RNA Sequencing & Expression Analysis

Alastair Droop

LIMM Methods Club, 27/02/2018

preprocessing

alignment

read counting

expression analysis

differential exon usage (rMATS)

other analyses

Part 1: Sequencing

Work out the sequence of all the RNA molecules in a sample

Three phases:

1. *Create a suitable library*;
2. *Sequence the library;* and
3. *Analyse the results from the sequencing*

Library Preparation

Create a suitable library for running the sequencer

Main issues:

- Ensure your sequences have the correct adapters for the sequencer to work;
- Ensure that only sequences you want to study are present; and
- Make sure you have enough sequence to use

Contamination is bad because

- Waste spots that could be used for real sequences; and
- Adds incorrect data to the sample

Sequencing

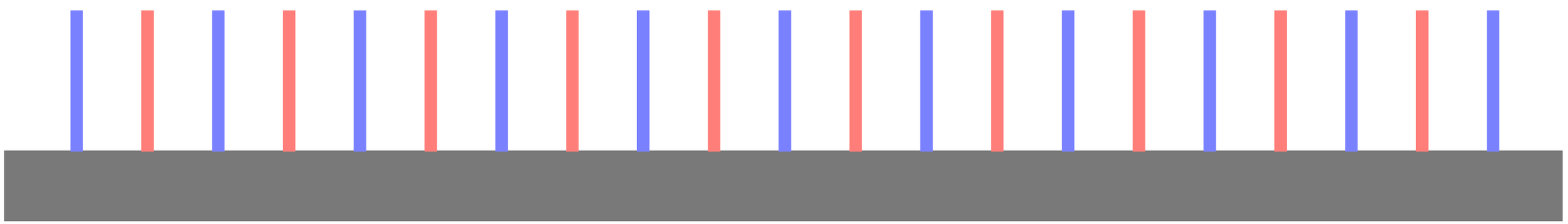
DNA spots are created from a library of DNA fragments

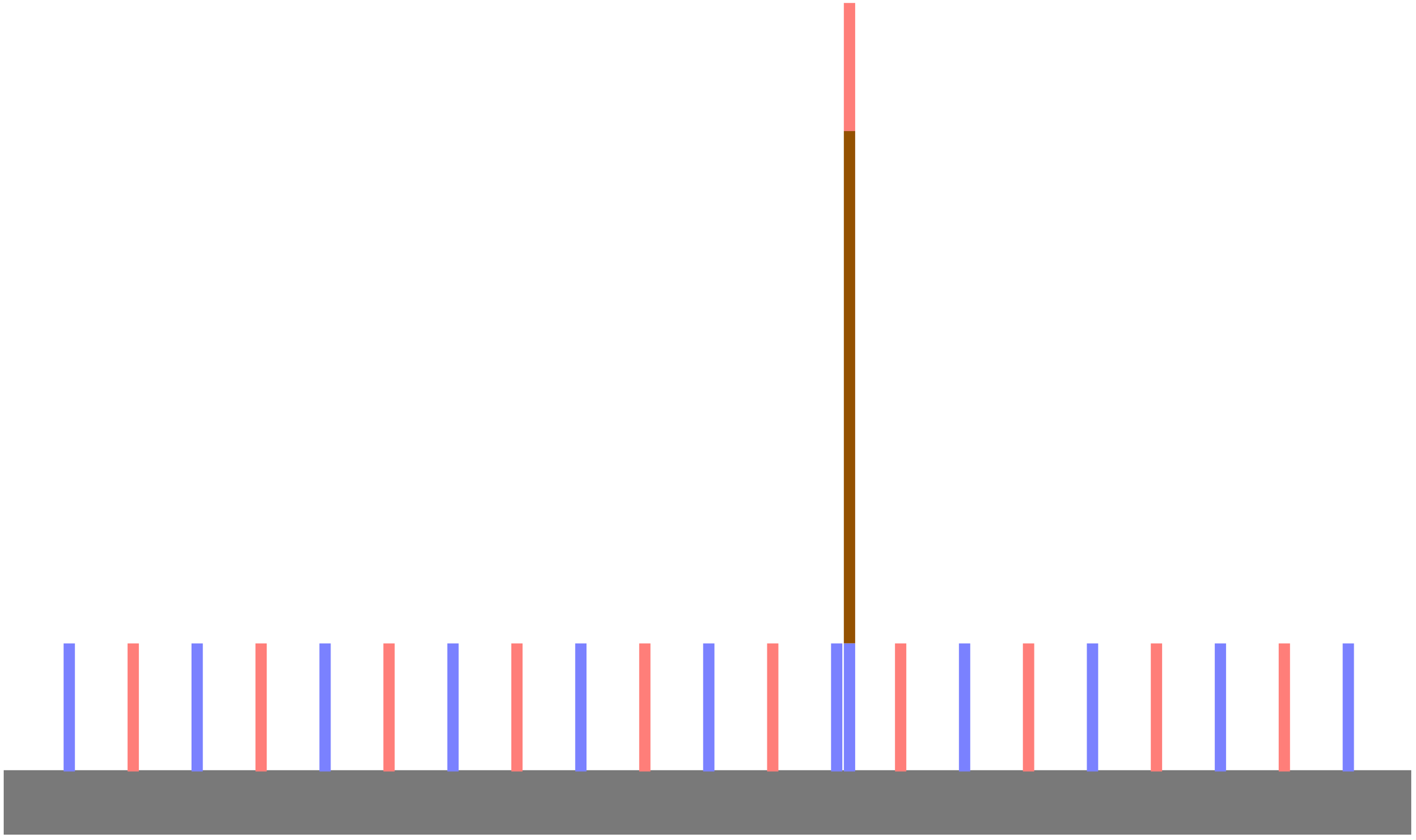
Each spot is made from a single DNA species from the prepared library

