Introduction to genomic data

Todd W. Pierson

16 October 2019

Contents

In this tutorial, we'll work through a basic introduction to handling raw genomic data. This will include brief lessons on:

1) formatting of FASTQ files;
2) multiplexing strategies and how to demultiplex reads; and
3) quality-filtering

0.0.1 Interpreting a FASTQ file

Most raw genomic data come in the FASTQ format. For comparison, it is worth reviewing the structure of a FASTA format.

We can peek at the first two lines of an example FASTA file of the `data` directory. If you are following along, the commands we'll use in this tutorial should be run in your Terminal window, rather than in `R`.

```
cat data/example.fasta
```

```
>sample_name
GGTCAACAAATCATAAAGATATTGGGACATTGTATATGATTTTTGGAACTTTGTCAGGAGTGGTGGGAACAACTTTGTCAGTCTGG
```

In the FASTA file, each sequence has two lines dedicated to it: one for the sample or sequence name and the other for the actual sequence. The FASTQ format is built off of this same premise, but it has a bit more information included.

We can peek at an example FASTQ file in the same directory.

```
cat data/example_R1.fastq
```

```
@M00313:165:000000000-D6BK4:1:1101:15780:1379 1:N:0:CAGAGTGT+CGATCGAT
GGTCAACAAATCATAAAGATATTGGGACATTGTATATGATTTTTGGAACTTTGTCAGGAGTGGTGGGAACAACTTTGTCAGTCTGG
+
BBABBBFFAFFFBGB5BBAFBGHGHHFHGFHHHHHGHHDHHHHHGGHHHHGHHGFHGHGFBFFDFDGFHCGHHHHHHHHHHHHHHH
```

This file clearly contains more information! FASTQ files have four lines per sequencing read; they represent:

1) sample information in a header; this string contains information about the sequencing platform, the physical position on the "tile" where this sequencing reaction took place, and (often) index information
2) the actual sequence read
3) "+"; this is just a placeholder, although sometimes other information is included on this line
4) quality score for each nucleotide of the read; this is the "Q" in "FASTQ" and is important for downstream applications.

The Wikipedia page for FASTQ files has much more useful information. In our simple example here, we can see the dual-index sequences (`CAGAGTGT` and `CGATCGAT`) at the end of the first line (more about that kind of information in the next section). Each of these four lines is present in a FASTQ file for *each molecule sequenced*. That means that for some super high-throughput platforms (e.g., the Illumina NovaSeq, which can generate up to 2.2 billion reads!), these FASTQ files can be huge.

You may notice that the FASTQ file we viewed has a "R1" designator in the file name. This is because on many sequencing platforms, each molecule is sequenced from both directions—generating a "read 1" (i.e., "R1") and "read 2" ("R2") read. Thus, we have a corresponding "R2" file. Let's look at it:

```
cat data/example_R2.fastq
```

```
@M00313:165:000000000-D6BK4:1:1101:15780:1379 1:N:0:CAGAGTGT+CGATCGAT
GGTCAACAAATCATAAAGATATTGGGACATTGTATATGATTTTTGGAACTTTGTCAGGAGTGGTGGGAACAACTTTGTCAGTCTGG
+
BBABBBFFAFFFBGB5BBAFBGHGHHFHGFHHHHHGHHDHHHHHGGHHHHGHHGFHGHGFBFFDFDGFHCGHHHHHHHHHHHHHHH
```

Note that the information from the first line is identical.

0.0.2 Multiplexing and demultiplexing

The best way to leverage the power of next-generation sequencing technologies to generate data for many individuals (e.g., for a typical population genetic or phylogenetic project) is to "multiplex" many samples onto a single sequencing run. For example, some Illumina NextSeq runs can generate 400 paired-end reads per sequencing lane. For many purposes (e.g., amplicon data from a microbiome project or RADseq-style data for a population genetic project), this amount of data would be incredible overkill (and cost-prohibitive!), so it's useful to spread these reads across many samples.

To do this, we probably want to mark molecules during the library preparation stage so that we can later determine which sample they came from. This is often called "indexing" or "tagging" (or confusing, "barcoding"), and the act of pooling these samples for sequencing is "multiplexing". Thus, when we receive our raw data, one of our first tasks is "demultiplexing"—or separate our reads by sample of origin. These indexes may be read in separate sequencing reads and stored in the FASTQ headers, or they could be in the actual sequencing read (sometimes called "in-line" indexes). There are many ways to skin a cat, and many application-specific software packages have their own way of demultiplexing reads.

