Coding task results for 31.05.2021

Basic shell tools for viewing and modifying data

Head, tail and less

1. Check head and tail on the sam-file. Is there anything unusual?

Input: `head coding_task_sample.sam / tail coding_task_sample.sam`

Output:
```
--SRR9077711.2287  0   chr5    149550844 […]
```

→ Head and tail can be used to have a first look at the data

→ Some common problems regarding format can be found and also blank lines

Head, tail and less

2. How many lines are printed and how can the amount of lines be changed?

Input: `head -n 5 coding_task_sample.sam`

Output: `-n x` lines of the file

→ Some formats can have a long header

→ Then more then 10 lines can be useful to see the full structure

Grep

1. To which chromosome has the read SRR9077711.3518 mapped?

Input: `grep SRR9077711.3518 coding_task_sample.sam`

Output:
```
SRR9077711.3518    0    chr10    117285431    […]
```

→ Grep is best used if you know what you are looking for (e.g. mapping of a speficifc read or sequence)

Grep

2. Can grep print more then one line after the found string? If yes how?

Input:

```
grep -A1 SRR9077711.3518 coding_task_sample.sam
```

Output:

```
SRR9077711.3518    2048    chr10   117290184   […]
SRR9077711.3519    16   GL000255.2 4406835     […]
```

→ Also file formats with more then one line per entry can be filtered or searched via grep

Grep

3. How much mappings have an edit-distance of 0?

Input: `grep NM:i:0 coding_task_sample.sam | wc -l`

Output: `2277`

→ Easy commands like this can be useful for pre-filtering data
(e.g. filter out multimappings or too many mismatches)

Grep

4. Find all mappings that contain the sequence "AGACACCG"

Input: `grep AGACACCG coding_task_sample.sam`

Output:
```
SRR9077711.3137    16  chr16   547748  […]
```

→ Only one mapping contains this sequence

→ Even sequence motifs can be directly searched on the shell

AWK

1. How many records are in the sample-file?

Input: `awk 'END {print NR}' coding_task_sample.sam`

Output: `3012`

→ On the command line you can also use: `wc -l <file_name>`

AWK
2. Print the first line of the sample file

Input: `awk 'NR == 1 {print $0}' coding_task_sample.sam`

Output: `--SRR9077711.2287 0 chr5 149550844 […]`

→ On the command line you can also use: `head -n 1 <file_name>`

AWK
3. Print every second line

Input: `awk 'NR%2 == 1' coding_task_sample.sam`

→ Most calculation operations can also be used in awk

AWK

4. Get the bitwise flag of "SRR9077711.3997"

Input: `awk '$1 == "SRR9077711.3997" {print $2}' coding_task_sample.sam`

Output: `16`

→ `$1` to `$n` contain the different columns of a line

→ Logical operators can be used in awk

AWK

5. What is the mean length of all segment sequences?

Input:

```
awk '{sum += length($10)}
END {print sum/NR}' coding_task_sample.sam
```

Output:

```
71.2958
```

→ It is possible to set own variables in an awk command

→ Variables can be set on the fly

→ There are basic functions in awk like length()

AWK

6. What is the maximum length of all segment sequences?

Input:

```
awk 'BEGIN {max = 0} {if (length($10) > max) max = length($10)}
END {print max}' coding_task_sample.sam
```

Output:

```
76
```

→ If, for and while are also available

→ Combination of logical and calculation operations can be used for extensive numerical filtering

Sed

1. Remove all mappings to non conventional chromosomes

Input: `sed -i '/\tchr/!d' coding_task_sample.sam`

→ For the direct alteration of files sed can be used

→ `-i` in the command overwrites existing file (use with care)

→ `/d` at the end deletes all which matches the string

Sed
2. Globally change SRR9077711 to SRR9077715

Input: `sed 's/SRR9077711/SRR9077715/g' coding_task_sample.sam`

Output:
```
--SRR9077715.5012  16  chr4    71023608    […]
```

→ Replacing certain strings is the most common usage

Sed

3. Replace all appearances of "--" with nothing

Input: `sed 's/--//g' coding_task_sample.sam`

Output:

```
SRR9077711.5012    0    chr12    21643935    […]
```

→ Strings can also be replaced with nothing, removing it

Remove all blank lines with awk or sed and write the result in a new file

Input:

```
sed '/^$/d' coding_task_sample.sam > filtered_sample.sam
awk NF coding_task_sample.sam > filtered_sample.sam
```

→ There are always different ways to achieve the same thing

→ If unsure: a search engine of one's choice is just a click away

→ For sed: `^` is the start of a line, and `$` the end of a line,
if nothing is between them, the line is deleted

→ For awk: `NF` contains number of fields in a line,
if non-zero the line is printed (default awk action)

Conclusion

→ Common toolbox for persons which are working with data

→ Shell commands are a easy and fast way to search
and filter files on unix systems

→ Knowing the right ones (or searching them) can safe a lot of time

→ Some analysis can be directly done via the shell

→ Shell commands can also be used in R or python scripts