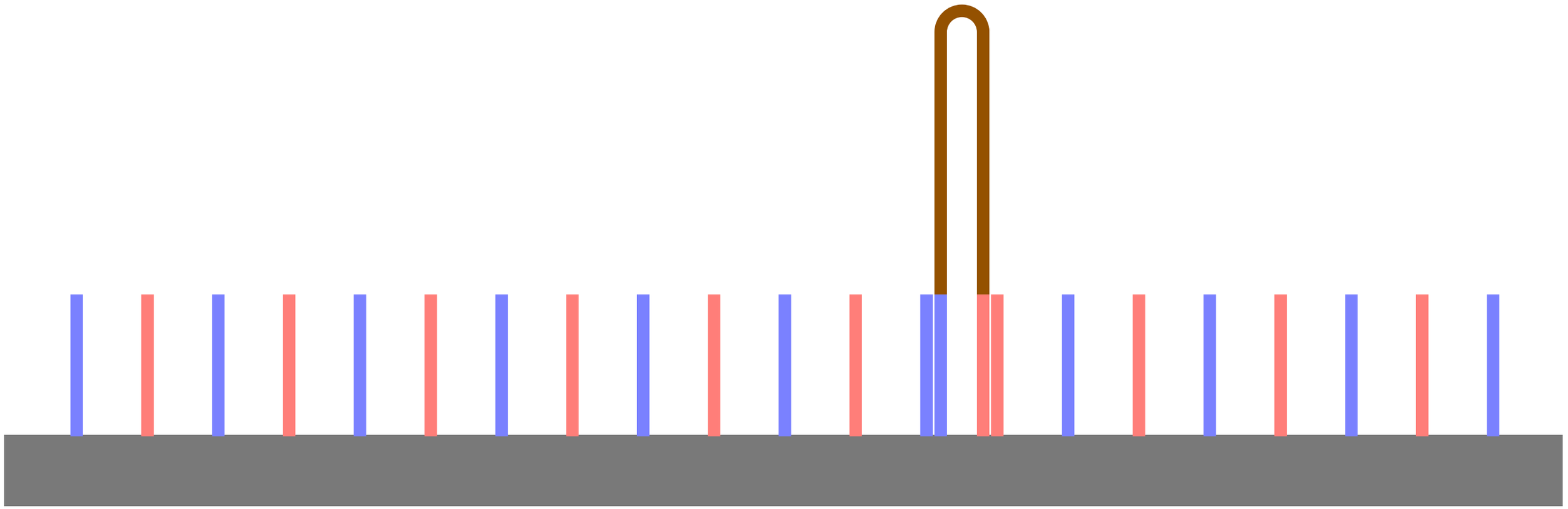
- Affixed to a glass plate (the “flow cell”)
- A single DNA molecule is ‘grown’ into a spot by bridge PCR
- Spots ~2μm diameter
- ~150×10^6 individual spots per lane of the flow cell (8 lanes)

