CSC BioWeek 2016: Using Taito cluster for high throughput data analysis

3. 2. 2016

Working with files in command line

A note on typography:

Some command lines are too long to fit a line in printed form. These are indicated by a backslash "\" at the end of line. It should not be included when typing in the command.

For example

```
example command \
continues \
and continues
```

Should be typed in as:

```
example command continues and continues
```

Exercise 1: Working with data columns

Login to **taito-shell.csc.fi**

```
cd $WRKDIR
mkdir ex1
cd ex1
```

Copy exercise data to your own directory:

```
cp /wrk/trng52/0302/ex1.zip .
```

Open zip package:

```
unzip ex1.zip
```

The files include a result file from a BLAST search (`pb_blast_results`)

You can try re-runnig the BLAST search if you wish, using these commands:

```
module load biokit
```

```
pb blastn -query R.fasta -dbnuc NC_009656.fna -out pb_blast_results -outfmt 7
```

By running command:

```
head -50 pb_blast_results
```

You can see that the result file contains columns, where the query sequences are in the first column and the hit sequences in the second column. You can now check, how many hits each query sequence got with command:

```
awk '{print $1}' pb_blast_results | grep -v "#" | sort | uniq -c | sort -k1n
```

Study how this command works by executing it step by step:

```
awk '{print $1}' pb_blast_results
awk '{print $1}' pb_blast_results | grep -v "#"
awk '{print $1}' pb_blast_results | grep -v "#" | sort
awk '{print $1}' pb_blast_results | grep -v "#" | sort | uniq -c
awk '{print $1}' pb_blast_results | grep -v "#" | sort | uniq -c | sort -k1n
```

You can check the number of query sequences in the input file with command:

```
grep -c ">" R.fasta
```

Alternative ways to count number of sequences with EMBOSS:

```
module load emboss
seqcount R.fasta
infoseq_summary R.fasta
```

If you compare that to the number query sequences in the result file:

```
awk '{print $1}' pb_blast_results | grep -v "#" | sort | uniq -c | wc -l
```

You can notice that the command used above shows no information for those query sequences that did not get any matches.

Excercise 2:

Often it is necessary to change files slightly to use them in different analysis programs. This exercise simulates some typical changes you need to do.

Copy a file “hsa.gff3” to your own directory:

```
cp /wrk/trng52/0302/hsa.gff3 .
```

Now do the following changes to the data:

1. Remove comment lines (lines starting with #)
2. Remove all lines that include tag ‘`miRNA_primary_transcript`’
3. Change chromosome names from format `chr1, chr2`, .. to format `1`, `2`,...
4. The 9. column is now format:
   `ID=MI0006363_1;Alias=MI0006363;Name=hsa-mir-1302-2`
   Change it to format:
   `gene_id “MI0006363_1”`
   You can omit the Alias and Name entries.
5. Sort the file by chromosome and by miRNA start position (4. column). Make sure to sort the chromosomes in numerical order, not in alphabetical (i.e 1,2,3... not 1,10,11..)
6. Output result to a file
7. Make another file that only has entries from chromosome 2

It’s possible to do the above in single command line, but you can use temporary files if you wish.

Extra exercise:

In Exercise 1 we only looked at sequences that did produce a match.

Use linux commands to generate a list of those query sequences that did **not** produce any hits.

There is no single right way to do this task. You probably need several commands and temporary files to get the result. You can use for example **`grep`** or **`diff`** commands.