viRome: an R package for the visualization and analysis of viral small RNA sequence datasets

Mick Watson

November 12, 2015

We have developed code in R to analyse short-read next-generation sequencing data from virus-infection studies where the siRNA or piRNA pathways have been implicated. The package provides a range of functions to help scientists visualise the results of such experiments.

viRome takes aligned data (in the form of a sorted, indexed BAM file) and produces a variety of graphs and reports.

viRome is completely open source and is available on Sourceforge: `http://sourceforge.net/projects/virome`

1 Computing requirements

The role of viRome is to summarise and plot millions of data points - you will therefore likely need a powerful computer. The package should work on standard desktops and laptops, but may take longer than expected. viRome was developed on machines with large amounts of RAM and significant processing power.

The tutorial below takes approximately 30 minutes on a 64 bit Windows PC with 24Gb of RAM and a dual-core 2.4GHz processor. If you find that the tutorial takes significantly longer, then please use a PC with more RAM and/or a faster processor.

2 Download and install R and necessary packages

You will need R: navigate to `http://www.r-project.org` and download and install the latest version of R for your operating system

The next step is to ensure we have all the dependencies installed. You only need to run this once:

```
> # get the necessary Bioconductor packages
> # source("http://www.bioconductor.org/biocLite.R")
> # biocLite("Rsamtools")
```

```
>
> # install optional Bioconductor packages
> # biocLite("seqLogo")
> # biocLite("motifStack")
>
> # install the necessary R packages
> # install.packages(c("plyr","gsubfn","seqinr","reshape2"))
>
> # install optional R packages
> # install.packages("ggplot2")
```

All of the above options are commented out; please uncomment and run the commands above as needed, depending on your existing environment i.e. if you already have Rsamtools installed, there is no need to install it again.

3 Install the viRome package

Next install and load the viRome package from: `http://sourceforge.net/projects/virome`

- On windows download the .zip file and install via the menu option Packages -> Install packages from local zip files
- For linux and Mac, download the .tar.gz file from http://sourceforge.net/projects/virome. Install via R CMD INSTALL

4 Example data

The data we are using is from:

- Vodovar N, Bronkhorst AW, van Cleef KW, Miesen P, Blanc H, van Rij RP, Saleh MC. (2012) Arbovirus-derived piRNAs exhibit a ping-pong signature in mosquito cells. PLoS One. 7(1):e30861. doi: 10.1371/journal.pone.0030861.

The data are in the SRA here: `http://www.ncbi.nlm.nih.gov/sra?term=SRR389184`

We have distributed aligned data from this study with the viRome package - the data were aligned to the Sindbis virus genome using Novoalign (`http://www.novocraft.com`). We recommend that you use BAM files that are sorted according to genomic location, and indexed. This can be carried out using samtools (`http://samtools.sourceforge.net/`)

5 Minimal commands

Each function in viRome has a number of parameters, and all of these paramters have default values. In order to read about these parameters, and to see their default values, please access the help for wach function:

```
> ?read.bam
```

```
No documentation for âĂŸread.bamâĂŹ in specified packages and libraries:
you could try âĂŸ??read.bamâĂŹ
```

```
> ?clip.bam
```

```
No documentation for âĂŸclip.bamâĂŹ in specified packages and libraries:
you could try âĂŸ??clip.bamâĂŹ
```

```
> ?barplot.bam
```

```
No documentation for âĂŸbarplot.bamâĂŹ in specified packages and libraries:
you could try âĂŸ??barplot.bamâĂŹ
```

```
> ?size.strand.bias.plot
```

```
No documentation for âĂŸsize.strand.bias.plotâĂŹ in specified packages and libraries:
you could try âĂŸ??size.strand.bias.plotâĂŹ
```

```
> ?summarise.by.length
```

```
No documentation for âĂŸsummarise.by.lengthâĂŹ in specified packages and libraries:
you could try âĂŸ??summarise.by.lengthâĂŹ
```

```
> ?size.position.heatmap
```

```
No documentation for âĂŸsize.position.heatmapâĂŹ in specified packages and libraries:
you could try âĂŸ??size.position.heatmapâĂŹ
```

```
> ?stacked.barplot
```

```
No documentation for âĂŸstacked.barplotâĂŹ in specified packages and libraries:
you could try âĂŸ??stacked.barplotâĂŹ
```

```
> ?position.barplot
```

```
No documentation for âĂŸposition.barplotâĂŹ in specified packages and libraries:
you could try âĂŸ??position.barplotâĂŹ
```

```
> ?sequence.report
```

```
No documentation for âĂŸsequence.reportâĂŹ in specified packages and libraries:
you could try âĂŸ??sequence.reportâĂŹ
```

```
> ?make.pwm

No documentation for âĂŸmake.pwmâĂŹ in specified packages and libraries:
you could try âĂŸ??make.pwmâĂŹ

> ?pwm.heatmap

No documentation for âĂŸpwm.heatmapâĂŹ in specified packages and libraries:
you could try âĂŸ??pwm.heatmapâĂŹ

> ?read.dist.plot

No documentation for âĂŸread.dist.plotâĂŹ in specified packages and libraries:
you could try âĂŸ??read.dist.plotâĂŹ
```

However, as we have already set the defaults, there are minimal commands which will create plots based on those defaults. The mimimum information required is the data that the command must read to summarise and create a plot, and this is typically given as the first argument to the function. Examples of these minimal commands, and the type of data they reuire, is below:

```
> # load the library
> # find example data
> library(viRome)
> infile <- system.file("data/SRR389184_vs_SINV_sorted.bam", package="viRome")
> # minimal commands
>
> # requires the full path to a bam file,
> # and the name of the reference the data are aligned to
> bam    <- read.bam(infile, chr="SINV")
> # requires only the output of read.bam()
> bamc   <- clip.bam(bam)
> # requires only the output of clip.bam()
> bpl    <- barplot.bam(bamc)
> # requires only the output of barplot.bam()
> ssp    <- size.strand.bias.plot(bpl)
> # requires only the output of clip.bam()
> dm     <- summarise.by.length(bamc)
> # requires only the output of summarise.by.length()
> sph    <- size.position.heatmap(dm)
> # requires only the output of summarise.by.length()
> sbp    <- stacked.barplot(dm)

[1] "WARNING: you have chosen to plot a lot of graphs to the current device"
[1] "WARNING: this may be too many for the device to cope with"
[1] "WARNING: if you get an error message, then consider using a bigger device"
[1] "WARNING: with bigger dimensions: see ?png ?jpeg ?pdf"
```

```
> # requires only the output of clip.bam()
> # though one should alter minlen, maxlen
> # and reflen
> sir    <- position.barplot(bamc)
> # requires only the output of clip.bam()
> sr     <- sequence.report(bamc)
> # requires only the output of clip.bam()
> pwm    <- make.pwm(bamc)
> # requires only the output of make.pwm()
> pmh    <- pwm.heatmap(pwm)
> # requires only the output of sequence.report()
> rdp    <- read.dist.plot(sr)
```

So those are the minimal commands, and every other paramter has default values. We encourage you to read the help files to learn about these parameters in more detail.