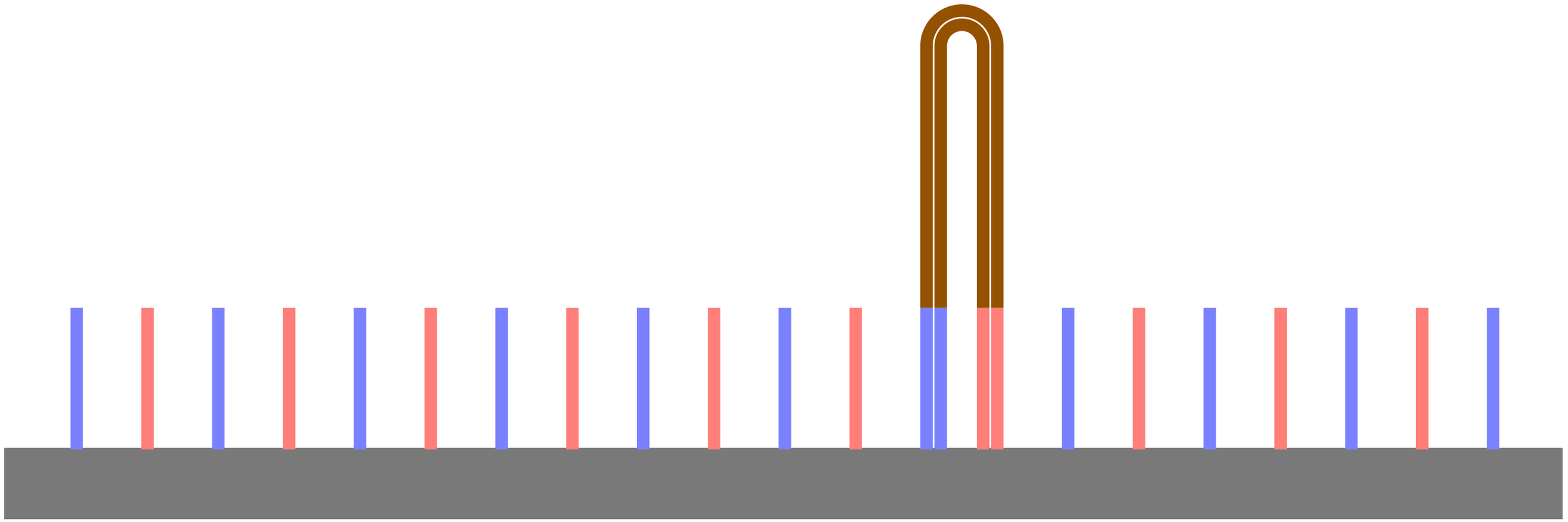
Sequence is read by Sanger sequencing

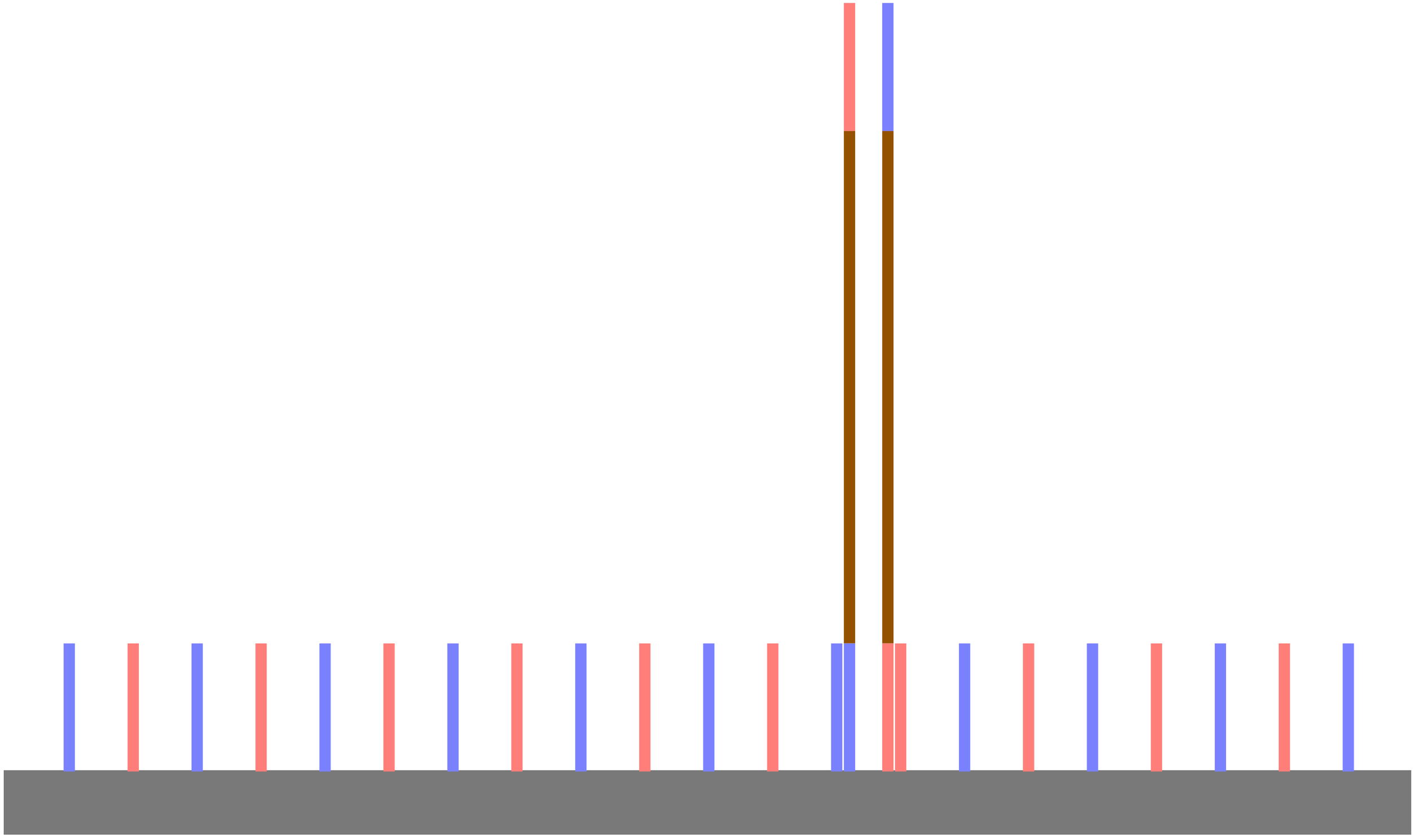
- Reversible dye-terminator technology

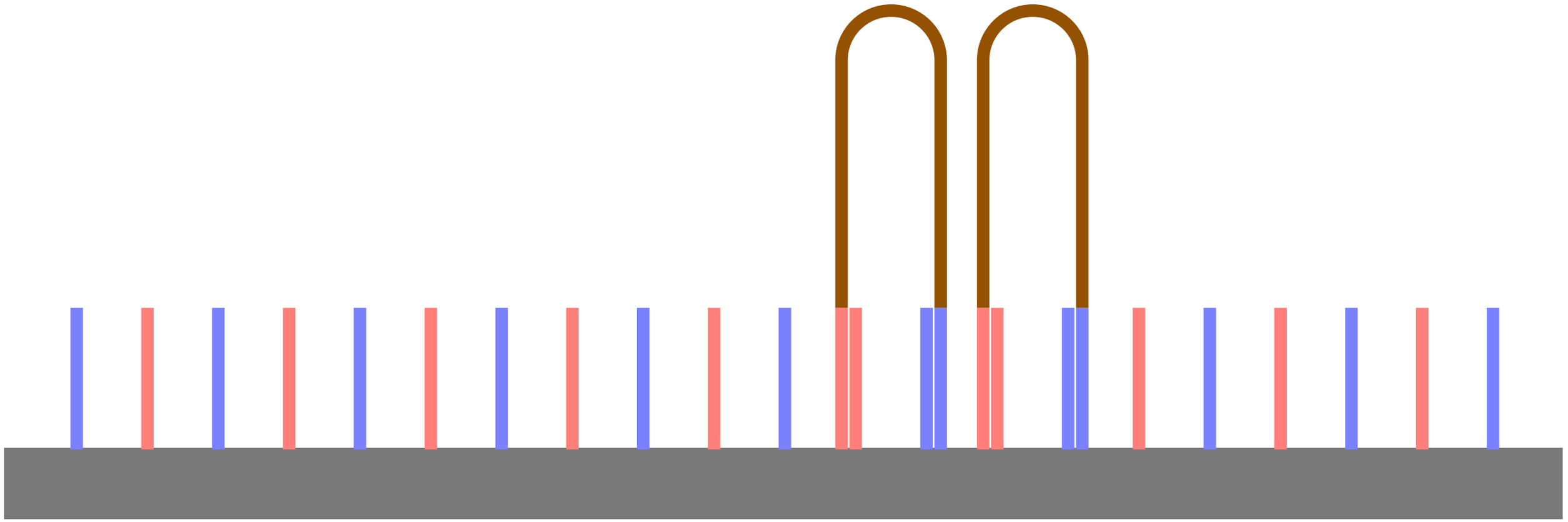


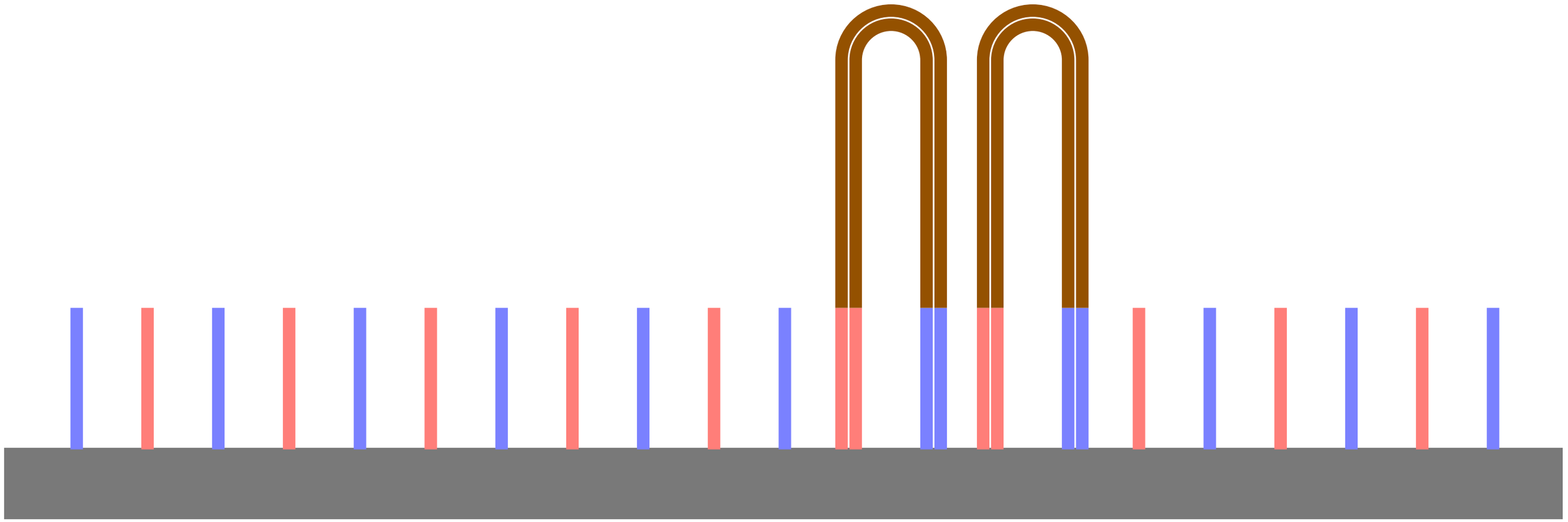


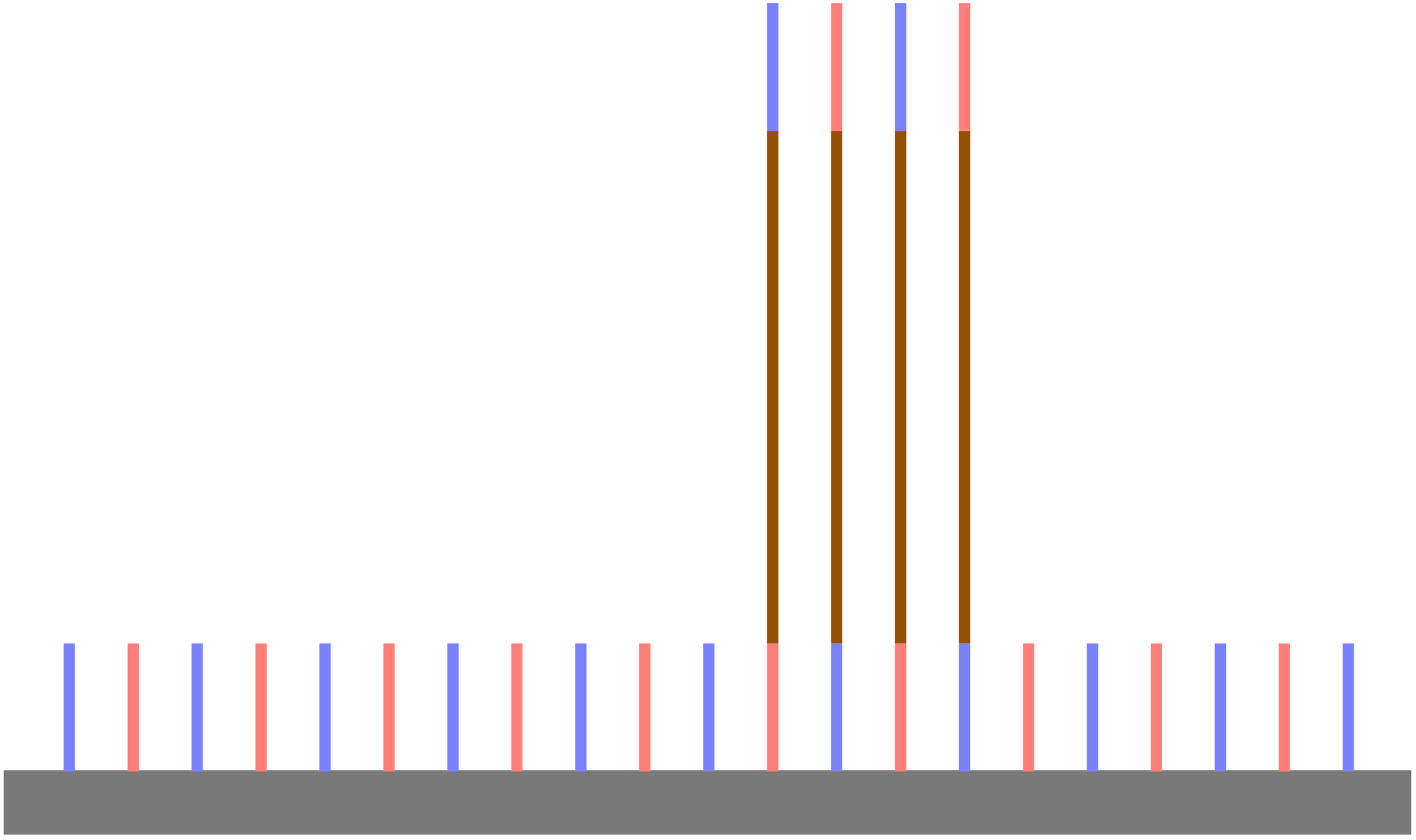










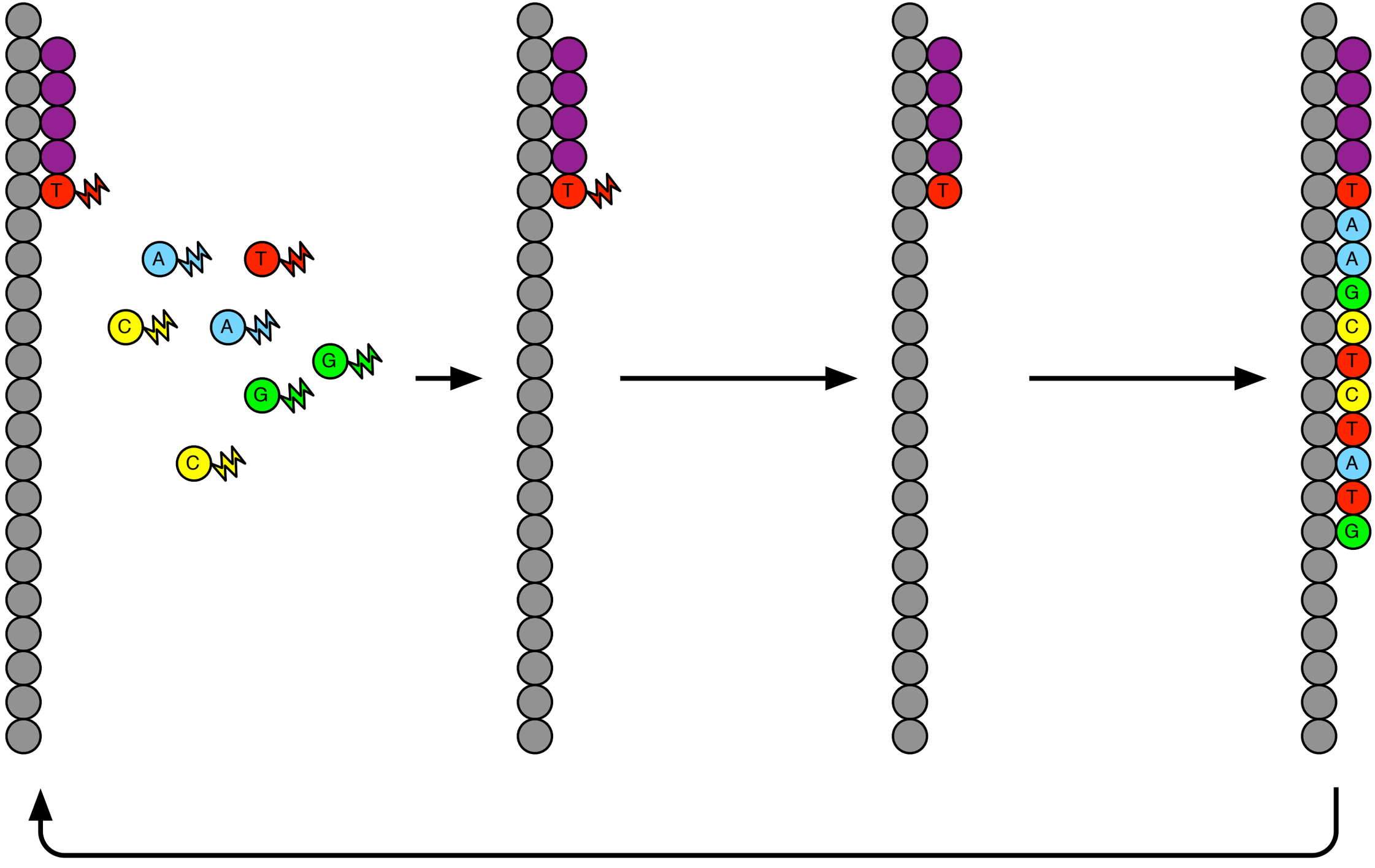


Incorporate single labelled nucleotide

Measure fluorescence from each spot

Cleave fluorescent dye & terminator

Repeat



Each spot is measured at all 4 wavelengths each cycle

