2. Introduction to SED

2.1. File content and odd line retrieval

1.Type sed on the command line and look at the options provided. What option is required to obtain the sed version number?

2. Inspect the file called exercises.fasta using less. How many sequences with fasta headers are in the exercises.fa file?

grep -c ">" exercises.fasta = 1000 sequences

3. Examine the sequences.fasta file – what do you notice

a. Are all the sequence characters the same case? No

b. The sequence characters are in a mixture of cases? Yes

4. From examining the sequences.fasta file, we know that each fasta header and sequence sit on a new line, with the fasta headers being on odd numbered lines. Let us try and print out the fasta header lines only using sed:

sed -n 'p;n' sequences.fasta

5.What command would you use to print out only the even number lines with the actual sequence?

sed -n 'n;p' sequences.fasta

2.2. Pattern modification and global and piping

6. Change the character cases to make them uppercase using sed

sed 's/[a-z]/\U&/' sequences.fasta

7. Not all the characters were changed to upper case – what would I need to add to the sed command to change everything to upper case?

sed 's/[a-z]/\U&/g' sequences.fasta

8. Look at the gene.gff file copied to the sed_practical directory

a. How many entries / lines are there?

wc -l genes.gff - 10

9. Convert the pattern chrm to Chromosome in the genes.gff file

sed 's/chr/Chromosome/g' genes.gff (try with global option and without – explain the difference)

10. Convert the gene.gff file into a comma separated file:

sed 's/\t/,/' genes.gff

11. The command above only worked for the first tab character (if you didn't use the global flag). Use the global flag (g):

sed 's/\t/,/g' genes.gff

12. From the sequences.fasta file, add the organism p_falciprium to the start of the fasta headers:

sed 's/^>/>p_falciprium_/' sequences.fasta

13. For the command above,don't use the global flag. Are all the fasta sequence headers modified, if so why?

Sed reads in a file line by line and makes a change to the first instance of the character on the new line – each fasta header starts on a new line

14.Change all the characters in the sequences.fasta file to upper case and add p_falciprium to the start of the fasta headers and then convert the uppercase fasta headers to lowercase using a series of sed pipes:

sed 's/[a-z]/\u&/g' sequences.fasta | sed 's/^>/>p_falciprium_/g' | sed 's/SEQUENCE/sequence_/g'

15. What would you need to do to the above command to send the output into a new file?

sed 's/[a-z]/\u&/g' sequences.fasta | sed 's/^>/>p_falciprium_/g' | sed 's/SEQUENCE/sequence_/g' > formatted_seqences.fasta

16. Using the genes.gff file, print out only lines 1, 3 to 5, 7 and 9:

sed -n '1p; 3,5p; 7p; 9p' genes.gff

2.3. Reformating BED files

17. Use the exercises.bed for this next set of exercises. Inspect the bed file:

less exercises.bed

18. How many lines / rows are in the exercises.bed file?

wc -l exercises.bed = 326 lines

19. For questions 19 to 22, pipe the output to less. Using sed, substitute the all the terms contig- with contig_ in the exercises.bed file using the global flag

sed 's/contig-/contig_/g' exercises.bed | less

20. Using sed, substitute gene- with Gene_ in the exercises.bed file using the global flag

sed 's/gene-/Gene_/g' exercises.bed | less

21.Using sed, substitute the term repeat with REPEAT in the exercises.bed file using the global flag

sed 's/repeat/REPEAT/g' exercises.bed | less

22. Repeat questions 21, 22, and 23, but without using the global flag substitution flag.

sed 's/contig-/contig_/' exercises.bed | less
sed 's/gene-/Gene_/' exercises.bed | less
sed 's/repeat/REPEAT/' exercises.bed | less

23. Is there a difference between using the global substitution flag Yes / No?

No

24. What would the explanation be?

Sed reads in a file pattern by pattern and makes the first substitution of that pattern in the line, these patterns only appear once in each line

25. Combine sed pipes to substitute contig- with contig_ gene- with gene_ and repeat with REPEAT and pipe the output to less to check if the substitutions are working

sed 's/contig-/contig_/g' exercises.bed | sed 's/gene-/gene_/g' | sed 's/repeat/REPEAT/g'

26. Once you are happy that your sed command is working, send the output to a file called formatted_exercises.bed

sed 's/contig-/contig_/g' exercises.bed | sed 's/gene-/gene_/g' | 's/repeat/REPEAT/' > formatted_exercises.bed

