Running command-line programs, shell scripting

Wildcards:
The Unix shell (and most programming languages) support wildcards, which are placeholder characters that have multiple matches in regular expressions. There are many wildcards and many ways to perfectly specify a pattern, but the most commonly used are *, ? and [ ]

* matches “anything,” meaning one or more characters of any type (this includes nothing)

$ ls
$ ls *
list all files and folders starting with the letter t
$ ls t*
list all files and folders starting with c and ending with a
$ ls c*a
list files and folders starting with c, that have an a following the c anywhere in the name
$ ls c*a*
list all files ending in .fa
$ ls *.fa

? matches exactly one character. To see how it works, let’s create a few emtpy files with boring names.

$ touch t
$ touch tt
$ touch ttt
$ touch filett
$ touch filess
$ touch file99
$ ls ?
$ ls ??
$ ls file?
$ ls file??

the square bracket notation enables a match to any character enclosed in the brackets. Using a dash, you can specify multiple characters easily, e.g. [a-e] or [0-9]

$ ls file[qrst]
list all files and directories whose names contain numbers
$ ls *[0-9]*

Command line programs

(eventual) goal: find variants in CYP2D6, a member of the CYP450 family of enzymes, that is important in metabolism of multiple drugs and may play a role in immunity.

Check out the gene entry for CYP2D6 to get a feel for the gene structure and what variants are known: https://www.ncbi.nlm.nih.gov/gene/1565

You are given a set of primer sequences that are supposed to amplify different regions of the gene in a set of samples, to look for variation.

Wisely, you decide to check the primer sequences.

You'll need to download the BLAST executable: ftp://ftp.ncbi.nlm.nih.gov/blast/executables/blast+/LATEST/
(choose ncbi-blast-2.7.1+-x64-macosx.tar.gz and the accompanying ncbi-blast-2.7.1+-x64-macosx.tar.gz.md5)

Check to make sure that the tarball wasn't corrupted during the download:

```
$ md5 ncbi-blast-2.7.1+-x64-macosx.tar.gz
MD5 (ncbi-blast-2.7.1+-x64-macosx.tar.gz) = 90696a27b9b7d20ea2919893f1377279
```

which matches the ncbi-blast-2.7.1+-x64-macosx.tar.gz.md5 file on the ftp site.

Unpack the BLAST archive and explore a little:

```
$ tar -xzf ncbi-blast-2.7.1+-x64-macosx.tar.gz
$ ls ncbi-blast-2.7.1+
$ ls ncbi-blast-2.7.1+/*
```

The `ncbi-blast-2.7.1+/bin` directory contains all of the programs that we want. We will leave the programs in here for now, but you would in practice want to move them to your /usr/local/bin directory so that you could access them from anywhere without typing the path.

The next task is to align the primers in primers.fa to chromosome 22. (normally you want to align primers to the entire genome, of course, but we'll take this shortcut in the interest of time)

BLAST requires an indexed database for its alignments. The makeblastdb program can do this for us. Remember tab completion:

Run the program by changing directories:

```
$ cd ncbi-blast-2.7.1+/bin
$ ./makeblastdb
```

Run the program using a relative path:

```
$ ./ncbi-blast-2.7.1+/bin/makeblastdb
```

Output is a little confusing:

```
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 2.7.1+

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

Error: Argument "dbtype". Mandatory value is missing:  `String,
`nucl', `prot''
Error:  (CArgException::eNoArg) Argument "dbtype". Mandatory value is
missing:  `String, `nucl', `prot''
```

```
$ ./ncbi-blast-2.7.1+/bin/makeblastdb -help
```

So now make the database:

```
$ ./ncbi-blast-2.7.1+/bin/makeblastdb -in hs_ref_GRCh38_chr22.fa -
dbtype nucl
```

and look in your directory. What did it do?