6 Load data

We load data into R by using the load.bam function. This function takes a number of parameters. We first load the package, then locate an example BAM file that is distributed with viRome, and read in the data:

```
> library(viRome)
> infile <- system.file("data/SRR389184_vs_SINV_sorted.bam", package="viRome")
> bam    <- read.bam(bamfile = infile, chr = "SINV", start = 1, end = 11703,
+                    what = c("qname", "flag", "rname", "strand",
+                             "pos", "qwidth", "mapq", "cigar", "mrnm",
+                             "mpos", "isize", "seq"),
+                    tag = c("NM"), removeN = TRUE)
```

To see documentation for all of the parameters to read.bam, access the help for the function:

```
> ?read.bam
```

The parameters we provide to read.bam are:

- bamfile: location of the target BAM file on disk
- chr: the name of the reference within the BAM file. We can only load data from one reference at a time
- start: the minimum base within the reference to load alignments from
- end: the maximum base within the reference to load alignments from
- what: the fields to load. read.bam uses Rsamtools to load the data, see ?scanBam for an explanation of these fields

- tag: the names of tags within the BAM file to load
- removeN: whether or not to remove sequences that contain N's

We then simply need to deal with any soft- and hard- clipping that has been applied to the reads by the aligner, and calculate some statistics about it:

```
> bamc <- clip.bam(bam)
```

To find out exactly what clip.bam does, you can read the help:

```
> ?clip.bam
```

However, here is an excerpt:

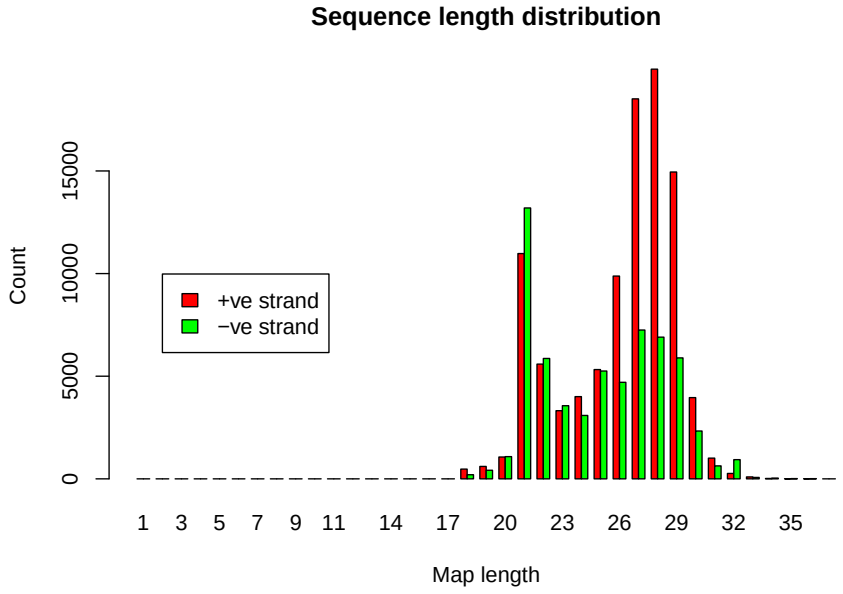
- Aligned sequences are stored in BAM files with the caveat that they may have been soft-clipped - i.e. the clipped sequence was not used in the alignment, but is included in the BAM file. This function looks at the CIGAR string and clips data in the "seq" column appropriately. For example, consider the CIGAR string "12S25M" and the sequence read "CACCCGAGAATACCCCAGAACCATTATGCTGTGACTT". The sequence read is 37bp long; however, the cigar string tells us that only 25 "matched" (25M) i.e. only 25 were used in the alignment. The CIGAR string also tells us that 12bp were soft-clipped (12S). We can tell that they were soft-clipped from the start of the read as 12S occurs before 25M

Therefore, clip.bam parses the CIGAR string for every read, and calculates mapping statistics based on the information.

7 Distribution of mapped read-lengths

Once the BAM file has been read into viRome, the first stage of any analysis is to look at the distribution of read lengths mapped to the genome. A peak at 21-22bp may indicate a siRNA response, and a peak at 25-29bp may inidcate a piRNA response:

```
> b <- barplot.bam(vdf = bamc, minlen = 1, maxlen = 37,
+                  poscol="red", negcol="green",
+                  main = "Sequence length distribution",
+                  xlab = "Map length", ylab = "Count",
+                  legend = c("+ve strand", "-ve strand"),
+                  legendx = NULL, legendy = NULL)
```



A feature of viRome is that many of the plotting functions also return the data to the user:

```
> b
```

	lgth	pos	neg
1	1	0	0
2	2	0	0
3	3	0	0
4	4	0	0
5	5	0	0
6	6	0	0
7	7	0	0
8	8	0	0
9	9	0	0
10	10	0	0
11	11	0	0
12	12	0	0
13	13	0	0
14	14	0	0
15	15	0	0
16	16	0	0
17	17	0	0
18	18	475	201
19	19	605	418

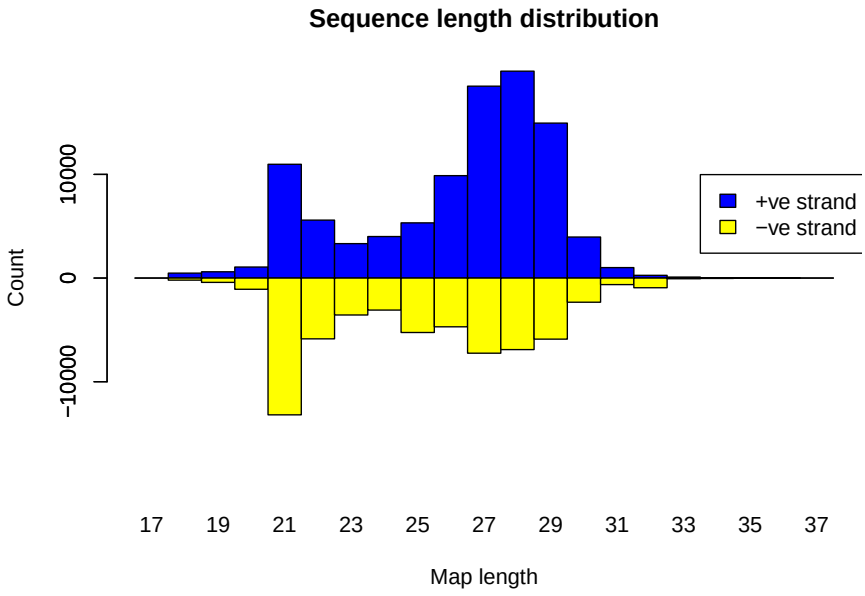
```
20   20  1064  1083
21   21 10976 13197
22   22  5590  5865
23   23  3321  3563
24   24  4006  3089
25   25  5324  5253
26   26  9879  4703
27   27 18512  7250
28   28 19959  6903
29   29 14949  5893
30   30  3958  2331
31   31  1009   632
32   32   263   936
33   33    94    69
34   34    19    37
35   35     4    12
36   36     5    10
37   37     0     0
```

One of the advantages of this is you may wish to plot the data using a different package.