- Ratio of wavelengths determines incorporated base (or can't call)

We get back ~150×10^6 individual reads

- Each one 100 bases
- Each read has both the read sequence (A, T, G, C, N) & quality score

'Paired end' sequencing

- Essentially reads both ends of the sequence
- Twice the data!

FASTQ Data

For each read, we get a sequence and a quality:

header `@D9414NS1:262:C5WBPACXX:2:1101:6808:2215 1:N:0:CCGTCC`

sequence `GGAGGCTGGAGGCTCAAGTCCAAGGCATTGCAGAAGACAGTTCCTGCTCTGGTT`

`+`

quality `CCCFFFFFGGHHHIJIBHAGHEGHIEGGIIIDIAHGIIIG:BAGGGHIHIG>BC`

data for cycle 8

FASTQ Quality Values

For each spot, each cycle outputs a base call and the quality score

The quality score reports how confident the sequencer is about the call

Quality scores are PHRED-encoded:

$$Q = -10 \times \log_{10} p$$

The probability value is encoded as a single character (Q+33)

So, the value F (chr 70) = 0.0001995262… = 0.0002

This corresponds to a probability that the call is wrong of 1 in 5000

We need to:

- Check for bad quality data (FastQC)
- Remove non-target DNA sequence (cutadapt)

