General regex

1. Create regular expressions for the following; this is a theoretical exercise, but you're welcome to try out the regex using **grep**:
    1. Only a number that is a multiple of 5

grep '[05]'

    2. Exactly 5 characters

grep '^.{5}$'

    3. Any letter followed by a number

grep '[A-Z|a-z][0-9]'

    4. The first 3 columns of a BED file (Google UCSC BED format to find out the specifications of the standard BED format)

cut –f 1-3 abcd.bed

    5. The first 3 bases in a DNA sequence

grep '^[AGTC]{3}'

    6. The last 3 bases in a DNA sequence

grep '[AGTC]{3}$'

    7. Two numbers followed by 2 lower case letters

grep '^[0-9]{2}[a-z]{2}'

    8. What does this regular expression match? `\d*\.\d{3}`

0 to any number of digits followed by '.' followed by 3 digits

Regular expression command exercises

1. Searching a file with **grep**
    1. Extract the **knownGene.txt.gz** from the files you downloaded from Canvas. Google the command if you don't know how to extract it.

gzip -d knownGene.txt.gz

    2. Use **grep** to get all genes on chr22

grep chr22 knownGene.txt>newf.txt

    3. Use **grep** to get all and only those genes that occur on chr1

sort -r -u -k1,1 knownGene.txt | uniq | grep chr1$

2. Editing data streams with **sed**
    1. Take the results from 2.2 and duplicate each line

grep chr22 knownGene.txt | sed 'p'

    2. Change the **chr** position of every other line to **cow**

sed 's/chr/cow/g' newf.txt

    3. Delete the lines that have **cow** in them

sed '/cow/d' newf.txt

4. Repeat 1-3, but this time do it "in-place". Read the **man** page to figure out what this means.

   grep chr22 knownGene.txt | sed -i 'p'

   sed -i 's/chr/cow/g' newf.txt

sed -i '/cow/d' newf.txt

Biologically-inspired problem

3. An *in silico* restriction enzyme digestion.[Text Wrapping Break]In a parallel universe, restriction enzymes are called **sed**, and cut microbial genomes on specific patterns. One such enzyme has magically found its way to your computer. Download the **M07149.fasta** from Canvas; we've got some cutting to do!
   1. The restriction enzyme works on the pattern GAATTC and cuts right after the G like this:

      G|AATTC
      CTTAA|G

      Cut the genome into pieces using this restriction enzyme (**sed**)! Store the fragmented genome in a new file. How many pieces did you get? (Don't count this manually – use a command like **wc**).

sed 's/GAATTC/G\nAATTC/g'/home/Downloads/M07149.fasta > new.4.txt

wc new4.txt

(304 lines)

   2. Upon further investigation, you found that the restriction enzyme is a little flexible. It can actually cut after the first base in the following patterns:

      GAATTC, GAATTG,
      GATTTC, GATTTG,
      CAATTC, CAATTG,
      CATTTC, CATTTG

      Update your pattern to cut the genome accordingly. How many pieces did you get this time?

sed
's/GAATTC/G\nAATTC/g;s/GAATTG/G\nAATTG/g;s/CAATTC/C\nAATTC/g;s/CAATTG/G\nAATTG/g;
s/GATTTC/G\nATTTC/g;s/GATTTG/G\nATTTG/g;s/CATTTC/C\nATTTC/g;s/CATTTG/C\nATTTG/g'
/home/Downloads/M07149.fasta > new4_2.txt

wc -l new4_2.txt

(6802 lines)

   3. You underestimated the strength of this enzyme – it can also vary its length. The updated list of patterns has the following letters being optional: third (A or T),

fourth (T) and last (C or G). Update the pattern to get the new number of pieces. How many did you get this time?

sed '' /home/downloads/M07149.fasta > new4_3.txt

wc -l new4_3.txt

(2 lines)

Harder installation problem

Continuing our installation discussion from last week, this week we will install MySQL without using root. MySQL is a relational database management system. If that doesn't mean anything to you right now that's okay, but databases are extremely useful in bioinformatics. I recommend relational databases (taught in CS 4400) for everybody. MySQL is also a good example for typical compilation/installation.

1. Download the latest source code for MySQL (http://dev.mysql.com/downloads/mysql/), not the precompiled binaries.

wget https://dev.mysql.com/get/Downloads/MySQL-8.0/mysql-8.0.30.tar.gz

2. Next step requires **cmake**. What is **cmake**?

Cmake is used to create test and package software,which is a bundle of open source Cross-platform tools and it does this by using a simple platform and compiler independent method.

3. Unpack the source and run **cmake** . in the directory you just created. If you don't have **cmake** in your system, get it using **apt-get**. Don't attempt **cmake** install without root, it's a harder install.

Sudo apt-get install cmake

4. Build the MySQL executables with **make**

Make

5. Try to install them with **make install**

make install

6. That should have failed. Why?

It failed because there was no sudo priveleges.

7. How would you get around this with **sudo**? How would you get around this with **cmake**? (Hint: you have to tell **cmake** where YOUR bin directory is. Run **cmake --help**)

Sudo make install