We can also make the plot look a bit nicer using options built into viRome:

```
> b <- barplot.bam(vdf = bamc, minlen = 17, maxlen = 37,
+                  poscol="blue", negcol="yellow",
+                  main = "Sequence length distribution",
+                  xlab = "Map length", ylab = "Count",
+                  legend = c("+ve strand", "-ve strand"),
+                  legendx = 17, legendy = NULL, down=TRUE,
+                  space=c(0,0))
```

We have changed the colour scheme, and made the counts on the negative strand point downwards:



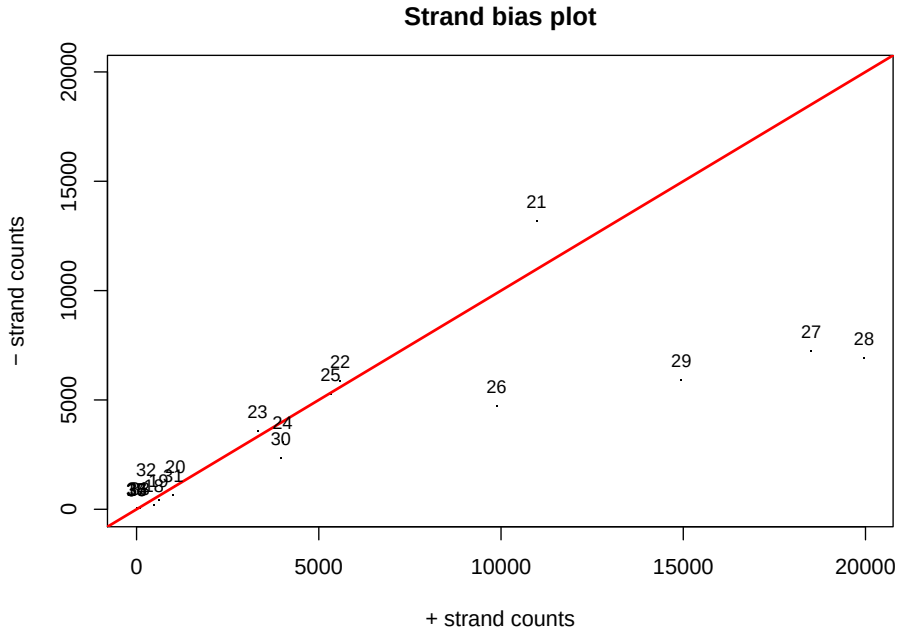
From inspecting this image we can see a number of patterns in the data

- There is a peak at 21bp, suggesting a siRNA response
- There is also a peak betwen 25-29bp, suggesting a piRNA response
- The number of reads on the positive and negative strands for 21bp reads appears equal
- There appear to be more 25-29bp reads on the positive strand compared to the negative strand

We can further visualise a potential strand bias using the size.strand.bias.plot function:

```
> size.strand.bias.plot(b, mar=c(4,4,3,1), pch=".", tpos=3, cextxt=0.8)
```

Here we provide the function with "b", the output of the barplot.bam function. We also set some additional plotting parameters, and we end up with this:



This is an XY-scatterplot of counts on the positive and negative strand for each read-length. The red line indicates y=x. We can see that for read lengths 26, 27, 28 and 29 there is a bias towards the positive strand.

8 Position of reads mapped to the genome: heatmaps

So, from the above analysis we have evidence that there is both an siRNA and a piRNA response in this sample. We also have a suggestion that there may be a strand bias in the 25-29bp reads.

The next stage is to analyse where the reads are mapping in the genome. We will first take a global view and look at all read lengths and all positions in the reference.

To do this, we must count the occurrence of each read length at each position in the genome. The function we use to do this is called summarise.by.length. This function requires a lot of RAM and processing power. However, on the windows PC specified above, these commands take approximately 20 seconds:

```
> dm <- summarise.by.length(bamc)
> dmp <- summarise.by.length(bamc, strand="pos")
> dmn <- summarise.by.length(bamc, strand="neg")
```

First, we cound occurrence for both strands, and store those counts in a variable called ”dm”. We then calculate similar summaries, but this time limit to the positive and negative strands respectably.

This will return a data matrix, the rows of which are genomic positions, the columns of which are read lengths and the values of which are counts:

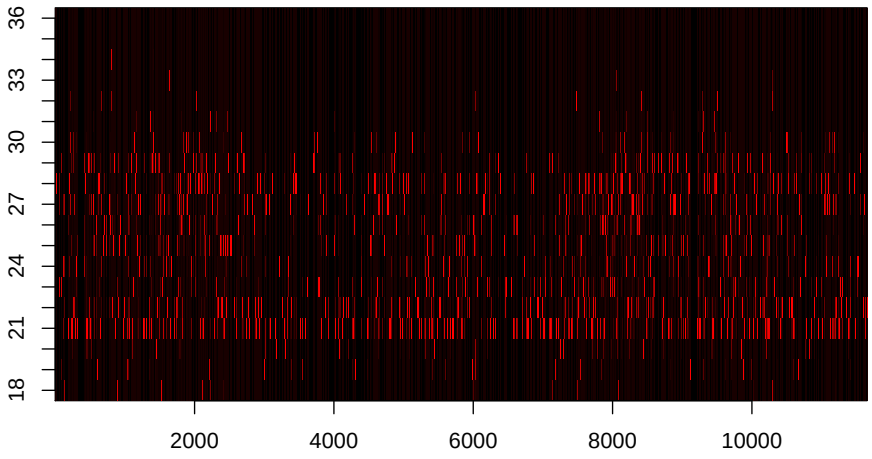
```
> dm[1:10,2:19]
```

	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36
1	0	3	1	34	33	106	76	34	33	5	15	0	0	0	0	0	0	0
2	0	0	0	3	2	10	2	1	4	0	1	0	0	0	0	0	0	0
3	1	2	0	61	10	8	15	1	11	0	4	0	0	0	0	0	0	0
4	0	3	0	2	4	6	2	1	2	9	11	1	0	0	0	0	0	0
5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
6	0	0	0	5	0	0	1	0	0	0	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
8	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
9	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
10	0	0	0	0	2	0	0	0	0	0	0	0	0	0	0	0	0	0

We can now use these in a range of plotting functions.
The first function we will use is size.position.heatmap:

```
> size.position.heatmap(dm, mar=c(3,3,2,1))
```

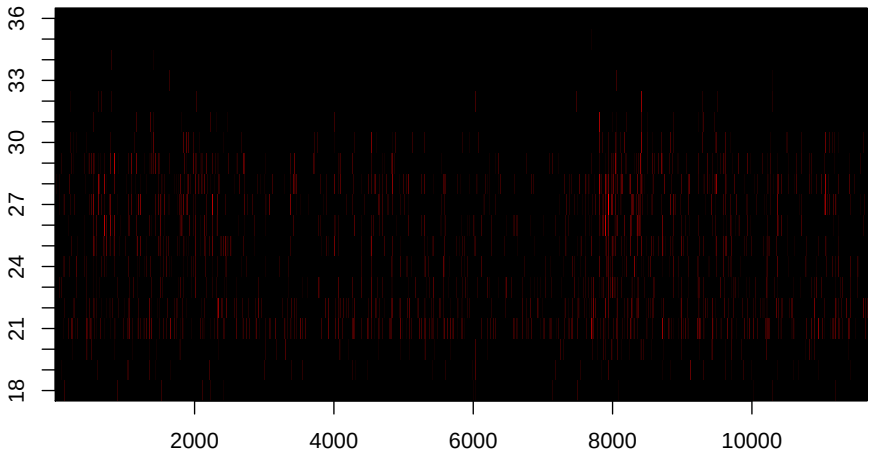
This draws a heatmap, with read lengths as rows and genomic position as columns. With default parameters, the data are scaled for each genomic position:



From this we can see the distribution of read lengths mapped throughout the genome. We can see hot-spots for each read length, and there is a suggestion that the 21bp reads are more evenly distributed throughout the genome. We can look at the data another way using additional options:

```
> size.position.heatmap(dm, scale=FALSE, log=TRUE, mar=c(3,3,2,1))
```

With these parameters, we are telling the function to log the data (this helps with scale), not to use R's in-built scale function and to adjust the plotting margins. The command produces:



As we have log transformed the data, and chosen not to scale it, we can begin to see additional patterns - specifically, we can see that there is a hotspot for 25-29bp reads around position 8000 in the genome.