0.0.3 Quality filtering

As discussed earlier, FASTQ files contain quality scores associated with each nucleotide of each read. Two general rules: 1) quality scores are generally lowest at the beginning and (especially) the end of reads; and 2) R1 data typically have higher quality scores than R2 data.

Why do these quality scores matter? Simply put, they tell us information about the probability that a given nucleotide is called in error. To reduce the probability of including these erroneous data in our analyses, we can ,for example, trim sections of reads that have low quality scores or throw out reads altogether that have low overall quality scores. There are many, many ways to visualize quality score data and filter reads.

We'll use a quick visualization in `R`. If you haven't already, install the `seqTools` package (installation instructions here). Then, open `R` and load the package.

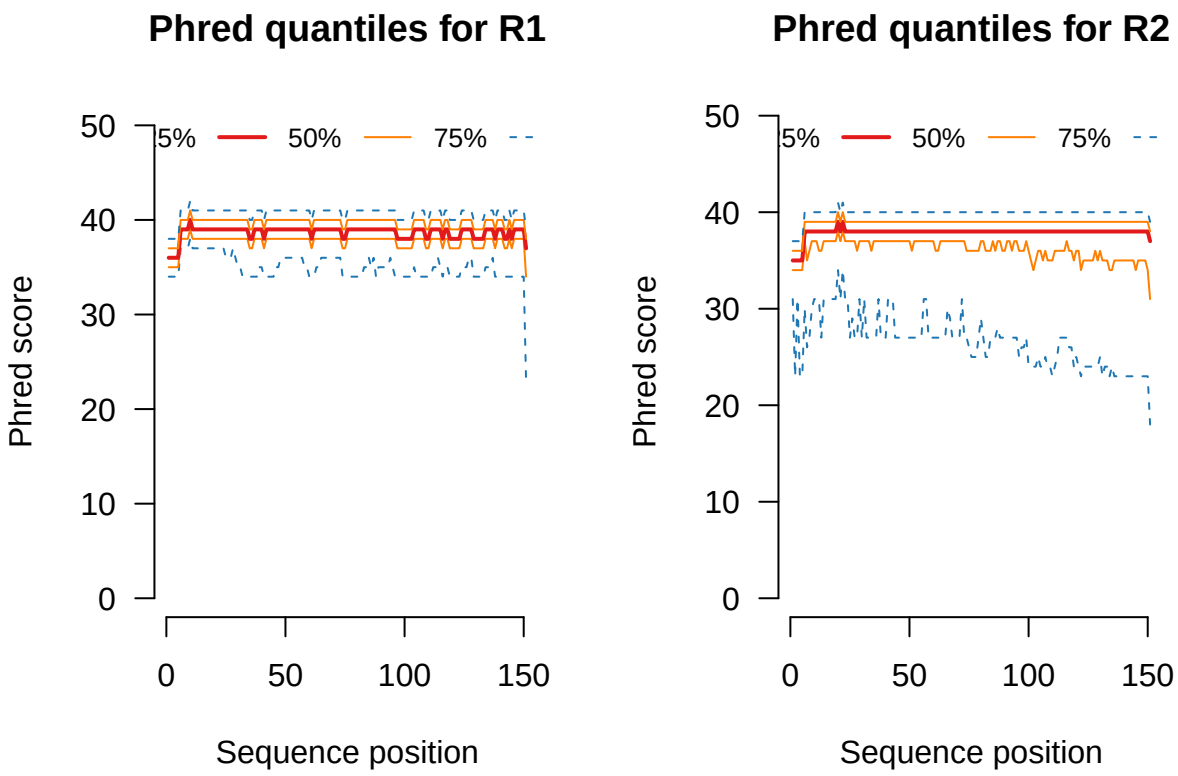
```
library(seqTools)
```

Next, we'll separately load our R1 and R2 data.

```
fq1 <- fastqq(c("data/eDNA_08_R1.fastq"))
fq2 <- fastqq(c("data/eDNA_08_R2.fastq"))
```

Then, we can plot our quality scores.

```
par(mfrow = c(1,2))
plotMergedPhredQuant(fq1, main = "Phred quantiles for R1")
plotMergedPhredQuant(fq2, main = "Phred quantiles for R2")
```



Now that we've covered some of the basic concepts of interpreting and handling genomic data, we'll move on to assembling an example dataset.