Adapter and quality trimming of illumina data

From the previous section, looking at over-represented sequences, we can see that fastqc thinks the following *may* be in our dataset:

- Illumina PCR Primer Index 1
- RNA PCR Primer, Index 47
- Illumina Single End Adapter 2
- Illumina Paired End PCR Primer 2

What are these sequences though? Well, Illumina send these sequences to customers as a letter in PDF format. To get this letter, you must register at www.illumina.com and request it. However, I can give the sequences below:

```
>PCRPrimerIndex1
CAAGCAGAAGACGGCATACGAGATCGTGATGTGACTGGAGTTC
>RNAPCRPrimerIndex47
CAAGCAGAAGACGGCATACGAGATCTTCGAGTGACTGGAGTTCCTTGGCACCCGAGAATTCCA
>IlluminaSingleEndAdapter2
CAAGCAGAAGACGGCATACGAGCTCTTCCGATCT
>PEPCRPrimer2.0
CAAGCAGAAGACGGCATACGAGATCGGTCTCGGCATTCCTGCTGAACCGCTCTTCCGATCT
```

As you can see, each of these sequences have a lot in common!

The list that FastQC actually uses is here: **/usr/share/java/fastqc-0.10.1/Contaminants/contaminant_list.txt**

We must also remember to consider the reverse complement of these sequences as FastQC will look for both!

```
>PCRPrimerIndex1_RC
GAACTCCAGTCACATCACGATCTCGTATGCCGTCTTCTGCTTG
>RNAPCRPrimerIndex47_RC
TGGAATTCTCGGGTGCCAAGGAACTCCAGTCACTCGAAGATCTCGTATGCCGTCTTCTGCTTG
>IlluminaSingleEndAdapter2_RC
AGATCGGAAGAGCTCGTATGCCGTCTTCTGCTTG
>PEPCRPrimer2.0_RC
AGATCGGAAGAGCGGTTCAGCAGGAATGCCGAGACCGATCTCGTATGCCGTCTTCTGCTTG
```

Now, I'm going to let you into a little secret - the adapter that is **actually** present in this data set is:

```
>SmallRNA3pAdapter_1.5
ATCTCGTATGCCGTCTTCTGCTTG
```

And if we look at again at the reverse complement of the adapters reported by FastQC, we can see:

```
>PCRPrimerIndex1_RC
GAACTCCAGTCACATCACGATCTCGTATGCCGTCTTCTGCTTG
>RNAPCRPrimerIndex47_RC
TGGAATTCTCGGGTGCCAAGGAACTCCAGTCACTCGAAGATCTCGTATGCCGTCTTCTGCTTG
>IlluminaSingleEndAdapter2_RC
AGATCGGAAGAGCTCGTATGCCGTCTTCTGCTTG
>PEPCRPrimer2.0_RC
AGATCGGAAGAGCGGTTCAGCAGGAATGCCGAGACCGATCTCGTATGCCGTCTTCTGCTTG
```

So, there are lessons to be learned here:

- FastQC can tell us about adapter contamination, but it may not tell us the correct adapter
- Detecting the adapters present in our data can take some detection work

One tip is to always ask the scientist who produced the data, but sometimes even they do not know.

