BIOL*3300 Lab5 F21

Advanced Command Lines

To be sure we are still working in the same place, please login to your Compute Canada Graham account and run the following commands:

```
cd scratch/Biol3300
mkdir Lab5
cd Lab5
pwd
```

The sed command

The unix command sed stands for Stream Editor. It can do many text manipulations. We will use it to extract or to replace text. A substitution happens on a line by line basis. It effectively allows us to do a fast search and replace on our data.

The general format is "s/<pattern>/<replacement>/<options>"

Here we use echo to produce input that is piped to sed.

```
echo 'hello' | sed 's/lo/p!/'
```

Make a file pets.txt. One way of doing this is to use echo and redirect/append its output to a file. Another way is with nano pets.txt
Your file should have the following lines:

```
I really love cats.
I think cats are great.
```

We can use cat filename to pipe data to sed lines from a file:

```
cat pets.txt | sed 's/cats/dogs/'
```

We can also give sed to the file directly. It does not open the entire file into memory, so it is

fast. What information is in pets_dog.txt?

```
sed 's/cats/dogs/' pets.txt
sed 's/cats/dogs/' pets.txt > pets_dog.txt
```

sed will use character classes for matches, what does [a-z] match?

```
echo 'hello' | sed 's/[a-z]/j/'
echo 'hello how are you' | sed 's/^hello/greetings/'
```

Here are some new characters and interpretations. A dot means "any character":

```
echo 'hello' | sed 's/h.llo/goodbye/'
```

The carat next to a character class's bracket's contents means "NOT":

```
echo 'hello' | sed 's/he[^p-z]lo/goodbye/'
echo 'hello' | sed 's/he[^l]lo/goodbye/'
```

Do you see the logic of these? It can get tricky. Try some more.

If you want to match a character such as **[** or **.** or **$** which have sepcial meanings, you need to escape it with a backslash:

```
echo 'How much $ was it' | sed 's/$/money/'
echo 'How much $ was it' | sed 's/\$/money/'
```

We can expand the flexibility of our patterns using "extended regular expressions" -E. There are some differences amongst machines as to what is included in basic patterns.

```
echo 'hello' | sed -E 's/[a-z]+/goodbye/'
echo 'Hello' | sed -E 's/[a-z]+/goodbye/'
```

+ means the preceding item will be matched "one or more" times
***** means the preceding item will be matched 0 or more times.
? means at most once.

Global and case insensitive matches

A trailing 'g' replaces all instances of the pattern on a line. If we omit it then 's///' will only replace the first instance of the patternon each line. A trailing 'i' makes the search case insensitive.

```
    echo 'hello' | sed 's/l/g/g'
    echo 'Hello' | sed 's/h/j/i'
```

A common thing that you may want to match is the tab. It is \t.

```
    echo 'hello how are you' | sed –E 's/ /\t/g'
```

Pattern matching and replacement questions in the superheroes.txt file

Note you can either copy it from your Lab3 folder or create it again with nano.

```
Batman  Bruce Wayne Hero    DC
Invisible Woman Susan Storm Richards    Hero    Marvel
Supergirl       Linda Danvers   Hero    DC
Superman        Clark Kent      Hero    DC
Wonder Woman    Diana Prince    Hero    DC
Iron Man    Tony Stark  Hero    Marvel
Captain America   Steve Rogers    Hero    Marvel
```

Replace all cases of "Super" with "Soso"

```
    sed 's/Super/Soso/' superheroes.txt
```

Replace all cases of "woman" with "man"

```
    sed 's/[wW]oman/man/' superheroes.txt
```

Replace "Susan Storm Richards" with "Susan Storm"

```
    sed 's/Susan Storm Richards/Susan Storm/' superheroes.txt
```

Using part or all of a pattern match as replacement text

You can save the part of a string that matches a pattern by enclosing the pattern in parentheses ().
You can then recover the matching string with \1, \2, etc...
Replace "Susan Storm Richards" with "Susan Richards"

```
sed –E 's/(Susan) Storm (Richards)/\1 \2/' superheroes.txt
```

Replace all cases of "man" with "woman"

```
sed –E 's/[wW]?[o]?[mM]an/woman/g' superheroes.txt
```

The awk command

awk is great for processing the records of tabular data. A field is a column's data. awk separates the fields. Field one's values are in $1, etc..

awk uses pattern-action pairs, but you can just match a pattern or just perform an action. The 'pattern' is a lot like an if statement. It evaluates something as true or false. If true, something is done.
If we omit the 'action' but specify the pattern, awk will print all records that match the pattern. If we omit the pattern, awk will run the action on all records.

First, we'll copy the gff3file from the Lab2 folder.
Here, have an action, print, and no pattern.
Print is a mimic of 'cat'. $0 means "the whole record", although we pipe to head to get only a few lines.

```
cp ~/scratch/Biol3300/Lab2/gff3file .
awk '{print $0}' gff3file | head
```

We can specify certain columns to print:

```
awk '{print $4 "\t" $5}' gff3file | head
```

The command is a mimic of:

```
cut –f4,5 gff3file | head
```

We can also do arithmetic with awk:

```
awk '{print $5 – $4 + 1}' gff3file | head
```

What do these numbers represent?

Second, we can use patterns and no actions.
Here, awk patterns evaluate whether an expression is true or not. If true, the record is printed by default.

```
awk '$3 == "exon"' gff3file | head
awk '$4 == 52843' gff3file
awk '$0 ~ /ENSMODT00000083036/' gff3file
```

Logical operators enable "yes/no" answers:
a == b a equals b.
a != b a is not the same as b.
a < b a is less than b.
a >= b a is more than or equal to b.
a ~ /b/ a matches a regular expression b
a !~/b/ a does not match a regular expression b

How many genes have a chromosomal position 270,000,000 or higher? (Column 4 has the feature location; column 3 has the feature name.)

```
awk  '$4 > 27000000' gff3file | cut –f3 | grep ^gene | wc
```

How many genes are there?

```
awk '$3 ~ /^gene/' gff3file | wc
```

These two commands are same:

```
awk '/gene/' gff3file | wc
grep gene gff3file | wc
```

Comparisons can be combined:
&& means "and"
|| means "or"

```
awk '$4 > 27000000 && $3 ~ /^gene/' gff3file | wc
```

We can apply arithmetic to the fields and evaluate the outcome:

```
awk '($5 - $4) > 1000000 && $3 ~ /^gene/' gff3file
```

Finally, we can work with pattern-action pairs. Prints all feature names that begin at 190,022,245.

```
awk '$4==190022245 {print $3}' gff3file
```

Prints all feature names that are gene or mRNA:

```
awk '$3 ~ /^gene/ || $3 ~ /mRNA/ {print $3}' gff3file | head
awk '$9 ~ /ENSMODG00000028894/ && $3 == "gene" {print $5-$4+1}' gff3file
```

References

These lab materials are from the following tutorials:

1. https://astrobiomike.github.io/unix/getting-started
2. https://ucdavis-bioinformatics-training.github.io/2020-Intro_Single_Cell_RNA_Seq/prerequisites/cli/advanced-command-line