Use less to look at primers.fa. What type of file is this?
How long is each primer?

```
$ echo TGGAACTACCACATTGCTTTATT | wc
       1       1      24
```

How many primers are there?

```
$ wc primers.fa
      30      30     735 primers.fa
```

But we can be more precise, knowing that the ids are all preceded by “>” in a fasta file. Remember that “>” is also the character used when you want to redirect the output of a command into a file, so be very careful when you try to match this as a pattern. For example,

```
$ grep > myfile
```

will grep the empty string, from no files, and will redirect the output (nothing) into myfile, overwriting the contents of myfile with a blank file. This is rarely what you wanted to accomplish. Using quotes will designate the “>” as a string and it will be properly handled.

```
$ grep ">" primers.fa
>chr22:42126499-42126521
>chr22:42126799-42126821
>chr22:42127099-42127121
```

```
>chr22:42127399-42127421
>chr22:42127699-42127721
>chr22:42127999-42128021
>chr22:42128299-42128321
>chr22:42128599-42128621
>chr22:42128899-42128921
>chr22:42129199-42129221
>chr22:42129499-42129521
>chr22:42129799-42129821
>chr22:42130099-42130121
>chr22:42130399-42130421
>chr22:42130699-42130721
$ grep ">" primers.fa  | wc
      15      15     375
```

Now let's try aligning these to chromosome 22:

```
$ ./ncbi-blast-2.7.1+/bin/blastn
BLAST query/options error: Either a BLAST database or subject
sequence(s) must be specified
Please refer to the BLAST+ user manual.
```

Hmmm. There are a few ways to get usage statements but these are not standardized or required, so you're at the mercy of the developers.

```
$ ./ncbi-blast-2.7.1+/bin/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] [-negative_seqidlist 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] [-max_hsps int_value]
    [-xdrop_ungap float_value] [-xdrop_gap float_value]
    [-xdrop_gap_final float_value] [-searchsp 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 2.7.1+

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

Better. Now let's try some alignments.

```
$ ./ncbi-blast-2.7.1+/bin/blastn -task blastn -db
hs_ref_GRCh38_chr22.fa -query primers.fa
```

Reformat these, as the alignments are hard to parse.

```
$ ./ncbi-blast-2.7.1+/bin/blastn -task blastn -db
hs_ref_GRCh38_chr22.fa -query primers.fa -outfmt 6
```

And then redirect the output to a file, primerblast.txt
Columns are:
query
database sequence (subject)
% identity
length of alignment
number of gaps
number of mismatches
start in query
end in query
start in subject
end in subject
Evalue
Bit score

Exercise:
Use shell commands to calculate the number of times each primer aligns perfectly to chromsome 22.
Tips
use awk to print only the lines that have 100% identity (100.000) and in which the alignment is 23 bases long.
use the cut command to get the query ids from these lines
look at the uniq utility (man uniq) to figure out how to get a count for the number of times each id occurs in the list. The ids are already sorted so you don't have to worry about that.

BLAST output is a little tricky in that the strand is not stated explicitly. Instead, alignments on the minus strand have start positions in the subject sequence that are greater than the end positions.

Exercise:

Write an awk script to print only the minus strand alignments from the original primerblast.txt file. Pipe this to wc to count the alignments.

Now the primers are ordered and you've done the multiplex PCR and sequencing. You get fastq files.

fastq is a format for sequencing data, in which each read is stored along with its quality scores. There are four lines of information for each read: ID, sequence, (variable line), quality scores.