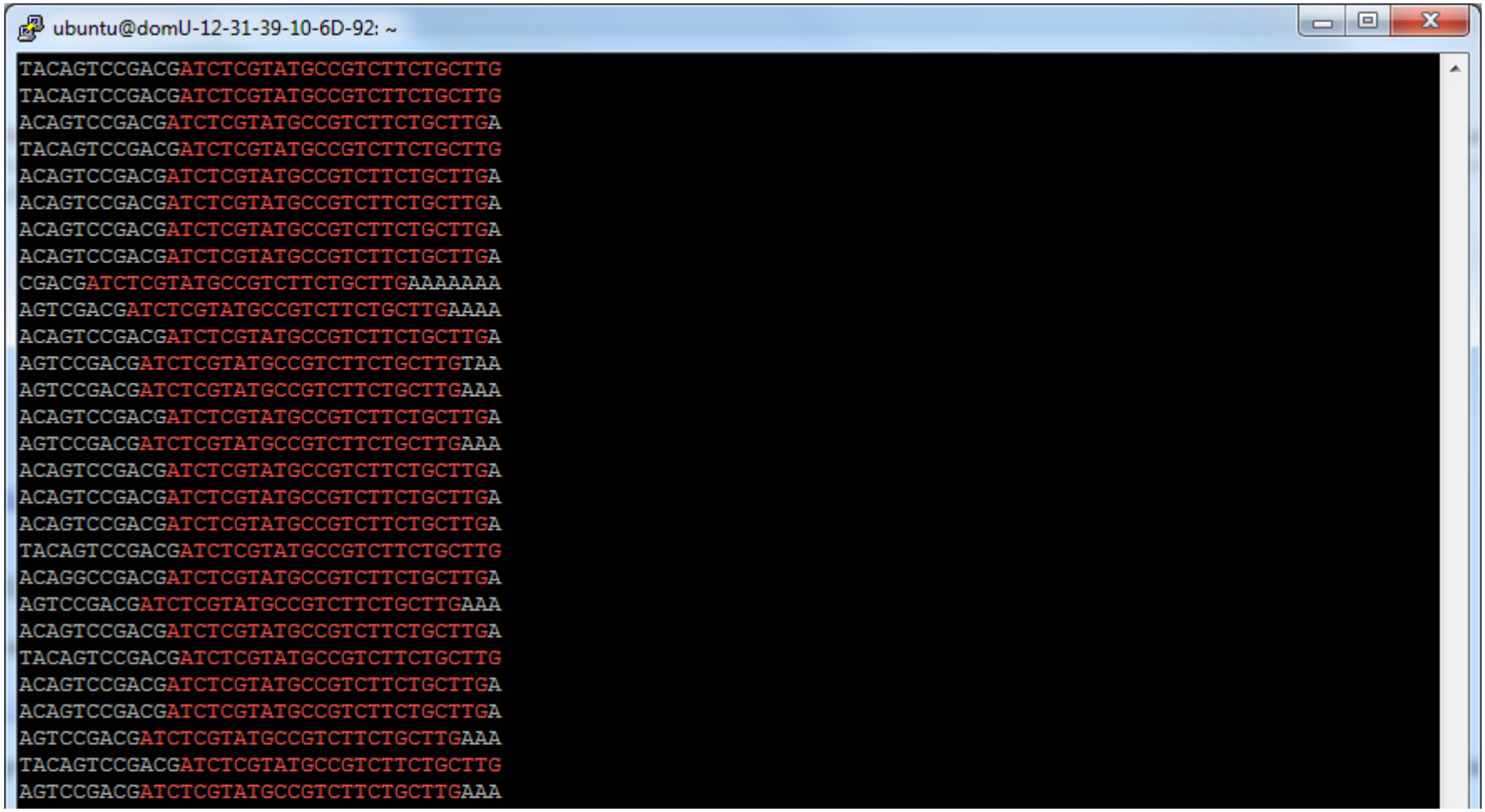
We are not finished yet, and this can get more complicated!

