

An introduction to rSFFreader

Matt Settles*

April 11, 2014

Contents

1	Introduction	1
2	A simple workflow	1
<code>> library("rSFFreader")</code>		
<code>> library("xtable")</code>		

1 Introduction

The SFF file format has been adopted by both Roche 454 and Ion Torrent next generation sequencing platforms. *rSFFreader* provides functionality for loading sequence, quality scores, and flowgram information from these files. This package has been modeled after the excellent (ShortRead) package released by Martin Morgan. It aims to maintain compatibility with that package while enabling direct processing of SFF files.

2 A simple workflow

Read in an SFF file:

```
> sff <- readSff(system.file("extdata", "Small454Test.sff", package="rSFFreader"))
```

```
Total number of reads to be read: 1000
```

```
reading header for sff file:/private/tmp/RtmpTFSC6V/Rinstfffc587c0b03/rSFFreader/extdata/Small454Test.sff
```

```
reading file:/private/tmp/RtmpTFSC6V/Rinstfffc587c0b03/rSFFreader/extdata/Small454Test.sff
```

```
called SffReads validitycalled SffReadsQ validity
```

Accessing the read, quality, and header information:

```
> sread(sff)
```

*msettles@uidaho.edu

```

A DNAStringSet instance of length 1000
      width seq                      names
[1]   422 ACACGACGACTT...GGCGCTCGCTC HRWLTHE02G15D7
[2]   157 ACACTACTCGTG...CTGGTGCCGGC HRWLTHE02H2PCX
[3]   376 ACACGACGACTG...CGTTACAAATC HRWLTHE02HB23L
[4]   243 ACACGACGACTC...GAGAAGATCAT HRWLTHE02IYLA2
[5]   727 ACACTACTCGTG...GGTCTCCGTTA HRWLTHE02F3E10
...   ...
[996] 652 ACACGACGACTC...CGCCTTCCTGC HRWLTHE02JSWSM
[997] 756 ACACGACGACTG...CCCGGTCACCG HRWLTHE02FJUSH
[998] 574 ACACGACGACTT...ACGAGGGGGGT HRWLTHE02GCJZT
[999] 693 ACACTACTCGTC...TACCGGCAGCA HRWLTHE02IFUFC
[1000] 573 ACACTACTCGTC...TTGTGAATACG HRWLTHE02GF2BA

> quality(sff)

class: FastqQuality
quality:
  A BStringSet instance of length 1000
        width seq
[1]   422 IIIIIIIIIHHHFFFFFFF>...55577;;997977777878787
[2]   157 IIHHIHAAADFHDAAACAA>A>@...>???FFD@?B@?=?;4/...7
[3]   376 >>FHHHHFFFFFFDCA@=?=77...777778<;;0002999:74444
[4]   243 IIIIIIIIIIEEFHHFHHHFII...887==<;;87555641.144/
[5]   727 IIIIIIIIIHHIIIIIFHHHHH...22--/,/+++-----+++-/
...   ...
[996] 652 IIIIIIIIIHHHGGDD>>>FII...0225231111.10///,,/,
[997] 756 IIIIIIIIIIIIIHHHHIII...100---))++001001*10-+
[998] 574 IIIIIIIHHHFFFA<444>EC...+++) ,.111111.,+++++
[999] 693 IIIIIIIIIIIIIIIIIII...01111101010-----,/+-
[1000] 573 IIIIIIIIIIIIIIIIIII...33355322500--)+.10-+++

> header(sff)

[[1]]
[[1]]$filename
[1] "/private/tmp/RtmpTFSC6V/Rinstfffc587c0b03/rSFFreader/extdata/Small454Test.sff"

[[1]]$magic_number
[1] 779314790

[[1]]$version
[1] ""

[[1]]$index_offset
[1] 6201592