```
@SRR1660308.3117 WTCHG_45084_231:3:1101:1236:66633 length=100
GTTTTGAGATGGAGTCTCACTCTGTTGCCCAGGCTGAAGTGCAGTGGCACGATCTTGGCTCACTGCAACCCCTGCCTCCCGGGTTCAAGCAATTCTCCTG
+SRR1660308.3117 WTCHG_45084_231:3:1101:1236:66633 length=100
CBCFFFFFHHHHHJHHIJJIJJJJIJJIJJJJJJJJJJFHHIJIJJJJIIIIIJJJJJJJJIIJJJIHHHHFFFDCEDDDDD8?BDDDDDDDC@>@>C:
@SRR1660308.3212 WTCHG_45084_231:3:1101:1238:166023 length=100
TTTAAAAGTCCTGGTGAATTGTAAAAAGTGTACGACTTCGGGAGGCCGAGGGGGGGGGGATTACAAGGTTAGGAGTTCGAGACTAGCTTGGCCAGCATGGT
+SRR1660308.3212 WTCHG_45084_231:3:1101:1238:166023 length=100
;?<D=::B+:C:+<22:CF@?C<F44A################################################################################
@SRR1660308.3215 WTCHG_45084_231:3:1101:1238:178965 length=100
TGTAATCCCAGCACTTTGGGAGGCCAAGGCGGGCGGATCACCTGAGGTCAGGAGTTTGAGACCAGCCTGACCAACATCAAGAAACGTCTCTACTAAAAAT
+SRR1660308.3215 WTCHG_45084_231:3:1101:1238:178965 length=100
CCCFFFFFHGHGGJIIJGJGGHGJJJJJJIIIIABB=BDDDDCCCA<3?ACD>@<?CA>@?C?CDDDDBCDCDD<8<CCCA@ACAB2<CDCDDDCDDDC8
@SRR1660308.3238 WTCHG_45084_231:3:1101:1239:110873 length=100
CCCCCGTCTCTACTAAAAATACAAAAATTAGCCAGGCATGATGGCAGGCGCCTGTAGTCCCAGCTGCTCAGGAGGCTGAGGCAGGAGAATTGCTTGAACC
+SRR1660308.3238 WTCHG_45084_231:3:1101:1239:110873 length=100
B@CFDFDDHGGFHJJJIGIGHIEHGIJDGGHGHIIJIGGJJIJJIGHHIGIGGIDHICEHH<EECDFFEDDCEDD@ABBDBD@@DB?A<@>CA>CC8A##
@SRR1660308.3269 WTCHG_45084_231:3:1101:1240:72372 length=100
AGAAGCTTGGAATTTATAAGACTTTCCTTTATAAGATATAGTGGGGGTTTTTTTGGGTGGAGGGGGGTTGTTTTTTGTTTTTTGTTTTCAAGACAGAGTC
+SRR1660308.3269 WTCHG_45084_231:3:1101:1240:72372 length=100
CC@FFFFFHHHHHJJJJJJJJJJJJJJJJJIJIJJIJIJIJFHIJGJBFHIJJHFFDBCDABBDDDD8BB@BC?B@@BDDDDDDDDDDDDDCCCCCCDC@
```

more often the placeholder line just looks like this:

```
@SRR1660308.3269 WTCHG_45084_231:3:1101:1240:72372 length=100
AGAAGCTTGGAATTTATAAGACTTTCCTTTATAAGATATAGTGGGGGTTTTTTTGGGTGGAGGGGGGTTGTTTTTTGTTTTTTGTTTTCAAGACAGAGTC
+
CC@FFFFFHHHHHJJJJJJJJJJJJJJJJJIJIJJIJIJIJFHIJGJBFHIJJHFFDBCDABBDDDD8BB@BC?B@@BDDDDDDDDDDDDDCCCCCCDC@
```

How many sequences are in the fastq file? note that you can't grep for the "@" character as this can occur in the quality string.

Next we'll align the sequences in this fastq file, to chromsome 22.

Download bowtie2 from the sourceforge: https://sourceforge.net/projects/bowtie-bio/files/bowtie2/

Get the current mac archive: bowtie2-2.3.4.1-macos-x86_64.zip
Unzip it and move the executable into the commandline directory.

As with BLAST, we need to make a database.