Both of these plots show counts for both strands - but what about strand bias? Well, with the following commands we can use split.screen() to compare counts for both strands with those for the positive and negative strands:

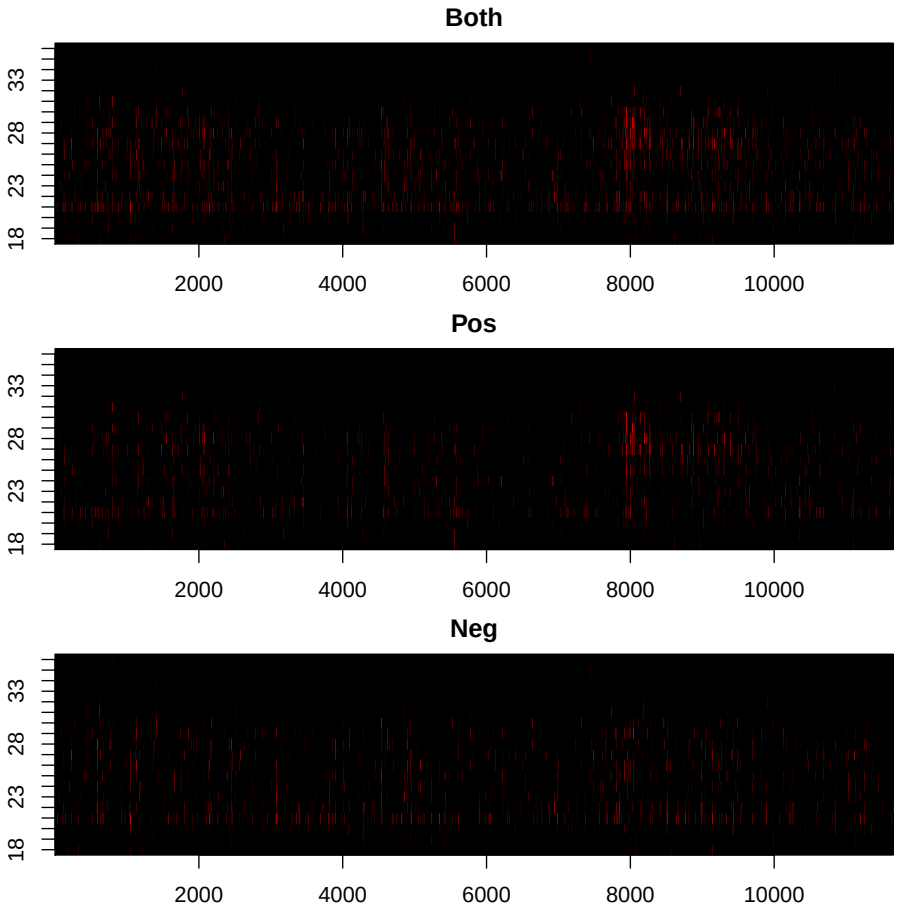
```
> split.screen(c(3,1))

[1] 1 2 3

> screen(1)
> size.position.heatmap(dm, log=TRUE, scale=FALSE, mar=c(2,2,2,1), main="Both")
> screen(2)
> size.position.heatmap(dmp, log=TRUE, scale=FALSE, mar=c(2,2,2,1), main="Pos")
> screen(3)
> size.position.heatmap(dmn, log=TRUE, scale=FALSE, mar=c(2,2,2,1), main="Neg")
> close.screen(all=TRUE)
```

Here we are using split.screen to create a plot with three graphics as 3 rows and 1 column. We then plot data for both strands at the top, the positive strand in the middle and the negative strand at the bottom:

```
[1] 1 2 3
```



This image indicates yet another pattern - not only does there appear to be a peak of 25-29bp matches around position 8000, but it also appears to be more prominent on the positive strand.

9 Position of reads mapped to the genome: barplots

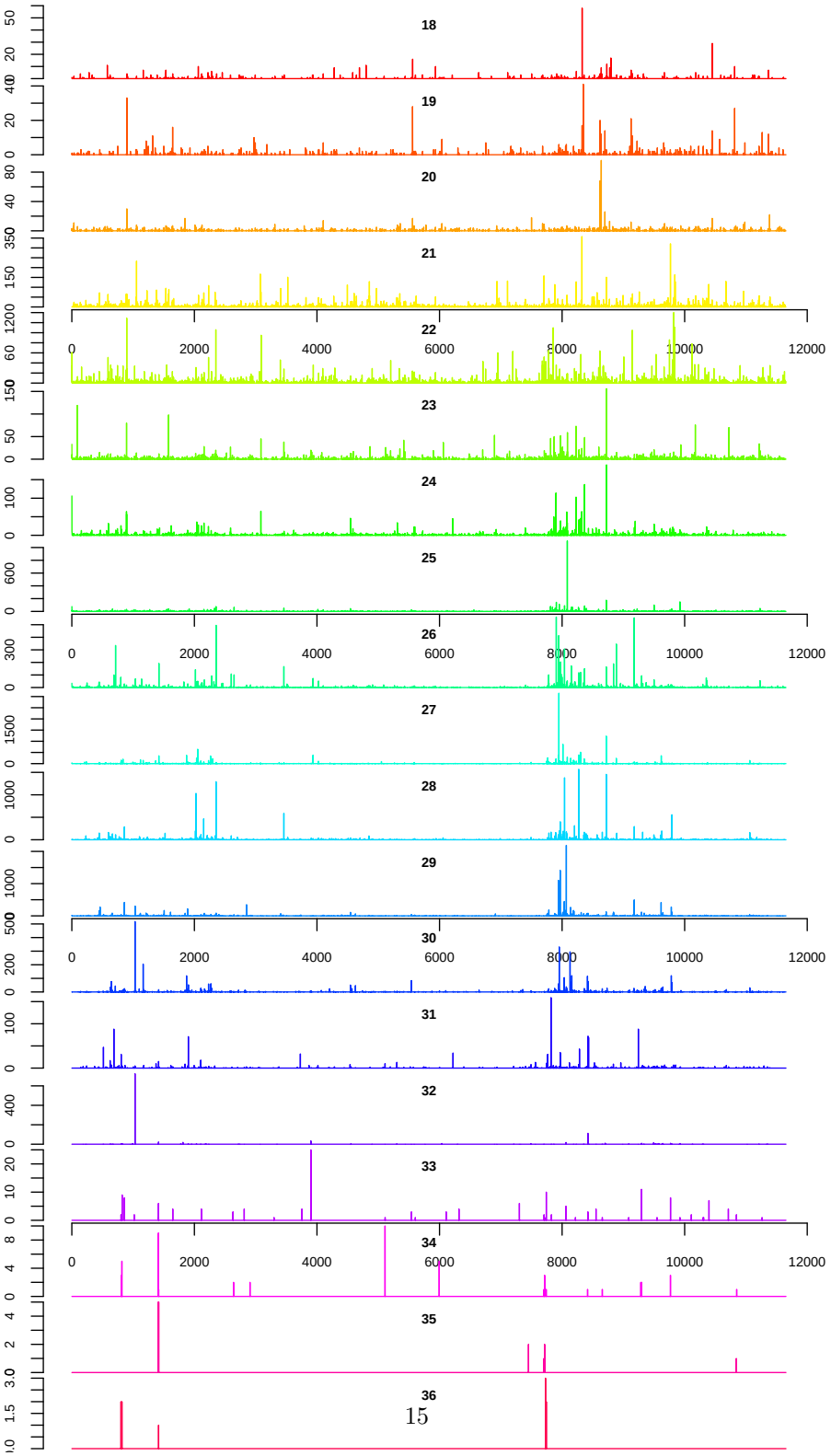
Using the data we calculated above, we can also examine the distribution of reads using barplots. The first function we will use looks at all reads across all positions, and plots a barplot for each read length:

```
> stacked.barplot(dm, internal.margins=c(0,2,0,1), skip.x=4, main.adj=0.5)

[1] "WARNING: you have chosen to plot a lot of graphs to the current device"
[1] "WARNING: this may be too many for the device to cope with"
[1] "WARNING: if you get an error message, then consider using a bigger device"
[1] "WARNING: with bigger dimensions: see ?png ?jpeg ?pdf"
```

