comparative tools, and access to primary sequence data associated with these fungi.

The FTP site for yeast see ftp://ftp.yeastgenome.org/sequence/S288C_reference for example which has access to the yeast ORFs proteins and coding sequence as well as many other resources like upstream promotor files and other features. Data from multiple *Candida* species is available from http://candidagenome.org/download/sequence/

Local databases

Once you have data files downloaded you can use these.

FASTA Files

HMMER esl-sfetch

I find esl-sfetch one of the better tools for fasta file indexing. Database must be indexed first.

```
module load hmmer
esl-sfetch --index database.fasta
```

```
# fetch record based on single ID passed in on cmdline
esl-sfetch database.fasta accession > accession.fa
# fetch multiple records based on a list of IDs passed in a file (note the -f option)
esl-sfetch -f database.fasta list_of_ids > seqs.fa
# fetch list of IDs passed in on STDIN using a pipe and specifying the input file as '-'
cat ids_to_fetch | esl-sfetch -f database.fasta - > seqs.fa
```

cdbfasta

cdbfasta (Constant database) is useful for indexing fasta and fastq files for retrieval by sequence ID.

```
module load cdbfasta
cdbfasta database.fasta

# to retrieve
echo "A" | cdbyank database.fasta.cidx  > A.fa
cat list_of_ids | cdbyank database.fasta.cidx > retrieved.fa
```

samtools (1.9 and later)

Samtools provides indexing and retrieval of FASTA

```
module load samtools

# index file
samtools faidx DNA_sequences.fasta
```

To retrieve a sequence read after the file is indexed, where accession is the first text after the > in FASTA file, eg `scaffold_1` is the accession in the following:

```
>scaffold_1
TGCATGTCTAAGTATAAGCAATTATACCGTGAAACTGCGAATGGCTCATTAAATCAGTTATCGTTTATTTGATAGTACCTTACTACTTGGATAACCGTGGTAATT
```

To retrieve this sequence from the indexed file use this (specify DNA_sequences.fasta if you used the uncompressed file).

```
module load samtools
samtools faidx DNA_sequences.fasta scaffold_1
```

BLAST indexing

The BLAST indexing to setup a database for sequence alignment and searching also allows retrieval of sequences by identifier.

```
module load ncbi-blast
# index a nucleotide database
# to index a protein database change -dbtype from 'nucl' to 'prot'
makeblastdb -in sequences.fasta -dbtype nucl -parse_seqids

# to retrieve sequences
blastdbcmd -entry ACCESSION -db sequences.fasta -out ACCESSION.fasta

# use this database for sequence searches
# report the output as tab delimited format (outfmt 6)
blastn -query myquery.fasta -db sequences.fasta -out myquery-vs-seqs.BLASTN -outfmt 6 -evalue 1e-5

# Do a protein db search
blastp -query myquery.fasta -db protseqdb.fasta -out myquery-vs-seqs.BLASTP -outfmt 6 -evalue 1e-4

# many other options for BLAST using blastx, tblastn, tblastx and many more options for running BLAST not
```

Databases

Lots of the data are already installed on the system.

```
module load db-ncbi
# will set the variable $BLASTDB so that
blastp -db ref_euk_rep_genomes -query query_file.fasta
```

DIAMOND indexing

DIAMOND is a rapid aligner for protein and translated searches which can operate on short sequence reads as well as assembled genomes.

DIAMOND does not provide a way to extract sequences back out from these indexed databases. Will report a tab delimited output file (m8 stands for the OLD NCBI `-m 8` output which is tab delimited).

```
module load diamond
diamond makedb --in my_protein_db.fasta -d mydb
diamond blastx -d mydb -q reads.fna -o hits.m8
```

Some already built diamond DBs for NCBI databases already installed in `/srv/projects/db/ncbi/diamond/` for example you can take a look.

Short read aligner database indexing

Indexing DNA database for aligning short read DNA sequences against this database (usually a genome).

Indexing for *bwa* in order to setup searches:

```
module load bwa
bwa index database.fa
```

Indexing for *bowtie2*:

```
module load bowtie2
bowtie2-build database.fa database
```

Indexing for *gmap/gsnap*:

```
module load gmap
gmap_build -D genome_index -d genome_name database.fa
```

Indexing for *kallisto* (RNASeq analysis):

```
module load kallisto
kallisto index -i transcripts.idx transcripts.fasta
```

FASTQ Files

cdbfasta

cdbfasta (Constant database) is useful for indexing fasta and fastq files for retrieval by sequence ID.

```
module load cdbfasta
cdbfasta -Q reads.fastq

# retrieve seqs by
echo "ACCESSION" | cdbyank reads.fastq.cidx > fetched_read.fq
cat list_of_ids | cdbyank reads.fastq.cidx > retrieved.fq
```

samtools

Samtools provides indexing and retrieval of FASTQ Files.

If the file is compressed (.gz) it must be compressed with the bgzip tool - which is part of the htslib package. So if the file exists already as a compressed file you need to uncompress and recompress with bgzip.

```
#!/bin/bash -l
#SBATCH -p short -N 1 -n 1 -c 4 --mem 4G --time 1:00:00
#SBATCH -J bgzip
#SBATCH -o logs/%x.%j.log
module load samtools

set -euo pipefail
CPU=${SLURM_CPUS_PER_TASK:-1}
pigz -p $CPU -d READFILE.fq.gz
bgzip --threads $CPU READFILE.fq

# now index
samtools fqidx READFILE.fq.gz

# you can also index an uncompressed file
samtools fqidx READFILE.fq
```

To retrieve a sequence read after the file is indexed, where accession is the first text after the @ in FASTQ file, eg `ERR1309286.4` is the accession in the following:

```
@ERR1309286.4 H4:C3F32ACXX:2:1101:1849:2436/1
CTCTATTTCATCACGTTCGAGAAGATCGCTACGCTTATCGAATTCCAGATTATCATTGTCCGCTTCAACTTCTAGAGAAACTGTGCATGATAATGAGATGC
+
@CCFFFFFGHHGHJJIIJIHJIJJJJIIIIIJIIFJIIJJGIGIEHGIHIIGJIIIJJJJJJIHGF:BDBEEEEEDEA>>CDDCDDDEDDDEDDDDD<>@C
```

To retrieve this sequence from the indexed file use

```
module load samtools
samtools fqidx READFILE.fq.gz ERR1309286.4
```