Analysis of Next Generation Sequencing Data
More loop constructs; data download and QC

Luce Skrabanek

28 January, 2020

1 More loop constructs

The `if` statement processes a list of commands if some condition is met. This loop structure can become very complex, as you can have any number of conditions, and different sets of commands that must be processed depending on whether the conditions are met or not.

The `test` command evaluates various expressions or determines whether they are true or false. There are a number of expressions that can be used, some of which are listed below. They are often used in the conditional loops that are detailed below. Expression tests return integers. When an expression is true, a value of 0 is returned, otherwise it returns a non-zero integer.

The spaces used in the list below are important. Note that you must also have spaces next to the brackets in the test expression.

Some of the more common tests include:

`-l <string>`	Returns the length of string
`string1 = string2`	Returns true (0) if string1 is equal to string2
`string1 != string2`	Returns true if string1 is not equal to string2
`integer1 -eq integer2`	Returns true if integer1 = integer2
`integer1 -ne integer2`	Returns true if integer1 != integer2
`integer1 -gt integer2`	Returns true if integer1 > integer2
`integer1 -ge integer2`	Returns true if integer1 ≥ integer2
`integer1 -lt integer2`	Returns true if integer1 < integer2
`integer1 -le integer2`	Returns true if integer1 ≤ integer2
`!  expr`	Returns true if expr is false
`expr1 -a expr2`	Returns true if both expr1 and expr2 are true
`expr1 -o expr2`	Returns true if either expr1 or expr2 is true

```
filelength=$(cat PRJEB5348.txt | wc -l)
if [ $filelength -gt 25 ]; then
  echo "Use less to view this file"
else
  echo "Use cat to view this file"
fi
```

Note the `if` construct syntax: `if [ ]; then ... fi`.

You can see the complete list by doing `man test`, as well as other testing options (like `-r`).

The `read` command is useful for reading input (either from a file or from an interactive user at the terminal) and assigning the results to a variable.

```
read name
echo Hello $name
```

Another loop construct is the `while` loop. This can take an expression, such as a `[]` test expression, which is evaluated and used as the condition for the loop to continue or terminate. The syntax is `while [ ]; do ... done`. You can also use the `read` command to read input from a file, and feed it into a `while` loop

```
while read line; do
  echo $line | wc -c
done < quotation
```

2 Downloading a FASTQ file and running FastQC

We have already seen some of the data from Gierlinski et al (2015), the most comprehensive RNA-seq dataset to date, containing mRNA from 48 replicates of two *S. cerevisiae* populations: wildtype and *snf2* knock-out mutants. All 96 samples were sequenced on one flowcell (Illumina HiSeq 2000); each sample was distributed over seven lanes, which means that there are seven technical replicates per sample. The accession number for the entire data set (consisting of 7 x 2 x 48 raw read files) is ERP004763.

We are going to download raw sequence data from the European instance of the SRA, which can be accessed via `https://www.ebi.ac.uk/ena/browser/`. At ENA, the sequencing reads are directly available in `FASTQ` format.

To download a set of `FASTQ` files:

1. Go to `https://www.ebi.ac.uk/ena/browser/`.
2. Search for the accession number of the project, e.g., ERP004763 (should be indicated in the published paper).
3. There are several ways to start the download from the command line:
   (a) Copy the link's address of the "Fastq files" column:
   (b) If there are many samples within one project, you can download the summary of the sample information from the "TSV" button. On the command line, go to the folder where you will store the data and use the appropriate column of the `TSV` file you just downloaded to feed individual `FTP URLs` of the different samples to the `wget` command:

```
wget -O PRJEB5348.txt 'https://www.ebi.ac.uk/ena/portal/api/
   filereport?accession=PRJEB5348&result=read_run&fields=
   fastq_ftp&format=tsv&download=true'
cut -f 2 <SampleInformation.text> | xargs wget
```

2.1 FastQC

`FastQC` is released by the Babraham Institute and can be freely downloaded at `https://www.bioinformatics.babraham.ac.uk/projects/fastqc/`. We can access it on our systems using:

```
spack load fastqc # has to be done once per session
```

To run `FastQC` on the first WT FASTQ file, use the following command:

```
fastqc ERR458493.fastq.gz --extract
```

What is the difference in output between running FastQC with and without the `--extract` option?