This command takes ”dm”, which includes counts for both strands and creates a stacked barplot. We use parameters to adjust the plotting margins, to adjust the size of the title for each plot and to only plot an x-axis every 4 plots:

```
[1] "WARNING: you have chosen to plot a lot of graphs to the current device"
[1] "WARNING: this may be too many for the device to cope with"
[1] "WARNING: if you get an error message, then consider using a bigger device"
[1] "WARNING: with bigger dimensions: see ?png ?jpeg ?pdf"
```

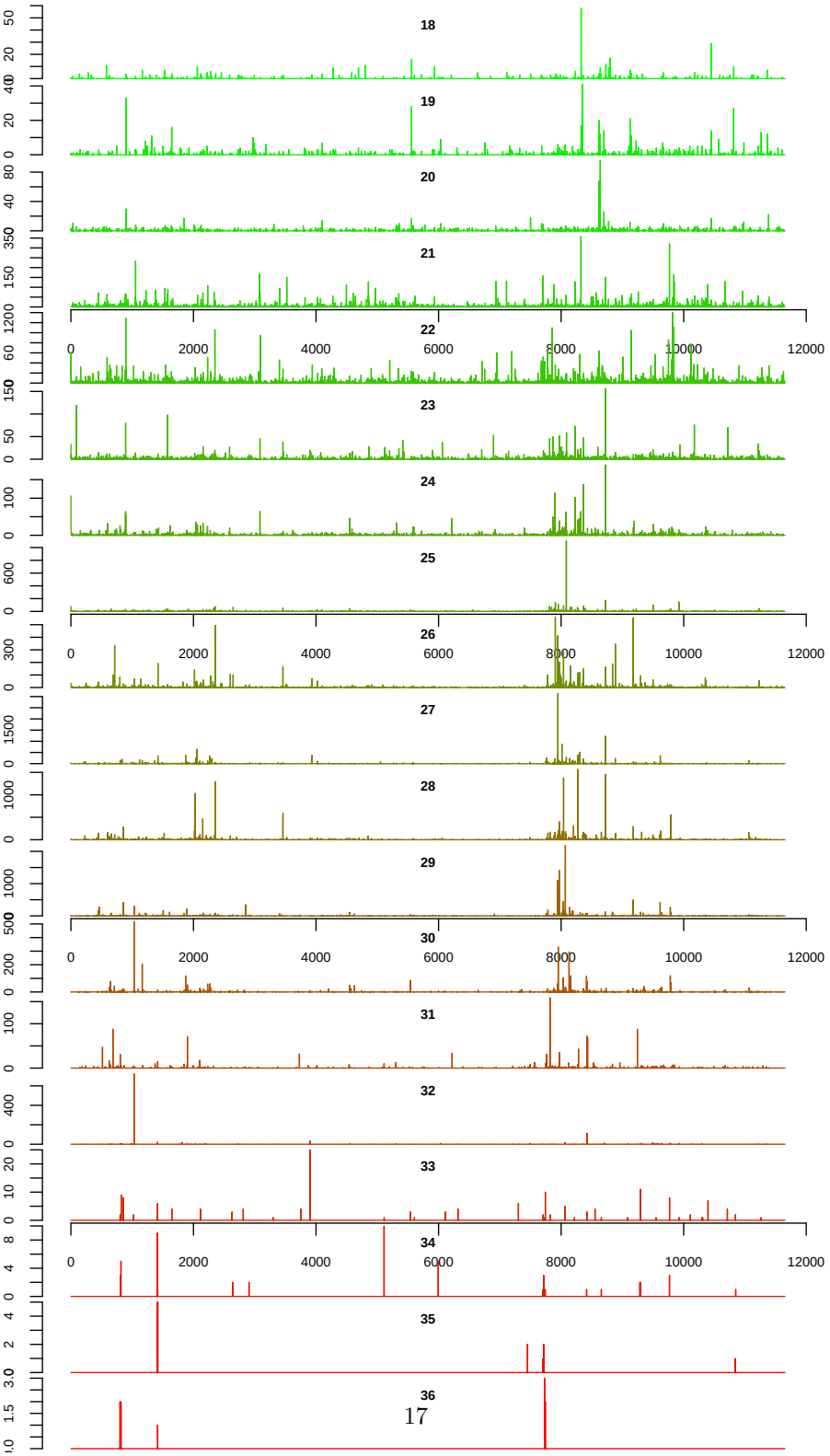

18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
0
2000
4000
6000
8000
10000
12000

The default colour scheme here is ”rainbow”, which can be difficult to see.
We can change this:

```
> stacked.barplot(dm, internal.margins=c(0,2,0,1), skip.x=4,
+                 main.adj=0.5,
+                 col.fun=colorRampPalette(c("green", "red"),
+                         space = "rgb"))
```

```
[1] "WARNING: you have chosen to plot a lot of graphs to the current device"
[1] "WARNING: this may be too many for the device to cope with"
[1] "WARNING: if you get an error message, then consider using a bigger device"
[1] "WARNING: with bigger dimensions: see ?png ?jpeg ?pdf"
```

```
[1] "WARNING: you have chosen to plot a lot of graphs to the current device"
[1] "WARNING: this may be too many for the device to cope with"
[1] "WARNING: if you get an error message, then consider using a bigger device"
[1] "WARNING: with bigger dimensions: see ?png ?jpeg ?pdf"
```


18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
0
2000
4000
6000
8000
10000
12000