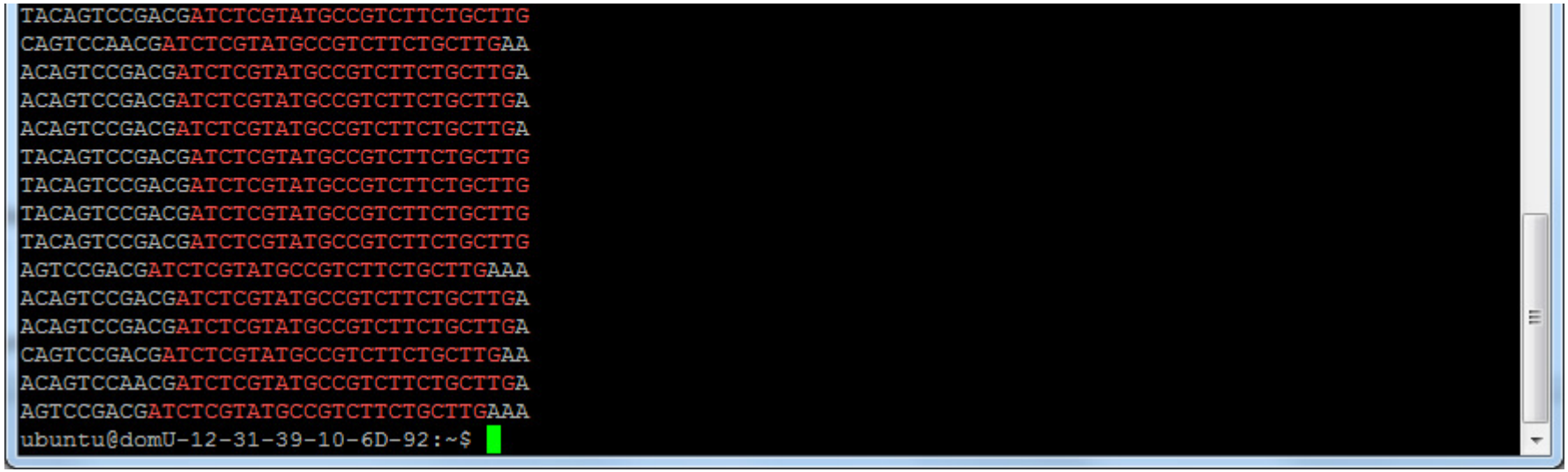
Let's look for our adapter:

cd

zcat SRR026762.fastq.gz | grep ATCTCGTATGCCGTCTTCTGCTTG | head -n 1000

Your putty window should look something like this:





Now, if you are sharp, you will notice that next to our adapter there is another repetative sequence!

```
TACAGTCCGACGATCTCGTATGCCGTCTTCTGCTTG
TACAGTCCGACGATCTCGTATGCCGTCTTCTGCTTG
TACAGTCCGACGATCTCGTATGCCGTCTTCTGCTTG
TACAGTCCGACGATCTCGTATGCCGTCTTCTGCTTG
```

It turns out this is part of the Small RNA sequencing primer:

```
>SmallRNASequencingPrimer
CGACAGGTTCAGAGTTCTACAGTCCGACGATC
```

When we combine the two, we get:

```
CGACAGGTTCAGAGTTCTACAGTCCGACGATC                          <- Small RNA Primer
                  TACAGTCCGACGATCTCGTATGCCGTCTTCTGCTTG     <- Our read(s)
                              ATCTCGTATGCCGTCTTCTGCTTG     <- Small RNA 5' adapter
```

Essentially, what is happening here, is that during the small RNA library prep stage, small RNA fragments are selected. This inevitable ends up with a huge amount of "primer-dimer" i.e. where the Illumina primers/adapters have no actual product between them.

So how do we deal with adapter contamination?

---

1) Using cutadapt to remove adapters

Cutadapt is tool specifically designed to remove adapters from NGS data. To see the full list of help and options, type:

cutadapt -h

As we have many adapters in our data, we are going to run cutadapt with all of them. The command is:

```
cutadapt -a CGACAGGTTCAGAGTTCTACAGTCCGACGATC \
         -a TACAGTCCGACGATC \
         -a ATCTCGTATGCCGTCTTCTGCTTG \
         -e 0.1 -O 5 -m 15 \
         -o SRR026762_adaprm.fastq SRR026762.fastq.gz
```

The options mean:

