Using the qrqc package to gather information about sequence qualities

Vince Buffalo

Bioinformatics Core
UC Davis Genome Center

vsbuffalo@ucdavis.edu

2010-01-19

Abstract

Many projects in bioinformatics begin with raw sequences from a high-throughput sequencer that must be quality checked before additional analysis steps can proceed. The *qrqc* (Quick Read Quality Control) package is a fast and extensible package that reports basic quality and summary statistics on FASTQ and FASTA files, including base and quality distribution by position, sequence length distribution, and common sequences.

1 Reading in a Sequence File

The *qrqc* package reads and processes FASTA and FASTQ files in C for speed, through the function `readSeqFile`. Optionally, sequences can be hashed (also done at the C level), to see the most frequent sequences in the file.

```
> library(qrqc)
> s.fastq <- readSeqFile(system.file('extdata', 'test.fastq', package='qrqc'))
```

Note that there is a maximum sequence length argument in `readSeqFile`, `max.length`. By default, this is 1,000. It is used to pre-allocate the matrices in C, and could be much larger than the largest sequence encountered without many downsides (its memory usage is relatively low). If a sequence larger than `max.length` is encountered, the function will stop, and the user can call the function again with a larger `max.length`.

`Readseqfile` produces a `FASTQSummary` object, which inherits from the `SequenceSummary` class. Printing the object lists a very short summary:

```
> s.fastq
```

```
Quality Information for: test.fastq
 100 sequences, 96 unique
 mean quality: 32.673900
 min sequence length: 84
 max sequence length: 84
```

Optionally, `readSeqFile` can be run without hashing, with `hash=FALSE`. `readSeqFile` also works on FASTA files, but `type=FASTA` must be specified.

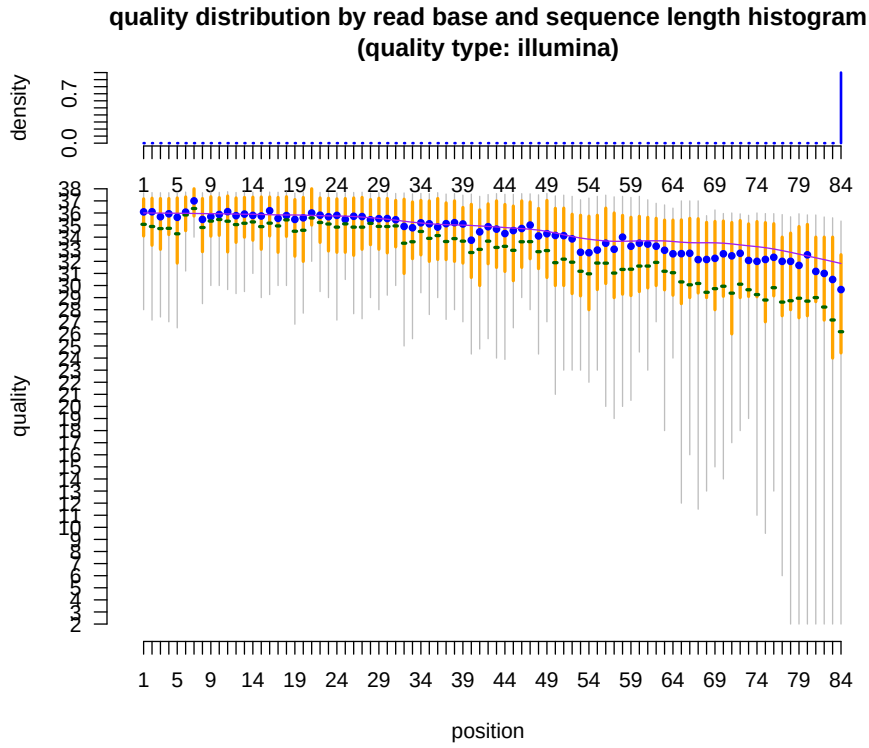

Figure 1: A plot of quality by base position, with sequence length histogram.