We run FastQC both before and after adapter removal

- Several metrics, including GC content, adapter contamination, quality, *etc*…

~10–30% of all bases will be adapter

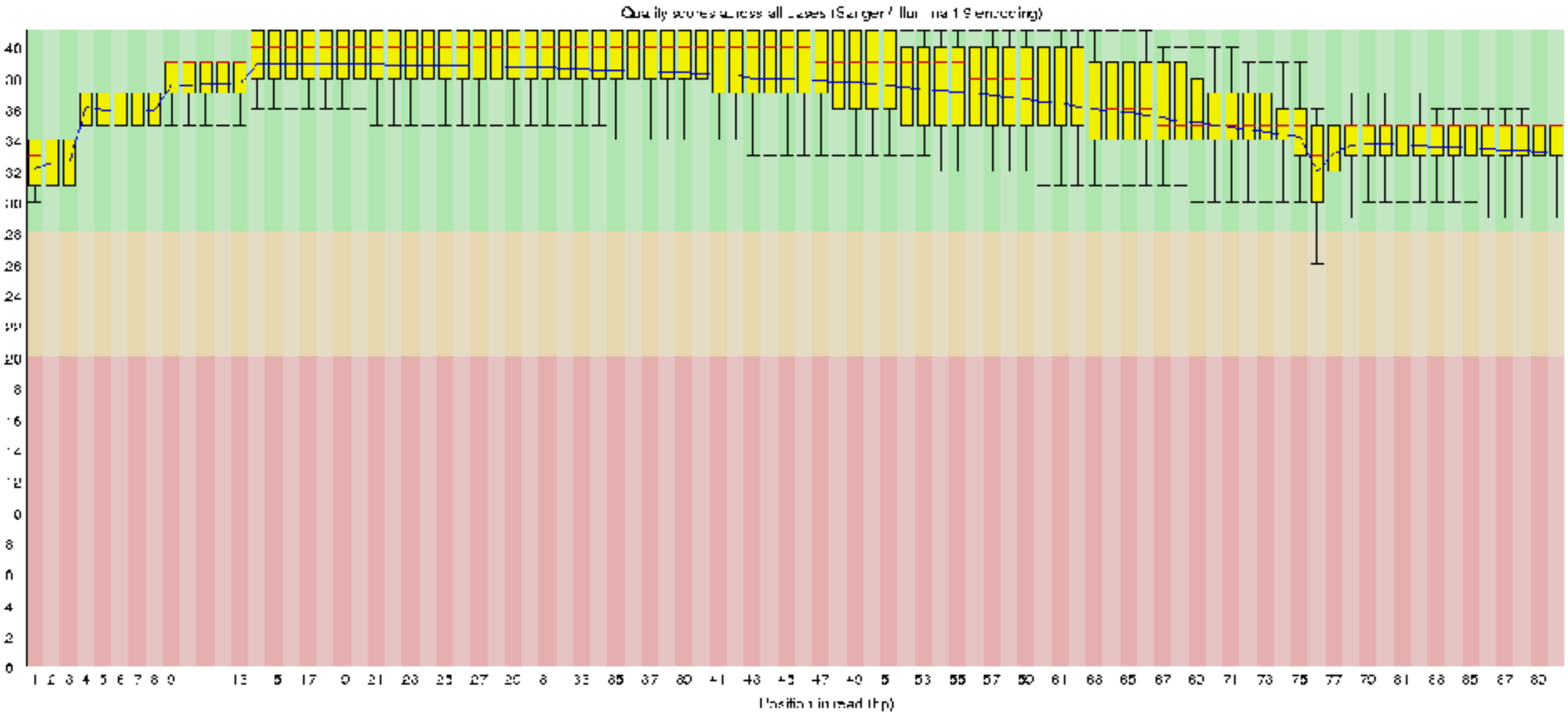
FastQC

http://www.bioinformatics.babraham.ac.uk/projects/fastqc/

Process each FASTQ file looking for anomalies

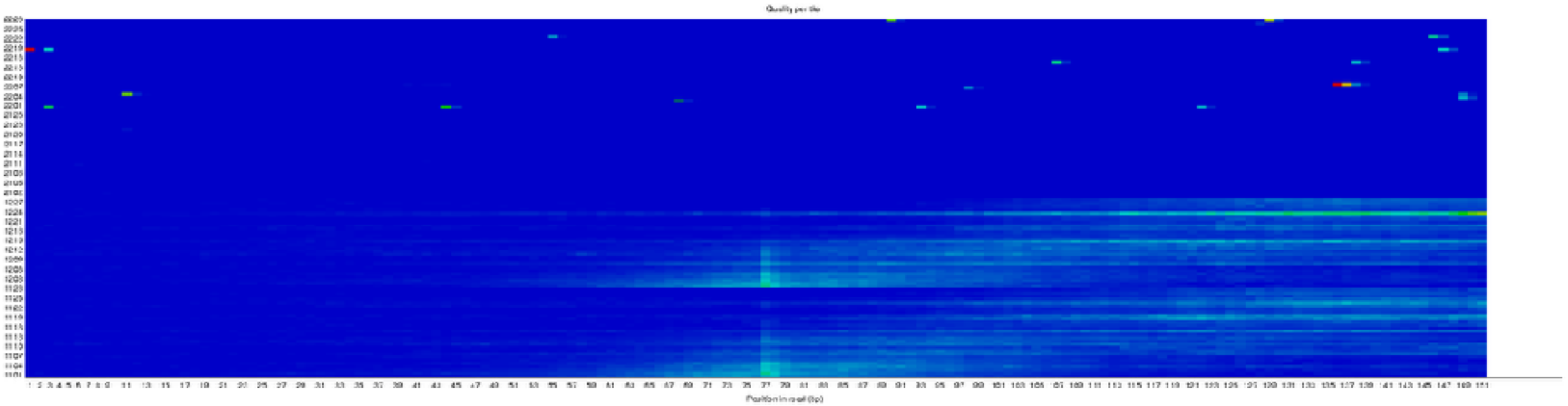
- Compare to ideal; often not possible to get all modules passing
- Run FastQC before and after preprocessing
- Look for *differences* between your samples, rather than simply pass / fail