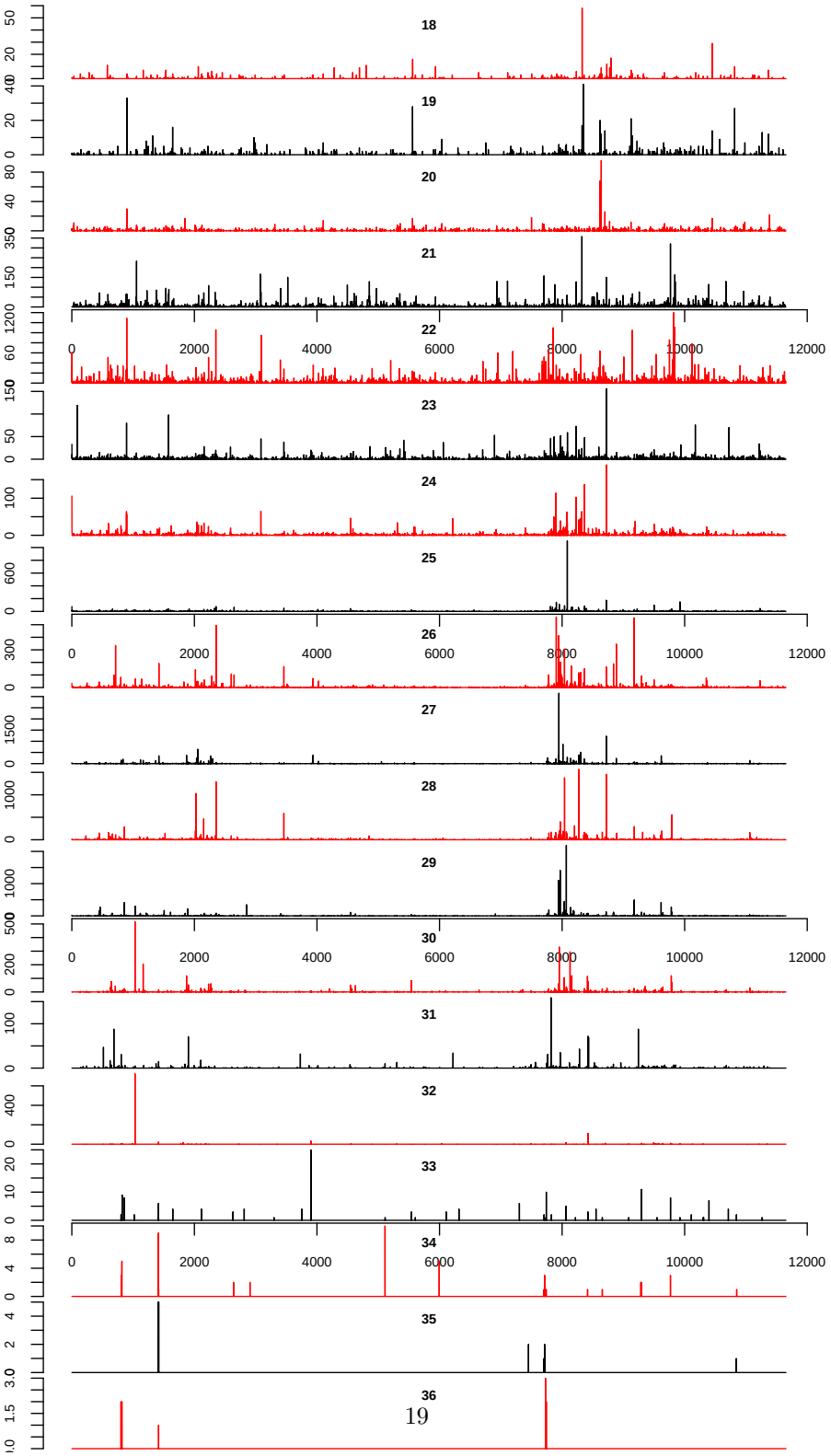
Here again we can see the pattern that 21 and 22bp reads seem to be distributed throughout the genome, but that 25-29bp reads are a bit more localised, with a particularly large peak around position 8000.

As of version 0.7, we have added two new parameters to stacked.barplot. The first, bicol, allows the user to provide a vector of two colours, and those colours will be used for alternate graphs:

```
> stacked.barplot(dm, internal.margins=c(0,2,0,1),
+                 skip.x=4, main.adj=0.5,
+                 bicol=c("red","black"))

[1] "WARNING: you have chosen to plot a lot of graphs to the current device"
[1] "WARNING: this may be too many for the device to cope with"
[1] "WARNING: if you get an error message, then consider using a bigger device"
[1] "WARNING: with bigger dimensions: see ?png ?jpeg ?pdf"

[1] "WARNING: you have chosen to plot a lot of graphs to the current device"
[1] "WARNING: this may be too many for the device to cope with"
[1] "WARNING: if you get an error message, then consider using a bigger device"
[1] "WARNING: with bigger dimensions: see ?png ?jpeg ?pdf"
```