2 Plotting Quality of FASTQSummary Objects

If the file read and summarized with `readSeqFile` is a FASTQ file (and thus the resulting object is from the `FASTQSummary` class), quality information by position can be plotted with `plotQuals`, which produces a graphic as in Figure 1:

```
> plotQuals(s.fastq)
```

By default, quality plots include a histogram of sequence lengths. This is recommended, as low quality at a particular base position is less worrisome if there are few reads with this sequence length. The grey lines indicate the 10% and 90% quantiles, orange lines indicate the lower and upper quartiles, the blue dot is the median, and the green dash the mean. A purple lowess curve is fit through the distributions. This line is fit by first randomly drawing values from the empirical (binned) distribution of qualities at a particular base, then fitting through these points.

`plotQuals` can be very useful in inspecting base qualities before and after read quality control pipelines. For example, the package contains `test-trimmed.fastq`, which has the same sequences as `test.fastq` after being trimmed with Nik Joshi's `Sickle`, a windowed adaptive quality trimmer.

```
> s.trimmed.fastq <- readSeqFile(system.file('extdata', 'test-trimmed.fastq', package='qrqc'))
> plotQuals(s.trimmed.fastq)
```

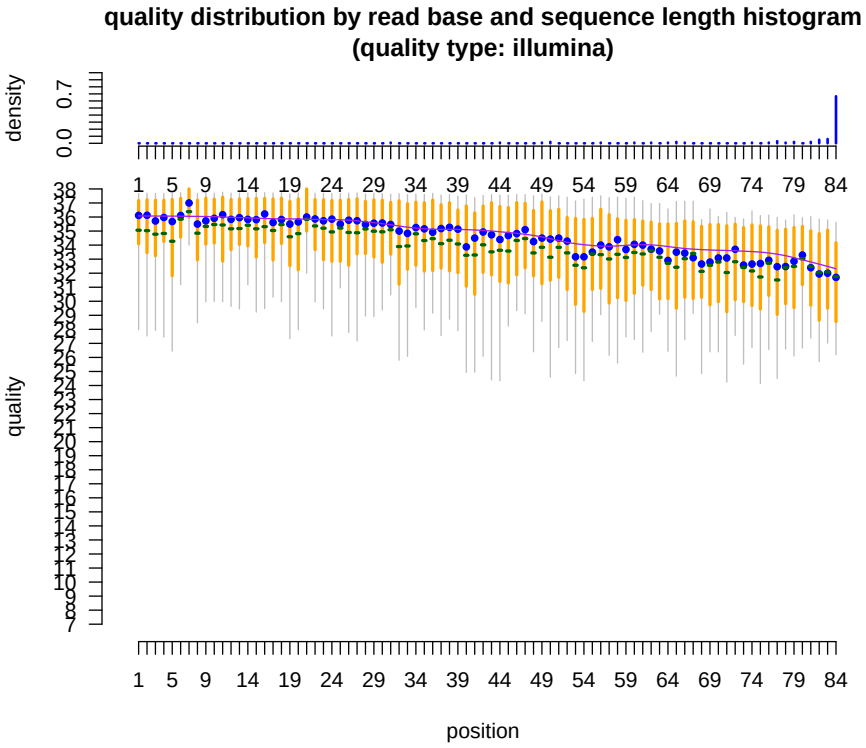

Figure 2: A plot of quality by base position after being trimmed with `Sickle`.