```

[1] 20728

```
[1] 1000
```

[1] 1640

[1] 4

```
[1] 1600
```

[1] 1

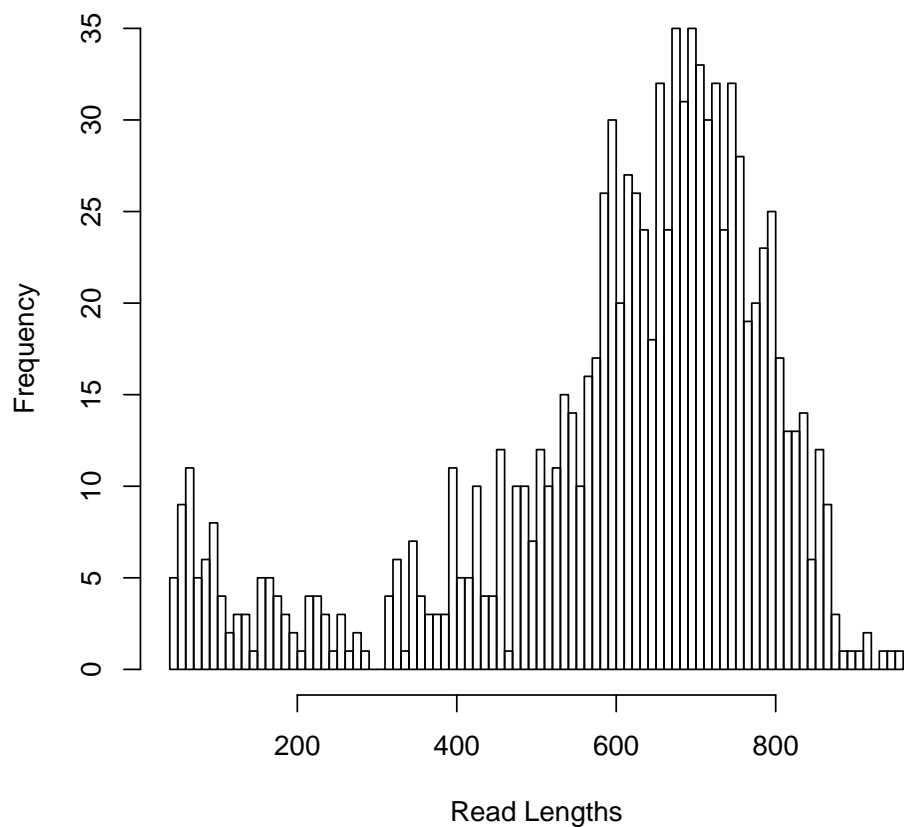
[illegible]

[1] "GACT"

Plot histogram of read lengths:

```
+ breaks=100)
```

Histogram of read lengths using full clip mode.



Setting the `clipMode` will change the read lengths that are reported by `width` and plotted by `hist`. Currently the following modes are supported:

- `adapter` : defined in the SFF file, and meant to remove adapter sequence
- `quality` : defined in the SFF file, and meant to remove low-quality regions of the sequence
- `full` : uses the "interior" of quality and adapter and is the most conservative
- `raw` : no clipping is applied and full length reads are returned
- `custom` : clip points set by the user as an *IRanges* object.

```
> availableClipModes(sff)
```

```
[1] "full"      "quality" "adapter" "raw"
```

```

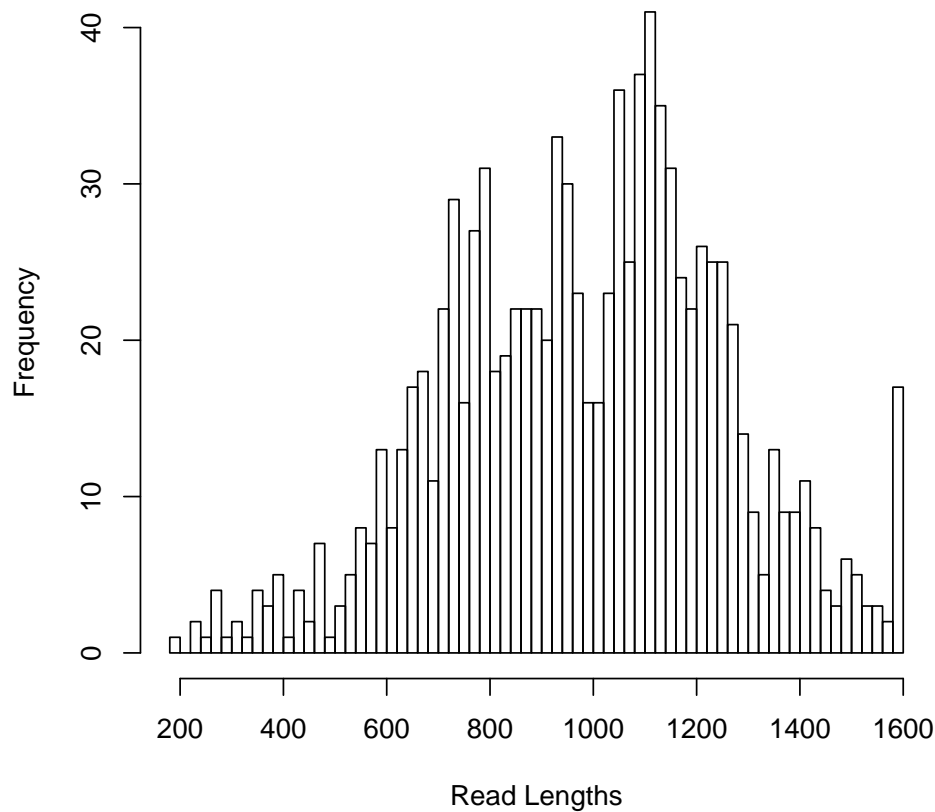
> clipMode(sff)

[1] "full"

> clipMode(sff) <- "raw"
> hist(width(sff), xlab="Read Lengths",
+       main=paste("Histogram of read lengths using", clipMode(sff), "clip mode."),
+       breaks=100)

```

Histogram of read lengths using raw clip mode.