```
$ ./bowtie2-2.3.4.1-macos-x86_64/bowtie2-build hs_ref_GRCh38_chr22.fa
hg38_chr22
```

this produces a lot of files.
Look at the options for the bowtie2 aligner:

```
$ ./bowtie2-2.3.4.1-macos-x86_64/bowtie2
```

We'll first align a set of unpaired reads that come from a low-pass sequencing experiment for NA12878 that is deposited in SRA (reads were extracted using SRA-BLAST to choose only those reads that align to the CYP2D6 locus).

Exercise
Look through the bowtie2 options to figure out how to align an unpaired fasta file to a reference, and produce a sam file named rawseqs.sam

Exercise
Find two ways in UNIX to print the coordinate of each alignment in rawseqs.sam to chromosome 22.

Python is fantastic for looking at tab-delimited files.

```
$ python
Python 2.7.10rc1 (v2.7.10rc1:80ccce248ba2, May 10 2015, 11:32:08)
[GCC 4.2.1 (Apple Inc. build 5666) (dot 3)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
myfile = file("rawseqs.sam", "r")
myfile
<open file 'rawseqs.sam', mode 'r' at 0x1006a08a0>
for line in myfile:
      print line
```

We can use the split function again to put every field in each line, into a list:

```
for line in myfile:
    pos = line.split("\t")[3]
    print pos
```

what happened??

```
myfile = file("rawseqs.sam", “r")
for line in myfile:
    fieldlist = line.split("\t")
    pos = fieldlist[3]
    print pos

Traceback (most recent call last):
  File "<stdin>", line 2, in <module>
IndexError: list index out of range
```

this means that there are some lines that don't have 4 fields.

For the following exercises, create a file named parsesam.py and type the code into it. The first line of the file should be
`#!/usr/bin/python`

Change the permissions so that your file is executable by the owner. Now you can run the program using either

```
$ python parsesam.py
```

or

```
$ ./parsesam.py
```

Exercise:
Add a conditional statement so that the above for loop works (i.e. it should only try to retrieve the 4th field if there are at least 4 fields).

Exercise:
Modify your program so that it prints the chromosome, position, and length of each aligned sequence.

Exercise:
Modify your program so that it prints the read name and alignment position of every read that aligns between chr22 42141116 and 42150170 (inclusive), but only if the alignment is 51 characters long.

Exercise:
Do any sequences occur more than once in this file?
Tips:
Create a dictionary whose keys are the sequences and the values are the numbers of times that each sequence is found (increment these values as you encounter each sequence). You'll need to check whether a key is in the dictionary, and if it is, you can increment it, and if not, you can add it with the value "1".

The parsesam.py program isn't very useful if is hardcoded to only parse files named rawseqs.sam. Fortunately, there are many ways to modify the behavior of a program with command line flags, and python has multiple methods for handling these.

Create a new text file, named optionexample.py:

```
#!/usr/bin/python

import sys

print sys.argv

for i in sys.argv:
    print i
```

We've done a few new things here. The import command is loading a module, enabling us to access a bunch of functions that are not built in to python. The sys module is critical for communicating with the command line. If we invoke this module, python will create a new variable, sys.argv, which is a list of everything that was typed on the command line. We'll use a for loop to iterate through this list and print out what was passed.

```
python optionexample.py one two three
['optionexample.py', 'one', 'two', 'three']
optionexample.py
one
two
three
```

sys.argv[0] is always the name of the program.

Exercise
Modify your parsesam.py program to use a file name that is passed from the command line. Don't worry about any flags (e.g. "-i" etc); the program should be invoked simply with
python parseblast.py samfile
where samfile is the name of the .sam file to be parsed.

I have included a demonstration program, fancyoptionexample.py, that shows how to parse command line options with flags.

Finally, we'll use samtools to get statistics and convert the alignments into something viewable.

First we need to convert the sam file to a bam file (binary format):

```
$ samtools view -bS rawseqs.sam > rawseqs.bam
```

and then sort the bam file:

```
$ samtools sort rawseqs.bam > rawseqs_sorted.bam
```

and finally index it:

```
$ samtools index rawseqs_sorted.bam
```

Look at some mapping stats:

```
$ samtools flagstat rawseqs_sorted.bam
```