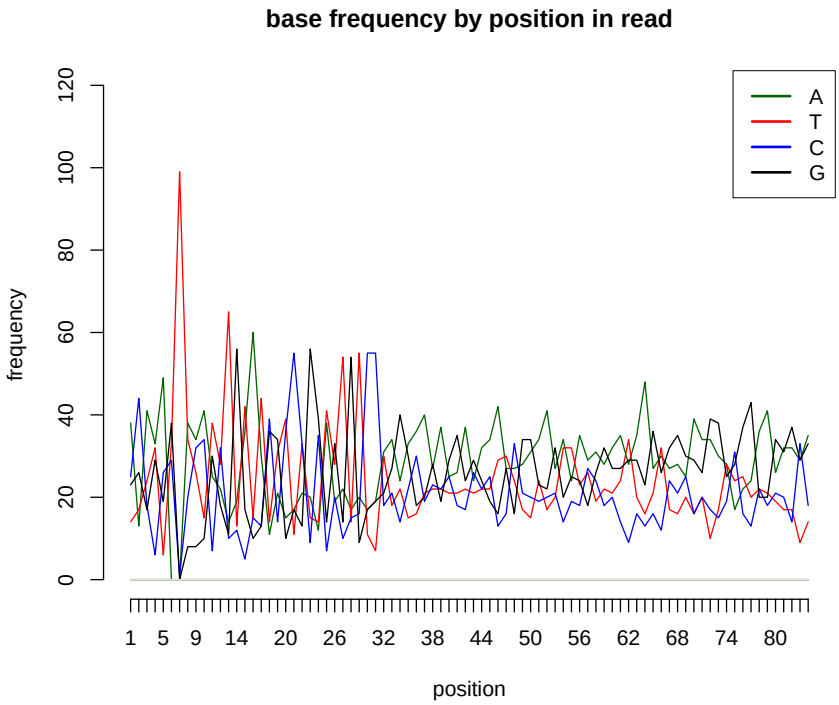

Figure 3: Base frequencies by position in sequence.

3 Other Plots of FASTQSummary and FASTASummary Objects

The following S4 methods work for FASTQSummary and FASTASummary objects. Base frequencies (counts) and base proportions by position can be plotted with `plotBases`. When used with `type='freq'`, `plotBases` produces a graphic as in Figure 3:

```
> plotBases(s.fastq, type="freq")
```

`plotBases` uses the Sanger base color scheme: blue is Cytosine, green is Adenine, black is Guanine, red is Thymine, and purple in N (any base). When further IUPAC nucleotides are found in the sequences, a legend is added.

`plotBases` also accepts a `bases` parameter, which can be used to specify specific bases. This is useful for plotting just the frequency of 'N'.

Base proportions by position can be plotted with `plotBases`, with `type='prop'`. This plot is basically identical to the plot produced with `type='freq'` with a different y scale:

```
> plotBases(s.fastq, type="prop")
```

Sequence length distribution can be plotted with `plotSeqLengths` (graphic shown in Figure 5):

```
> plotSeqLengths(s.trimmed.fastq)
```