FastQC: Per base sequence quality



Looking for high-quality; all boxes ≥ ~28

FastQC: Per tile sequence quality

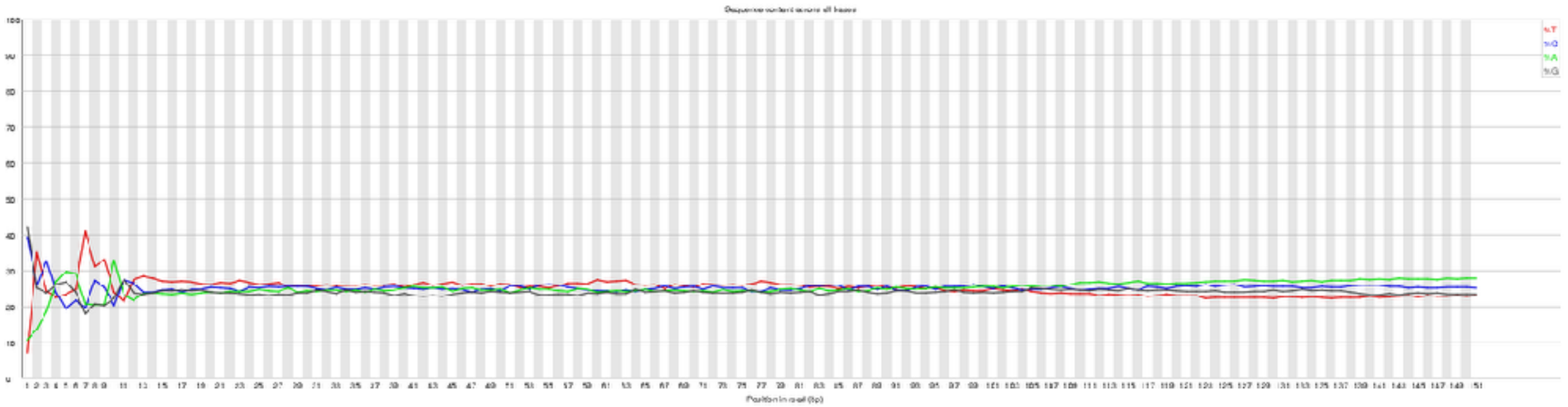


Only available if FASTQ headers are in Illumina format

Shows average quality scores per tile on the flowcell

Transient issues OK; more worrying artefacts are issues with the run

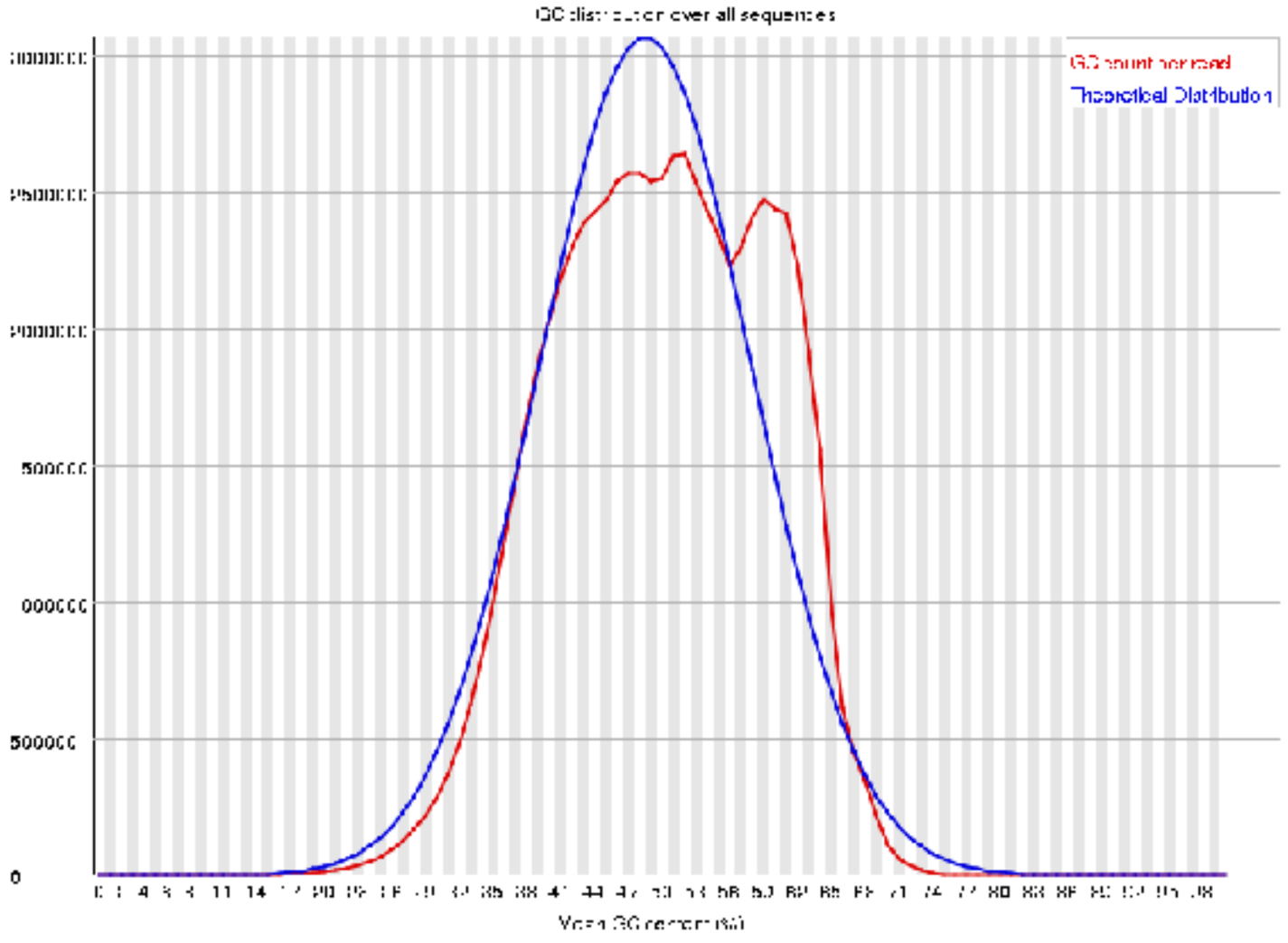
FastQC: Per base sequence content



Looking for flat lines; base content approximately as expected

Frequently, first few bases show bias due to library preparation method

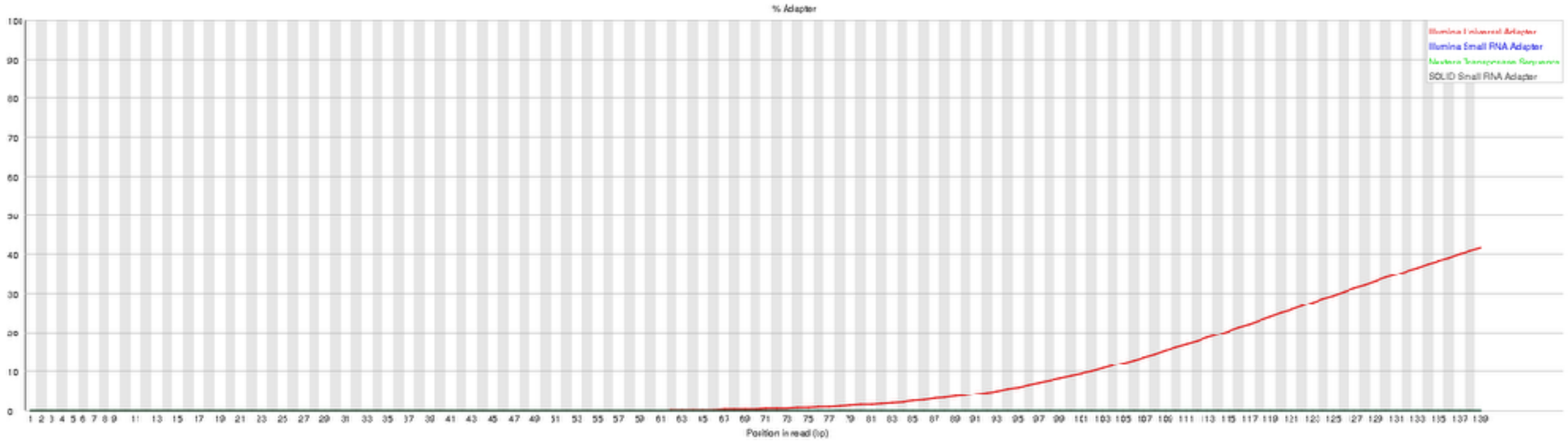
FastQC: Per sequence GC content



Shows distribution of GC content across all reads

Sharp spikes indicate possible overrepresented sequences

FastQC: Adapter content



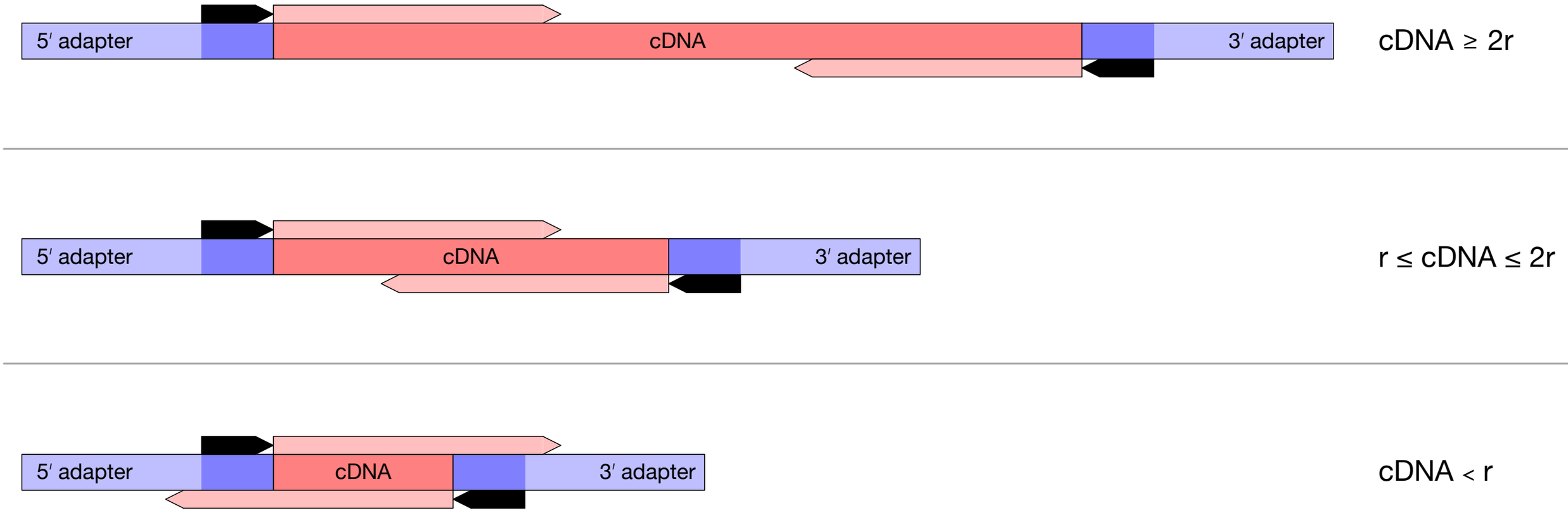
Shows adapter contamination

After QC, there should be none

Cutadapt

Process each FASTQ file pair, cleaning reads

- Remove low-quality data (technical artefacts of the sequencer);
- Remove high-quality data that aren't part of the biological sample; and
- Remove reads that are too short



