Tutorial 04 - Shell scripts and finding stuff

the many uses of $ in Unix

- $ is used with a previously defined shell variable, but NOT when defining the variable

  ```
  variable=2

  echo $variable
  ```

- $() can be used to execute a series of Unix commands; this allows us to store the results of the commands in a variable or as an argument to another function call

  ```
  variable=$(cat input.txt | wc -l)

  wc -l $(find . -name '*.txt')
  ```

- variable=$(()) can be used to execute math with integers and store in a variable

  ```
  variable=$(2 + 2)
  ```

why write a shell script?

- repeat tasks easily
- reproducible and documented (comments!!!!)
- modularize your analysis processes

grep vs. find

- grep works inside files
- find works outside files

grep

```
cat  file.txt | grep -flags pattern
```

lots of useful flags with grep:

-c counts number of matches

-e specify a regular expression (more next week)

-i ignore upper vs. lower case

-m stop searching after a specified number of matches

-n return line numbers with matched lines

-v return lines that do not match

-w match complete words only

`find`

- powerful search function
- searches for files
- allows you to do complex and recursive `ls`
  - complex: you can specify files or directories and search by name of files
  - recursive: across multiple directories
- can embed a find call in $() and pass it as an argument to other Unix functions

  e.g. `wc -l $(find . -name '*.txt')`

passing information into a script #1

A limited number of arguments that are heterogeneous (e.g. file name and arguments to Unix functions)

```
# usage: bash script.sh input.csv , 1,2

cat $1 | cut -d $2 -f $3 | sort
```

passing information into a script #2

A set of file names to be used in a for loop

```
# usage: bash script.sh *.csv
for file in "$@"
do
    cat $file | head -n 5 | sort -u
done
```

passing information into a script #3

You want your script to be orthoganol and take information from stdin and return it to stdout

```
# usage: cat input.csv | bash script.sh
```

```
# usage: cat *.csv | sort | tail -n 100 | bash
script.sh
```

```
cut -d , -f 1,2 | sort
```

passing information into a script #4a

You want to loop through a set of files, but also want to pass additional arguments for use with Unix functions in the script

- using ' ' to delay evaluation of wildcards

```
# usage: bash script.sh '*.csv' 5

for file in $1

do

    cat $file | head -n $2 | sort -u

done
```

passing information into a script #4b

You want to loop through a set of files, but also want to pass additional arguments for use with Unix functions in the script

- using $(cat) to put the set of file names from stdin into a script variable

```
# usage: find . -name "*.csv" | bash script.sh 5

list=$(cat)

for file in $list

do

    cat $file | head -n $1 | sort -u

done
```

challenge

You have been provided a zip file with short DNA sequences from a number of samples from four different sites. These DNA files are distributed in four separate directories (one per site). Each file (from an individual sampled at a site) has some number of lines, with each line representing a different random DNA sequence recovered from an individual at a site. We want to accomplish three different tasks with these DNA sequences and will write a shell script for each task.

challenge

1. Write a shell script that can count the number of sequences that include the start of a gene (ATG) for each individual (file) at a single site. Make this script flexible so that it could work on any site regardless of how many individuals were sampled. Also, you can't count on .txt as always being the file extension as different sequencing centers use different file extensions, so make sure your script works with any file extension.
2. Write a shell script that can count the total number of sequences that include the start of a gene (ATG) for all individuals (files) combined at a single site.

challenge

3. Write a shell script so it will work to count the number of sequences that include the start of a gene (ATG) for each individual (file) at all sites in one run of the script. However, the company that did the DNA sequencing for you sent a note that any sequences beyond the first five from an individual (file) are not reliable and should be ignored. Make the script flexible so anyone could reuse it regardless of how many sites or individuals they have. Also, the sequencing company said future batches might have similar issues with only the first N sequences being good, but N won't necessarily be five each time. Please make your script flexible for this issue in the future.

function review

`grep`

`find`

`$( )`