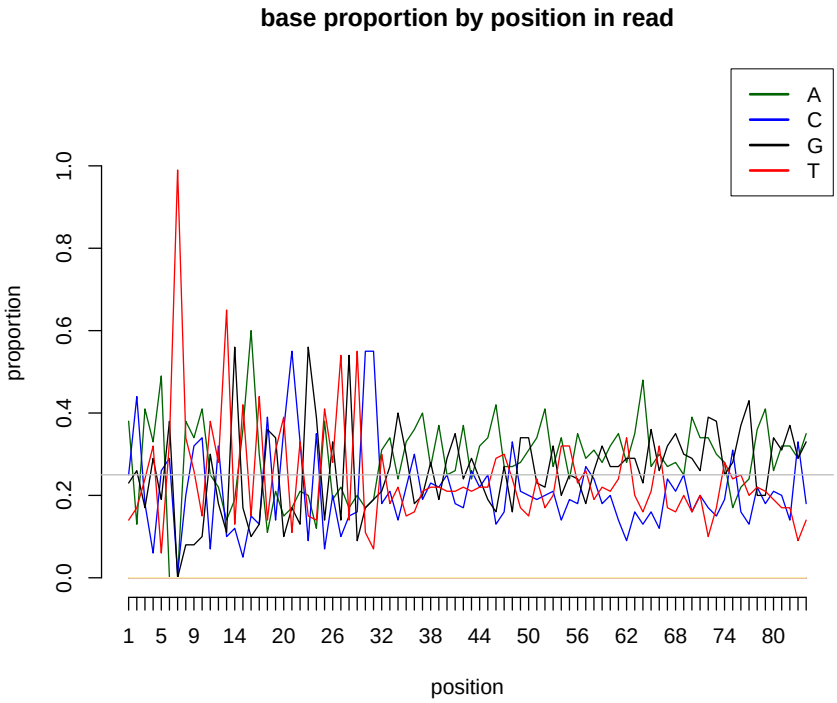

Figure 4: Base proportions by position in sequence.

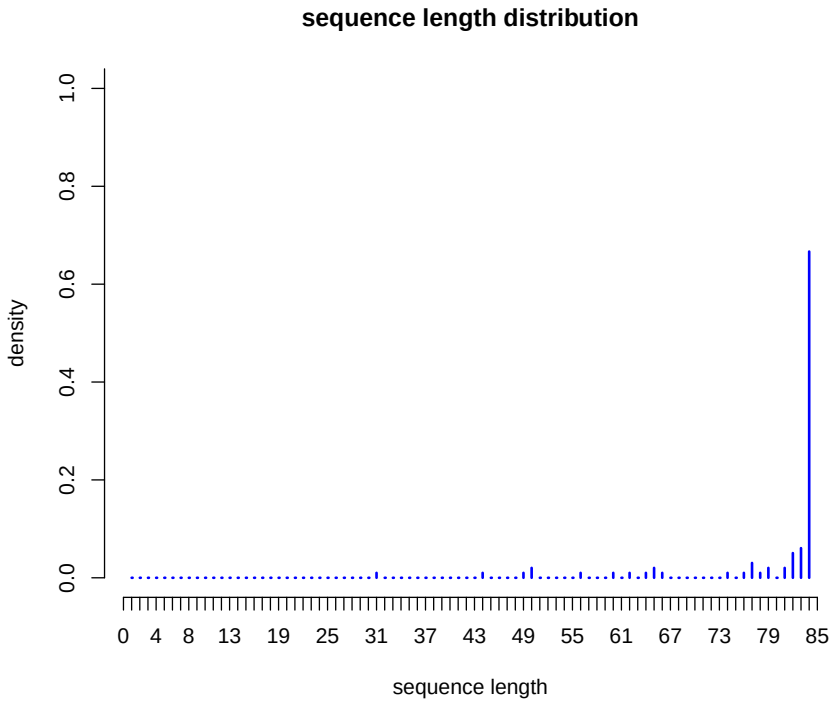

Figure 5: Histogram of sequence lengths after quality trimming.