QC & Pre-Alignment Cleaning

Multiple possible options, but mainly

- Clipping low-quality read ends
- Identify adapters

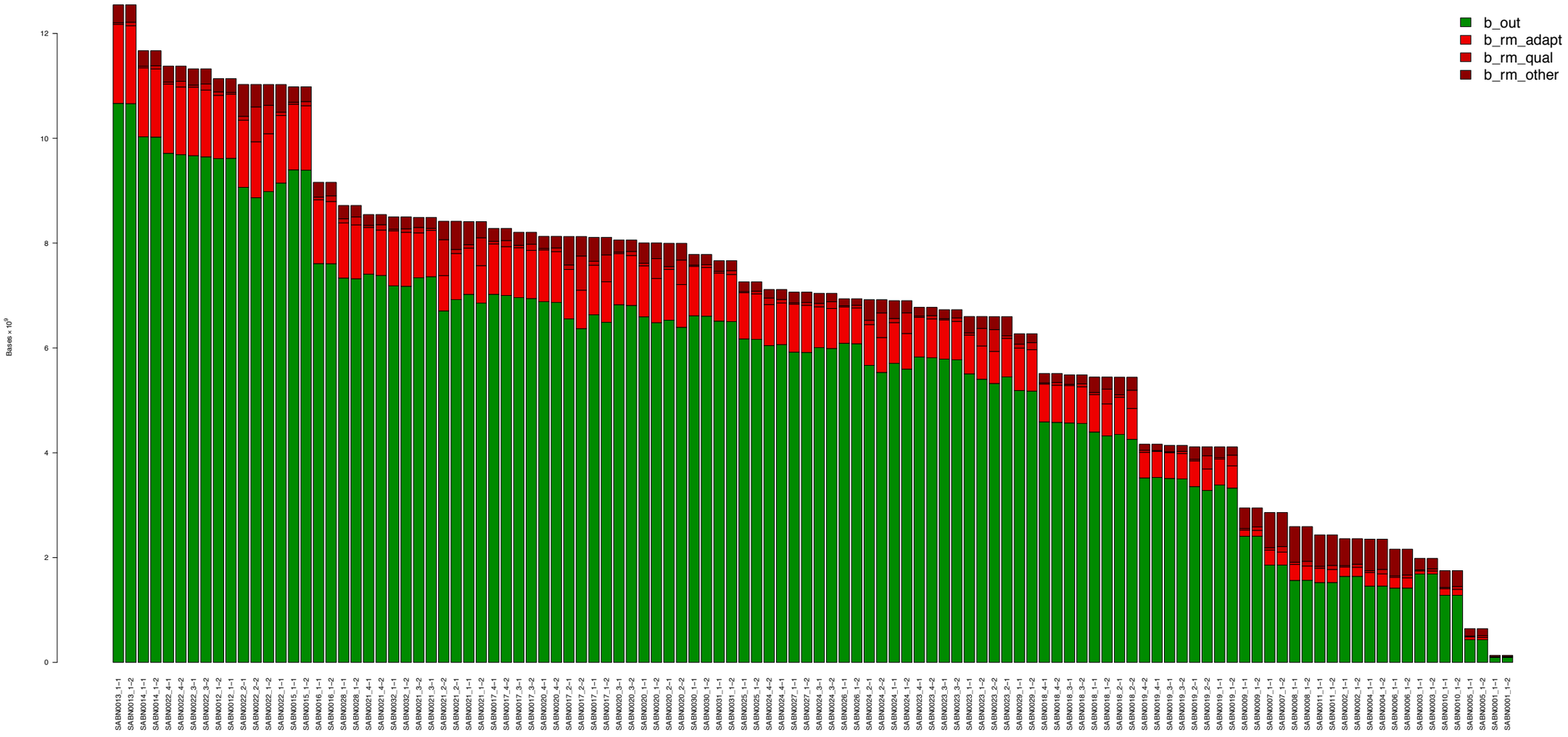
Use results of first-pass FastQC if you don't know what adapters to expect

Standard Illumina TruSeq:

- pair 1: `AGATCGGAAGAGCACACGTCTGAACTCCAGTCAC`
- pair 2: `AGATCGGAAGAGCGTCGTGTAGGGAAAGAGTGTA`

Look for differences in behaviour between samples, rather than simple percentages

Use `cutadapt-stats` to pull out overall statistics
`https://github.com/alastair-droop/cutadapt-stats`



Alignment

A complex process. Some thoughts:

- Use a modern reference unless you have good reason not to
- Use an appropriate aligner (bwa, or RNA-STAR)
- Use (but *understand*) published / accepted settings (tinkering usually hurts)

Cleaning your aligned BAM files is an art

- Remove duplicates with care when performing RNAseq
- Remove low-quality & misaligned reads
- Index & sort

Check BAM file stats (`samtools idxstats` & `flagstat`)

Part 2: Differential Expression Analysis

Read Counting

If you've used RNA-STAR, the easiest way is to use featureCounts:

- Uses the GTF file you used to generate the reference
- Give it all your BAM files, and it will make a single matrix of counts
- Usually per-gene counts found by counting individual exons

`http://subread.sourceforge.net`

DESeq2

https://bioconductor.org/packages/release/bioc/vignettes/DESeq2/inst/doc/DESeq2.html

Love *et al. Genome Biology* (2014) 15:550
DOI 10.1186/s13059-014-0550-8

Genome Biology

METHOD Open Access

Moderated estimation of fold change and dispersion for RNA-seq data with DESeq2

Michael I Love[1,2,3], Wolfgang Huber[2] and Simon Anders[2*]

DESeq2 runs in R, so you will need:

- A basic understanding of R (quite hard);
- A knowledge of the statistics you are going to use (much harder); and
- Your data

Critical to success is knowing which samples you want to compare

- "Simple" analyses are very easy:

```
dds <- DESeqDataSetFromMatrix(countData = M, colData = coldata, design= ~condition)
dds <- DESeq(dds)
res <- lfcShrink(dds)
```

- The key is to *sort out your data and metadata correctly*.