Linux and Bioinformatics

So, we have seen an introduction to Linux, now how is this relevant to bioinformatics? Here are some brief examples that show the power of using Linux

1) The FASTQ format

There is an excellent page that described the FASTQ format here: http://en.wikipedia.org/wiki/FASTQ_format. This also explains how quality values are encoded in the file. Please read this document until you understand it! The vast majority of NGS data are now provided in FASTQ format, and a very simple example would be:

```
@SEQ_ID
GATTTGGGGTTCAAAGCAGTATCGATCAAATAGTAAATCCATTTGTTCAACTCACAGTTT
+
!''*((((***+))%%%++)(%%%%).1***-+*''))**55CCF>>>>>>CCCCCCC65
```

This is **one read**. As you can see, it consists of four lines. So the top four lines are the first read, the next four lines are the second read etc. The four lines are:

- @SEQ_ID - a unique identifier for the sequence read
- GATTTGGGGTTCAAAGC.... - the sequence read
- + - a separator between the read and the quality values (this sometimes replicates the sequence ID)
- !"*((((***+) .... - the quality values. For more information, see http://en.wikipedia.org/wiki/FASTQ_format#Encoding

2) Working with FASTQ on the command line

We have already seen how to count the number of reads in a file:

zcat training/rnaseq/ERR022486_chr22_read1.fastq.gz | grep ^@ERR | wc -l

We have also seen how we can browse the file using **less**:

zcat training/rnaseq/ERR022486_chr22_read1.fastq.gz | less

If we want, we can take subsets of the data using **head** and **tail**:

zcat training/rnaseq/ERR022486_chr22_read1.fastq.gz | head -n 40000 > top_10000.fastq

This command takes the top 10000 (NB: why do we ask for 40000 lines if we want the top 10000 reads?) and puts them into a file called top_10000.fastq. Similarly:

zcat training/rnaseq/ERR022486_chr22_read1.fastq.gz | tail -n 40000 > bottom_10000.fastq

Here we take the bottom 10000.

---

3) Using paste to get your data into a format you may prefer

FASTQ format is the format of data that all bioinformatics tools prefer and understand, however, sometimes it is preferable to look at data as a table i.e. in columns. We can do this with the **paste** command. The **paste** command writes lines in one file out as columns, separated by a tab character. The command can take "-" as an option, which means read from stdin. So if we give it the option "- - - -", this means "read four lines, and write it them out as four columns). Let's take a look:

zcat training/rnaseq/ERR022486_chr22_read1.fastq.gz | paste - - - - | less

Press Q to exit. This can be very useful, at times. For the purposes of this practical, run the following command:

zcat training/rnaseq/ERR022486_chr22_read1.fastq.gz | paste - - - - | head -n 1000 > top_1000_tab.txt

What do you think this does?

---

4) Using awk and working with data in columns

Now that we have some data in tabular format, we can look at using **awk**. Awk is a programming language which allows easy manipulation of structured data and the generation of formatted reports. Awk stands for the names of its authors “Aho, Weinberger, and Kernighan”. Awk commands look a bit like this:

awk '/search pattern/ {Actions}' *filename*

Briefly, awk goes through each line in *filename* and if the line matches the search pattern, the action is performed. Let's see some examples:

awk '/N/ {print}' top_1000_tab.txt

This simply prints out all lines in the file that contain an N - could be useful, but we can do that with **grep** anyway. What happens when you execute:

awk '/N/ {print $1,"\t",$2,"\t",$3,"\t",$4}' top_1000_tab.txt

?? What do you think the $1, $2 etc mean? How about the "\t"? Instead, try this:

awk '/N/ {print $1,"\t",$3}' top_1000_tab.txt

What happens now? Do you know what $1, $2, $3, $4 and $5 stand for now? Even better, we can use these in the match operation e.g.

awk '$3 ~ /N/ {print $1,"\t",$3}' top_1000_tab.txt

Up until now, we were asking the question "Does any part of the line contain an N?". In this last case, we are asking the question: for each line, does the data in column 3 contain an N? Note column 3 is the sequence, and so you may be interested in which lines have sequences that contain N's:

awk '$3 ~ /N/ {print $1}' top_1000_tab.txt

Identifies which reads contains N's.

awk '$3 ~ /N/ {print $1}' top_1000_tab.txt | wc -l

Counts the number of reads that contain N's. Note we can also do this by pipeing data into awk e.g.

cat top_1000_tab.txt | awk '$3 ~ /N/ {print $1}' | wc -l

---

5) Putting it together into something useful

How about a quick fastq to fasta converter?

zcat training/rnaseq/ERR022486_chr22_read1.fastq.gz | paste - - - - | awk {'print ">"$1,$2,"\n"$3}'

This is very crude, but in brief: we use zcat to unzip the fastq file, which we pipe into paste to convert to tabular data. We pipe this into awk. As we don't use a search pattern, awk accepts every line. First we print out the ">", and then columns 1 and 2 to make up the identifer. We then print out a new line ("\n") and then finally the third column which is the sequence.

---

6) Advanced

OK, you're on your own here, how about:

- Download the file ftp://ftp.sra.ebi.ac.uk/vol1/fastq/SRR026/SRR026762/SRR026762.fastq.gz (hint: use **wget**)
- Information about this study can be found here: http://www.ebi.ac.uk/ena/data/view/SRR026762
- How many lines are in the file?
- How many sequences are in the file?
- This is an miRNA study. These reads are 36bp in length, yet microRNAs are only 21-23bp. After the sequencer has read the microRNA, it starts reading the illumin adapters
- The 3' small RNA adapter from illumina has the sequence: ATCTCGTATGCCGTCTTCTGCTTG
- How many reads contain this sequence?
- How many reads have a partial version of this sequence?

You may wish to try this if you get bored: http://www.ark-genomics.org/events-online-training/counting-known-micrornas-five-easy-steps

---

Next - Assessing quality of Illumina data