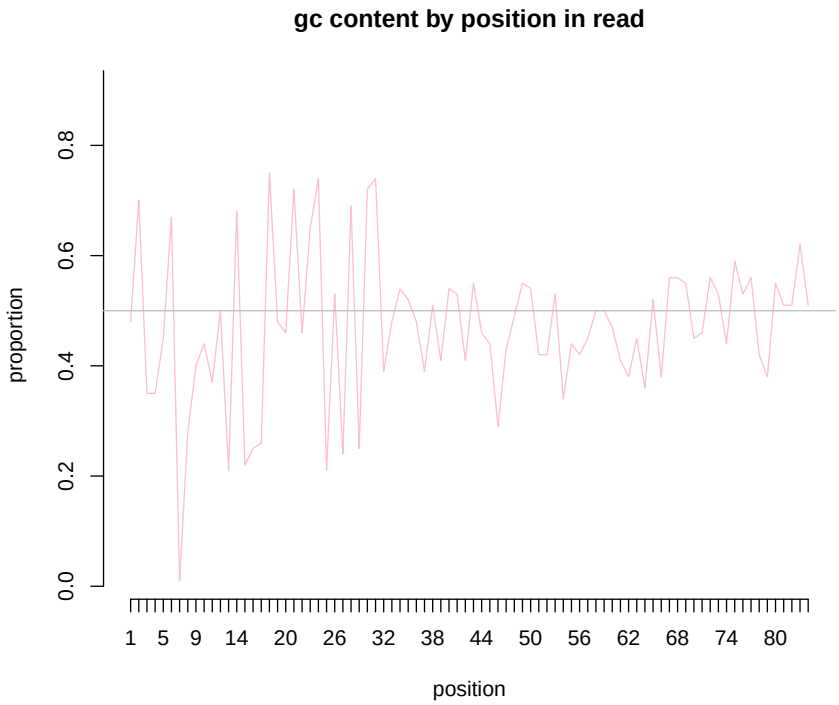

Figure 6: GC content by position

```
Called from: plotSeqLengths(s.trimmed.fastq)
```

The GC content can be plotted with `plotGC` (graphic shown in Figure 6):

```
> plotGC(s.fastq)
```

4 HTML Report Generation

With the help of *brew* and *xtable*, *qrqc* can generate an HTML summary report. This is created with `makeReport`. Reports are stored in their own directories, with images kept under 'images/' and the report is "report.html". A specific output directory can be passed through the `outputDir` argument, but the present directory is used as a default. Multiple reports generated in the same directory will have an incremental naming scheme.

5 Working with the FASTQSummary and FASTASummary classes

qrqc provides the `FASTQSummary` and `FASTASummary` classes for users to build functions and applications around. Both inherit from `SequenceSummary`, which should not be used directly.

`FASTASummary` has the same slots as `FASTQSummary`, except the latter provides additional slots for quality information. Both contain:

- `filename`: the filename of the file read and summarized with `readSeqFile`.
- `base.freqs`: a dataframe containing the frequency counts of each base.
- `seq.lengths`: a numeric vector containing the sequences lengths (counts by position).
- `hash`: a numeric vector containing the counts of unique sequences (the actual sequences are the names attribute).
- `hashed`: a logical indicating whether sequence hashing to count unique sequences was done.

Additionally, `FASTQSummary` provides:

- `qual.freqs`: a dataframe containing the counts of bases of a particular quality by position.
- `mean.qual`: a numeric giving the mean quality, weighted by sequence lengths.

6 Acknowledgements

Thanks to Simon Andrews for his work on FastQC (a similar program written in Java) from which this project was inspired. Also thanks to Joseph Fass and Dawei Lin for their advice and helpful comments.

Thanks to Heng Li for his work on `kseq.h` `khash.h` (both MIT License) which this package uses through *RSamtools*. More on these header files can be found at `http://lh3lh3.users.sourceforge.net/kseq.shtml` and `http://attractivechaos.awardspace.com/khash.h.html`.

`Sickle` can be downloaded or cloned from the UC Davis Bioinformatics Github repository: `https://github.com/ucdavis-bioinformatics/sickle`.

7 Session Info

```
> sessionInfo()

R version 2.13.0 beta (2011-03-29 r55167)
Platform: x86_64-apple-darwin10.6.0 (64-bit)

locale:
[1] C/en_US.UTF-8/C/C/en_US.UTF-8/en_US.UTF-8

attached base packages:
[1] tools     stats     graphics  grDevices utils     datasets  methods
[8] base

other attached packages:
[1] qrqc_0.99.81         Rsamtools_1.3.29     Biostrings_2.19.17
[4] GenomicRanges_1.3.29 IRanges_1.9.27

loaded via a namespace (and not attached):
[1] Biobase_2.11.10 brew_1.0-5      evaluate_0.3    plyr_1.4
[5] reshape_0.8.4   stringr_0.4     testthat_0.4    xtable_1.5-6
```