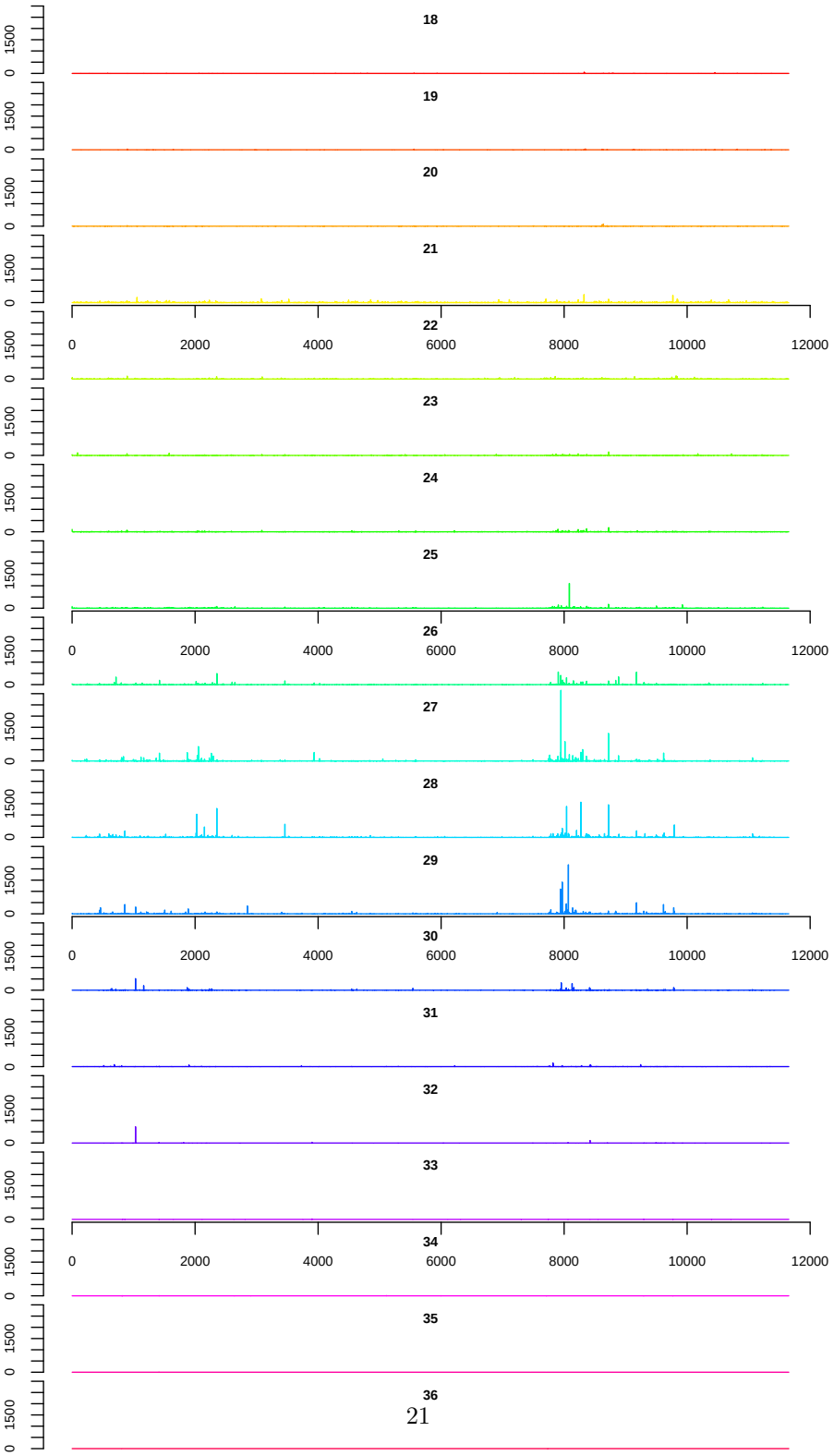

18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
0
2000
4000
6000
8000
10000
12000

The second, samey, forces stacked.barplot to create the same Y-axis for each of the stacked barplot (by default, each has its own):

```
> stacked.barplot(dm, internal.margins=c(0,2,0,1),
+                 skip.x=4, main.adj=0.5,
+                 samey=TRUE)

[1] "WARNING: you have chosen to plot a lot of graphs to the current device"
[1] "WARNING: this may be too many for the device to cope with"
[1] "WARNING: if you get an error message, then consider using a bigger device"
[1] "WARNING: with bigger dimensions: see ?png ?jpeg ?pdf"

[1] "WARNING: you have chosen to plot a lot of graphs to the current device"
[1] "WARNING: this may be too many for the device to cope with"
[1] "WARNING: if you get an error message, then consider using a bigger device"
[1] "WARNING: with bigger dimensions: see ?png ?jpeg ?pdf"
```


18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
0
1500
0
2000
4000
6000
8000
10000
12000

As can be seen above, due to the predominance of the longer reads, the bars for the shorter reads do not appear - however, this plot accurately relays that information.

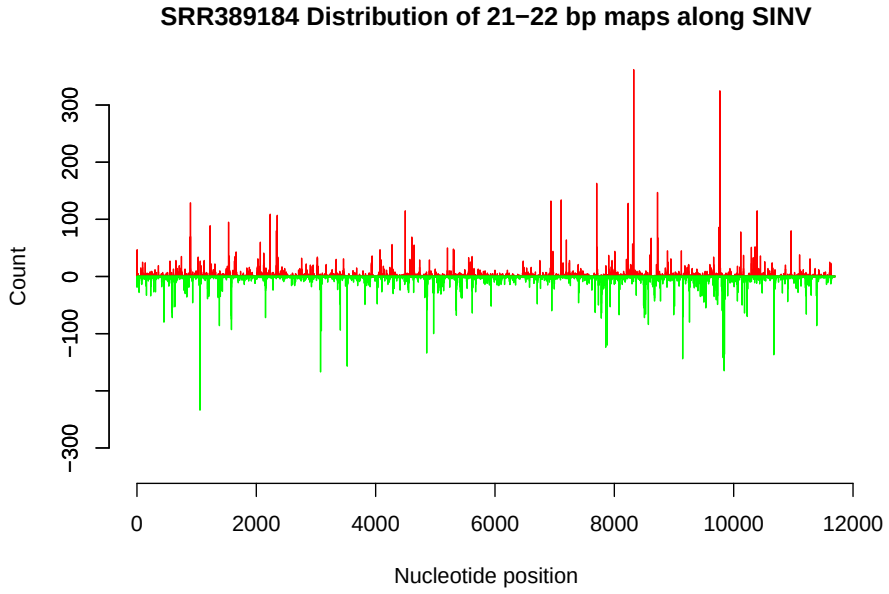
It's now time to focus in on specific patterns in the 21-22bp (siRNA) and 25-29bp (piRNA) data.

10 Genomic position of 21-22bp reads

The above plots are great for looking at global patterns, but we can now focus in on particular subsets of reads. For example, to look at the genomic position of the 21-22bp reads (siRNA response):

```
> sirna <- position.barplot(vdf = bamc, minlen = 21,
+                           maxlen = 22, reflen = 11703,
+                           samp = "SRR389184")
```

This function takes the output of clip.bam, filters on reads longer than minlen and shorter than maxlen and produces a barplot:



Using this image we can get a far more accurate picture of where in the genome the 21-22bp reads are mapping, and where hotspots exist.

As with the barplot.bam function, here the data are returned should you wish to plot the data using another function:

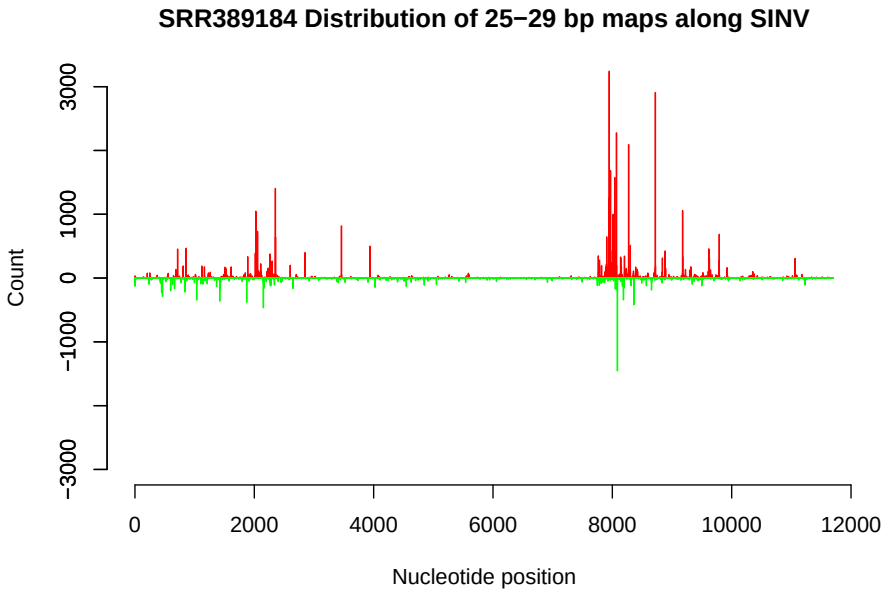
```
> sirna[1:20,]
```

```
   position poscount negcount
1         1       16       19
2         2        0        3
3         3       47       14
4         4        0        2
5         5        0        0
6         6        0        5
7         7        0        0
8         8        0        0
9         9        0        0
10       10        0        0
11       11        0        0
12       12        0        0
13       13        0        0
14       14        0        0
15       15        0        0
16       16        0        0
17       17        0        0
18       18        0        0
19       19        0        0
20       20        0        0
```

11 Genomic position of 25-29bp reads

We can also use the same function to look at 25-29bp reads (the piRNA response):

```
> pirna <- position.barplot(vdf = bamc, minlen = 25,
+                           maxlen = 29, reflen = 11703,
+                           samp = "SRR389184")
```



In contrast to the 21-22bp reads, we see far clearer localisation of the 25-29bp reads, specifically:

- A large peak on the positive strand around position 8000
- The above peak is mostly concentrated on the psoitive strand
- A smaller peak just after around position 2100, again focused on the positive strand
- A large stretch in the middle of the genome with very few 25-29bp reads

At this stage this is an explanatory analysis - we make no attempt to interpret these patterns. The purpose of viRome is merely to find and present those patterns. Differences in read lengths, position in the genome and biases in the strand may be due to issues with the sequencing technology used, or indeed the software used to map the data. These options should be eliminated before interpreting the data in terms of a biological phenomenon.