- -a CGACAGGTTCAGAGTTCTACAGTCCGACGATC : the first adapter to remove, this is the full Small RNA primer
- -a TACAGTCCGACGATC : the second adapater to remove, this is the portion of the Small RNA primer that we have observed
- -a ATCTCGTATGCCGTCTTCTGCTTG : the third adapter to remove, this is the full Small RNA 5' adapter
- -e 0.1 : allow a mismatch rate of 1 mismatch in ten bases between the read and the adapters
- -O 5 : The overlap must be at least 5 base-pairs
- -m 15 : after trimming, reads less than 15bp are thrown out
- -o SRR026762_adaprm.fastq : put the trimmer data in this file

The output of the command will look something like:

```
Maximum error rate: 10.00%
   Processed reads: 5677631
     Trimmed reads: 1927223 ( 33.9%)
   Total basepairs:    204394716 (204.4 Mbp)
 Trimmed basepairs:     41594144 (41.6 Mbp) (20.35% of total)
```

```
   Too short reads: 1016544 ( 17.9% of processed reads)
    Too long reads: 0 (  0.0% of processed reads)
        Total time:    287.36 s
     Time per read:      0.05 ms

=== Adapter 1 ===

Adapter 'CGACAGGTTCAGAGTTCTACAGTCCGACGATC', length 32, was trimmed 333 times.

Lengths of removed sequences
length  count   expected
5       331     5544.6
7       1       346.5
10      1       5.4

=== Adapter 2 ===

Adapter 'TACAGTCCGACGATC', length 15, was trimmed 52873 times.

Lengths of removed sequences
length  count   expected
5       93      5544.6
6       1121    1386.1
7       1082    346.5
8       7       86.6
9       2       21.7
10      3       5.4
11      2       1.4
>=17    50563   0.1

=== Adapter 3 ===

Adapter 'ATCTCGTATGCCGTCTTCTGCTTG', length 24, was trimmed 1874017 times.

Lengths of removed sequences
```

```
length  count   expected
5       322     5544.6
6       90      1386.1
7       162     346.5
8       20      86.6
9       132     21.7
10      384     5.4
11      1257    1.4
12      4026    0.3
13      10585   0.1
14      33836   0.0
15      84695   0.0
16      104889  0.0
17      86715   0.0
18      77590   0.0
19      121253  0.0
20      199881  0.0
21      182110  0.0
22      205659  0.0
23      184010  0.0
24      232069  0.0
25      137080  0.0
>=26    207252  0.0
```

You can see from this output that over 33% of reads had some form of adapter in them! Also, over 17% of the reads were subsequently shorter than 15bp and discarded. The results are in **SRR026762_adaprm.fastq**

How many reads are in SRR026762_adaprm.fastq?

Can you think of a combination of **cat**, **paste** and **awk** that would print out a column of sequence lengths?

Can you think of how you might add **sort**, and **uniq -c** to count how many there are of each length?

(something like **cat SRR026762_adaprm.fastq | paste - - - - | awk '{print length($3)}' | sort | uniq -c**)

---

2) Using sickle to trim reads based on quality

As we have seen, with Illumina reads (the same is true for many technologies) the quality of the data begins to fade towards the end. It can therefore be important to trim off these poor quality bases, in case we mistakenly assemble them into a genome, or use them to call SNPs. We recommend **sickle** to do this, which is a quality trimmer which is paired-end aware.

What do you mean, paired-end aware?!

Paired-end sequencing is where we sequence both ends of a longer DNA fragment. Thus we know that these paired-reads come from the same fragment, plus how far apart they should be, and this is very useful when we come to align and assemble them. The most common representation of this is to have two files, the first containing "read 1" and the second "read 2". Thus, the first read in the first file is paired with the first read in the second file, the second read paired with the second read etc etc etc.

Why is this an issue?

Well, the **order** of reads in each file needs to be maintained. If we remove a read from one file because it is poor quality, or it contains adapter, then we must also remove the corresponding read from the other file. **Sickle** does this very well. Unfortunately, **cutadapt** does not!