Custom clip points can be set using *IRanges*. For example, it is sometimes useful to look for barcodes (MID tags) in the first 15 bases of a set of reads.

```

> customClip(sff) <- IRanges(start = 1, end = 15)
> clipMode(sff) <- "custom"
> t = table(counts=as.character(sread(sff)))

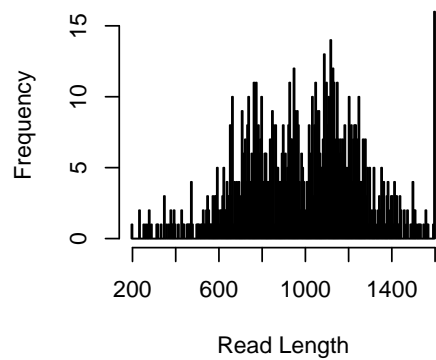
```

	counts
GACTACACGACGACT	284
GACTACACGTAGTAT	377
GACTACACTACTCGT	316

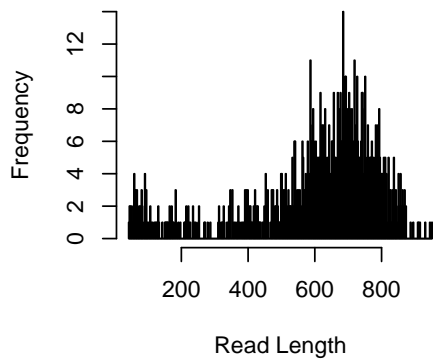
Finally, we can generate some useful QA plots and

```
> ## Generate some QA plots:
> ## Read length histograms:
> par(mfrow=c(2,2))
> clipMode(sff) <- "raw"
> hist(width(sff),breaks=500,col="grey",xlab="Read Length",main="Raw Read Length")
> clipMode(sff) <- "full"
> hist(width(sff),breaks=500,col="grey",xlab="Read Length",main="Clipped Read Length")
> ## Base by position plots:
> clipMode(sff) <- "raw"
> ac <- alphabetByCycle(sread(sff),alphabet=c("A","C","T","G","N"))
> ac.reads <- apply(ac,2,sum)
> acf <- sweep(ac,MARGIN=2,FUN="/",STATS=apply(ac,2,sum))
> matplot(cbind(t(acf),ac.reads/ac.reads[1]),col=c("green","blue","black","red","darkgrey","purple"),
+         type="l",lty=1,xlab="Base Position",ylab="Base Frequency",
+         main="Base by position")
> cols <- c("green","blue","black","red","darkgrey","purple")
> leg <- c("A","C","T","G","N","% reads")
> legend("topright", col=cols, legend=leg, pch=18, cex=.8)
> clipMode(sff) <- "full"
> ac <- alphabetByCycle(sread(sff),alphabet=c("A","C","T","G","N"))
> ac.reads <- apply(ac,2,sum)
> acf <- sweep(ac,MARGIN=2,FUN="/",STATS=apply(ac,2,sum))
> matplot(cbind(t(acf),ac.reads/ac.reads[1]),col=c("green","blue","black","red","darkgrey","purple"),
+         type="l",lty=1,xlab="Base Position",ylab="Base Frequency",
+         main="Base by position")
> legend("topright", col=cols, legend=leg, pch=18, cex=.8)
>
```

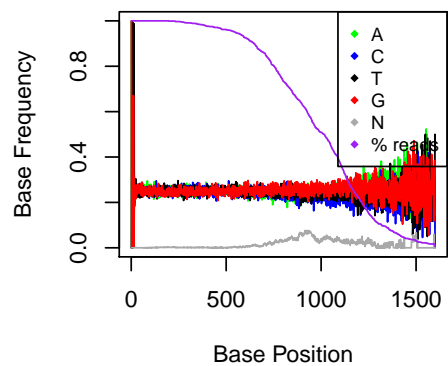
Raw Read Length



Clipped Read Length



Base by position



Base by position