3. Introduction to AWK

3.1. Visualisation, filters and calculations

1. Using awk, print out the first column of the genes.gff file:

awk '{print $1}' genes.gff

2. Print out column 9 of the genes.gff file using awk:

awk '{print $9}' genes.gff

4. Change the awk command slightly to take into account a default delimiter, a \t in this case

awk -F "\t" '{print $9}' genes.gff

5. How many columns are in the dataset genes.gff, use the awk NF function

awk '{print NF}' genes.gff

6.Do you get the differences between 8 to 10 columns which is not correct as a general feature format file should have 9 columns that should be split by tabs (see the url below for an explanation of a gff file: https://www.ensembl.org/info/website/upload/gff.html Try again to the get the correct number of fields by splitting on the correct delimiter which is tab-seperated. Just in case you never used "\t" for the previous question.

awk -F "\t" '{print NF}' genes.gff

7. Find out how many unique chromosomes are contained in our gene.gff file using awk and sort:

awk -F"\t" '{print $1}' genes.gff | sort -u

8. Extract columns 1, 3, 6 and 9 from the genes.gff file while keeping the formatting

awk -F "\t" '{print $1, $3, $6, $9}' genes.gff

9. Use awk's BEGIN and OFS functions to get the output in tab delimited format of columns 1,3,6 and 9:

```
awk -F "\t" 'BEGIN {OFS="\t"} {print $1,$3,$6,$9}' genes.gff
```

10.Extract all genes that map to chromosome 1 within the genes.gff file and redirect it to a file named chromosome_1_genes.gff

```
awk -F"\t" '$1=="chr1" {print $0}' genes.gff > chromosome_1_genes.gff
```

11. Filter the genes.gff file to get all entries with chromosome 1 and annotations as genes using the && operator:

```
awk -F"\t" '$1=="chr1" && $3=="gene"' genes.gff
```

12. Print out a specific column using the filtering criteria above e.g. column 9

```
awk -F"\t" '$1=="chr1" && $3=="gene" {print $9}' genes.gff
```

13. Pull out all the rows where the column is equal to chromosome 1, or the column 3 is equal to a gene using the || operator

```
awk -F"\t" '$1=="chr1" || $3=="gene"' genes.gff
```

14. modify the previous awk construct to also filter on numerical values

```
awk -F"\t" '$1=="chr1" && $3=="gene" && $4 < 1100' genes.gff
```

15.change the values in the source field to H_sapiens and print out a new gff file using awk

```
awk -F"\t" 'BEGIN {OFS="\t"} {$2="H_sapiens"; print $0}' genes.gff
```

16. Using awk, get the length of repeats from genes.gff file keeping in mind the offset +1

```
awk -F"\t" '$3=="repeat" {print $5 - $4 + 1}' genes.gff
```

17. get the total length of repeats from the genes.gff file

```
awk -F"\t" 'BEGIN{sum=0} $3=="repeat" {sum =+ $5 - $4 + 1} END{print sum}' genes.gff
```

18. calculate the mean of the scores in the gene.gff file using a count as well

```
awk -F"\t" 'BEGIN{sum=0; count=0} $3=="gene" {sum =+ $6; count++} END{print sum/count}' genes.gff
```

19. Extract all genes that map to chromosome 2 within the genes.gff file

```
awk -F"\t" '$1=="chr2" {print $0}' genes.gff
```

20. Print out columns 1, 2, 4 and 8 using awk and insert a tab delimiter

```
awk -F "\t" 'BEGIN {OFS="\t"} {print $1,$2,$4,$8}' genes.gff
```

21. Sum up the total length of genes in the genes.gff file

```
awk -F"\t" 'BEGIN{sum=0} $3=="gene" {sum =+ $5 - $4 + 1} END{print sum}' genes.gff (
```

22. Calculate the mean gene length in the genes.gff file using awk

```
awk -F"\t" 'BEGIN{sum=0; count=0} $3=="gene" {sum =+ $5 -$4 +1; count++} END{print sum/count}' genes.gff (55.6667)
```

23. Find all the entries labelled as gene in column 3 and have a score greater than 0.55 in column

```
awk -F"\t" '$3=="gene" && $6 > 0.55' genes.gff
```