Type:

sickle

sickle se

sickle pe

This should show the help for each command. The microRNA dataset we have just trimmed adapters from is a single-end sequence file, so we can trim based on quality by running:

sickle se -f SRR026762_adaprm.fastq -t sanger -o SRR026762_adaprm_trim.fastq -q 30 -l 15

The options are:

- se : use single end mode
- -f SRR026762_adaprm.fastq : this is the input fastq file
- -t sanger : the quality encoding. All data downloaded from EBI or NCBI will be "sanger" encoded. For an explanation: http://en.wikipedia.org/wiki/FASTQ_format#Encoding
- -o SRR026762_adaprm_trim.fastq : the output file
- -q 30 : the quality value to use. Bases below this will be trimmed, using a sliding window
- -l 15 : the minimum length allowed after trimming. Here we remove reads with less than 15bp

The output will be something like:

```
FastQ records kept: 2622021
FastQ records discarded: 2039066
```

As you can see, a lot of reads have been discarded.

Try trimming using a lower quality value (e.g. 20) - how many reads are kept then?

Try allowing shorter reads to be kept (e.g. 10bp) - how many reads are kept then?

How about trimming paired reads? We have paired reads in the **training** directory:

ubuntu@domU-12-31-39-10-6D-92:~$ ls -l training/rnaseq/*.fastq.gz
-rw-rw-r-- 1 ubuntu ubuntu 65721000 Oct 5 2011 training/rnaseq/ERR022486_chr22_read1.fastq.gz
-rw-rw-r-- 1 ubuntu ubuntu 62899231 Oct 5 2011 training/rnaseq/ERR022486_chr22_read2.fastq.gz
-rw-rw-r-- 1 ubuntu ubuntu 41417843 Oct 5 2011 training/rnaseq/ERR022488_chr22_read1.fastq.gz
-rw-rw-r-- 1 ubuntu ubuntu 39405527 Oct 5 2011 training/rnaseq/ERR022488_chr22_read2.fastq.gz

The command we might use to trim one of these would be:

```
sickle pe -f training/rnaseq/ERR022486_chr22_read1.fastq.gz \
          -r training/rnaseq/ERR022486_chr22_read2.fastq.gz \
          -t sanger \
          -o ERR022486_chr22_read1_trim.fastq \
          -p ERR022486_chr22_read2_trim.fastq \
          -s ERR022486_chr22_single_trim.fastq \
          -q 30 -l 15
```

An explanation of the paramters:

- pe : use paired-end mode
- -f training/rnaseq/ERR022486_chr22_read1.fastq.gz : the fastq file for read 1
- -r training/rnaseq/ERR022486_chr22_read2.fastq.gz : the fastq file for read 2
- -t sanger : the quality encoding. All data downloaded from EBI or NCBI will be "sanger" encoded. For an explanation: http://en.wikipedia.org/wiki/FASTQ_format#Encoding
- -o ERR022486_chr22_read1_trim.fastq : the output file for trimmed reads from read 1
- -p ERR022486_chr22_read2_trim.fastq : the output file for trimmed reads from read 2
- -s ERR022486_chr22_single_trim.fastq : the output file for reads where the mate has failed the quality or length filter

- -q 30 : the quality value to use. Bases below this will be trimmed, using a sliding window
- -l 15 : the minimum length allowed after trimming. Here we remove reads with less than 15bp

The output will look something like this

FastQ paired records kept: 1357006 (678503 pairs)
FastQ single records kept: 118391 (from PE1: 89173, from PE2: 29218)
FastQ paired records discarded: 106702 (53351 pairs)
FastQ single records discarded: 118391 (from PE1: 29218, from PE2: 89173)

What happens if you set a quality threshold of 20?

What happens if you set the length threshold at 10? At 0?

And there we have it! You are now able to trim adapters and assess the quality of single- and paired-end Illumina data.

If you have more spare time, feel free to browse the EBI European Nucleotide Archive, download some data and practice some of your new-found skills (e.g. http://www.ebi.ac.uk/ena/data/view/SRP008449)