Other useful options include:

1. `-o/--outdir`
   Creates the output files in the specified directory, which must already exist.
2. `--nogroup`
   Disables grouping of bases for reads >50bp. All reports will show data for every base in the read.
3. `-t/--threads`
   Specifies the number of files which can be processed simultaneously.
4. `-a/--adapters`
   Specifies a file which contains the list of adapter sequences to search for (formatted as name[tab]sequence).

3 Adapter trimming

There are many adapter trimming tools available. We are going to use one called `TrimGalore`, which is a wrapper around `cutadapt`, written by the same group that developed FastQC.

TrimGalore knows about the most common adapters for many sequencing types, so you don't have to enter them yourself. TrimGalore (`https://github.com/FelixKrueger/TrimGalore/blob/master/Docs/Trim_Galore_User_Guide.md`) will either determine the adapter being used by scanning the first million sequences in the file, or you can supply that information yourself (using the options `--illumina`, `--nextera`, `small_rna`, or by supplying the actual adapter sequence(s) with `-a/--adapter ADAPTER`. While this does streamline the adapter trimming process, it is a good idea to check that the adapters used to generate your data where correctly identified.

TrimGalore available on our systems through spack:

```
spack load -r trimgalore
trim_galore --help
trim_galore --illumina illumina_100K.fastq.gz
```

3.1 TrimGalore output

The output from TrimGalore/cutadapt gives you a summary of the parameters that were used to do the trimming, including the adapter sequence itself, and tells you how many reads were processed and how many bases were trimmed off.

For each adapter, the output indicates the sequence, its length, and how many times it was trimmed. It also lists, as a reminder, and based on your parameter of 'Maximum trimming error rate', the maximum number of errors allowed per length of matched sequence.

The section "Overview of removed sequences" shows more detailed information on the sequences that were trimmed off. Specifically, each row tells us:

1. **Length** How long the removed sequence was. Note that this refers to the length of the removed sequence, not the length of the match!
2. **count** How many sequences of that length were removed.
3. **expect** Gives a rough estimate of the number of sequences that would have been expected to match randomly. If we see that the expected number is similar to the actual count, it might indicate that we should set a more stringent threshold of how many bases must match (instead of the default of 1).
4. **max.err** Maximum number of errors allowed in a sequence of that length.
5. **error counts** The number of removed sequences that had 0, 1, 2, etc mismatches.

Further information on how to read the output can be found in the cutadapt manual at `https://cutadapt.readthedocs.io/en/stable/guide.html`.

We can now use FastQC to compare the sequences, before and after trimming.

```
fastqc illumina_100K.fastq.gz  --extract
fastqc illumina_100K_trimmed.fq.gz  --extract
```

4 Downloading using SRA-toolkit

In some cases, it may be more convenient to access data from another source. Another common data source is NCBI. NCBI hosts the same SRA data as ENA, although it must be accessed via their own tool called `sra-toolkit`, which is available on our systems through spack.

```
spack load sra-toolkit
```

From this suite, the tool used to download data is `prefetch` (`https://github.com/ncbi/sra-tools/wiki/HowTo:-Access-SRA-Data`). To download the first file in the Gierlinski dataset:

```
prefetch ERR458493   # downloads in compressed format to a cache area
srapath ERR458493    # lists the location of the cached file
fastq-dump ERR458493 # converts the cached file to FASTQ
```

If your data is in SFF format, use the following command to convert the data from SFF format to FASTQ:

```
sff-dump ERR458493
```