However, we can speculate from these graphs that, whilst the siRNA response seems to occur throughout the genome, the piRNA response appears more localised, and we can design experiments to attempt to confirm these hypotheses.

12 Generating a sequence report

The SAM/BAM format is well known to bioinformaticians, however, often we need to summarise and count each individual sequence and where it maps. This

is done using the sequence.report function:

```
> sr <- sequence.report(bamc, minlen=1, maxlen=37)
```

Again, the function can be used to look at all reads, or be limited to subsets. An example of the report is below:

```
> sr[1:10,]
```

```
    ref pos strand                          seq len count
6  SINV   1      +        ATTGACGGCGTAGTACACACTA  22    16
4  SINV   1      +       ATTGACGGCGTAGTACACACTAT  23    10
1  SINV   1      +      ATTGACGGCGTAGTACACACTATT  24    34
12 SINV   1      +     ATTGACGGCGTAGTACACACTATTG  25    11
17 SINV   1      +    ATTGACGGCGTAGTACACACTATTGA  26     1
9  SINV   1      +   ATTGACGGCGTAGTACACACTATTGAA  27     7
11 SINV   1      + ATTGACGGCGTAGTACACACTATTGAATC  29    15
15 SINV   1      -            ATTGACGGCGTAGTACAC  18     1
14 SINV   1      -          ATTGACGGCGTAGTACACAC  20     3
16 SINV   1      -         ATTGACGGCGTAGTACACACT  21     1
```

This can be exported to csv format using write.csv, and opened in Excel.

13 Base composition

Research has shown that RNAs within the piRNA pathway often show a U1(T1) and A10 bias - that is, far more often than you would expect by chance, piRNAs have a U/T at the first base and an A at the tenth base.

NOTE some users have pointed out that we should be showing U instead of T (as we are representing RNA). However, we disagree. It is a technicality, but we have not actually measured RNA, we have measured the cDNA form of that RNA. As far as we know, the Helicos system is the only system to directly sequence RNA, all other systems (including the Illumina/Solexa system used in this data) converts RNA to DNA first. As we have measured DNA, we show T and not U.

Relative frequencies of the four bases can be calculated using the make.pwm function:

```
> pwm1 <- make.pwm(bamc, minlen=25, maxlen=29, strand="neg")
> pwm2 <- make.pwm(bamc, minlen=25, maxlen=29)
```

This function takes the output of clip.bam, limits to reads within a certain range, and calculates counts based on reads mapping to the positive (default) or negative strand.

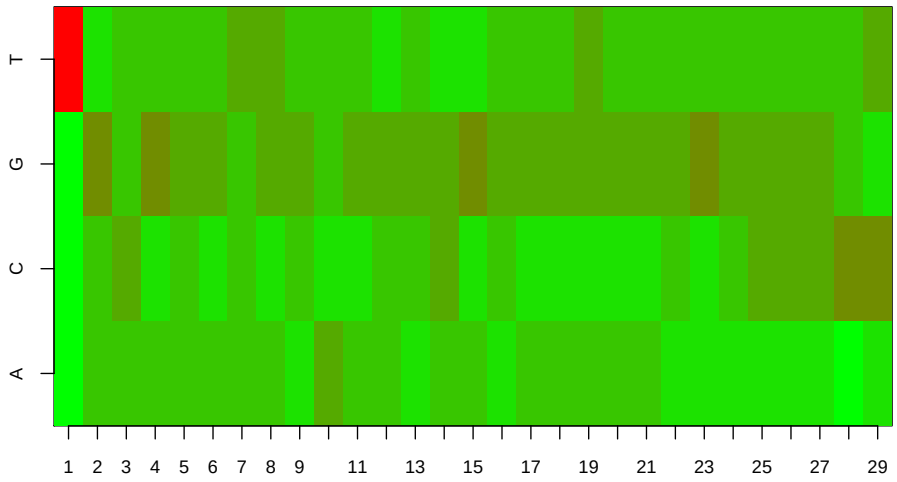
The data returned are pretty self explanatory - rows are bases, columns are positions and the values are relative proportions:

```
> pwm1[,1:8]
```

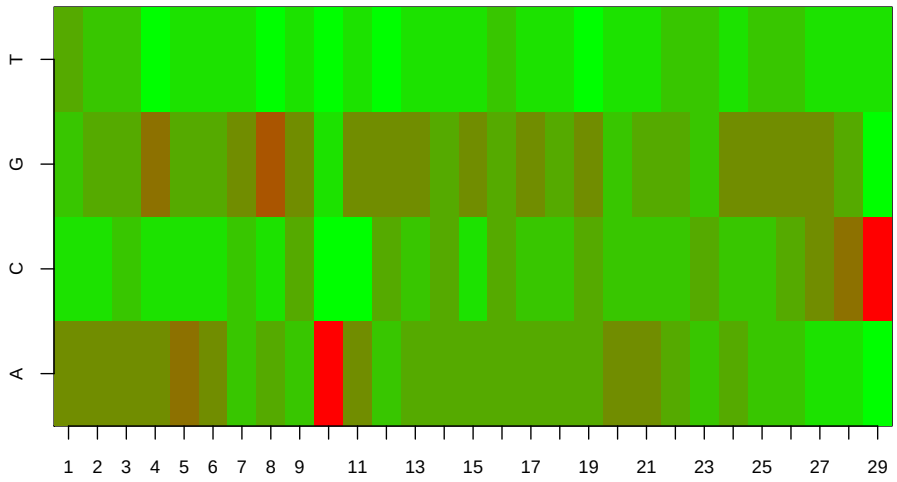
```
           1         2         3         4         5         6         7
A 0.07719485 0.2305846 0.2146524 0.2315179 0.2404173 0.2260183 0.2196854
C 0.05496300 0.2228185 0.3230118 0.1675555 0.2316846 0.1797880 0.2191187
G 0.07396174 0.3946070 0.2035864 0.3807413 0.3002800 0.3438437 0.2723152
T 0.79388041 0.1519899 0.2587494 0.2201853 0.2276182 0.2503500 0.2888807
          8
A 0.2233851
C 0.1824545
G 0.2948803
T 0.2992800
```

We can visualise these in a number of ways. Firstly, using the viRome function pwm.heatmap:

```
> pwm.heatmap(pwm1, col.fun=colorRampPalette(c("green","red"), space="rgb"))
```



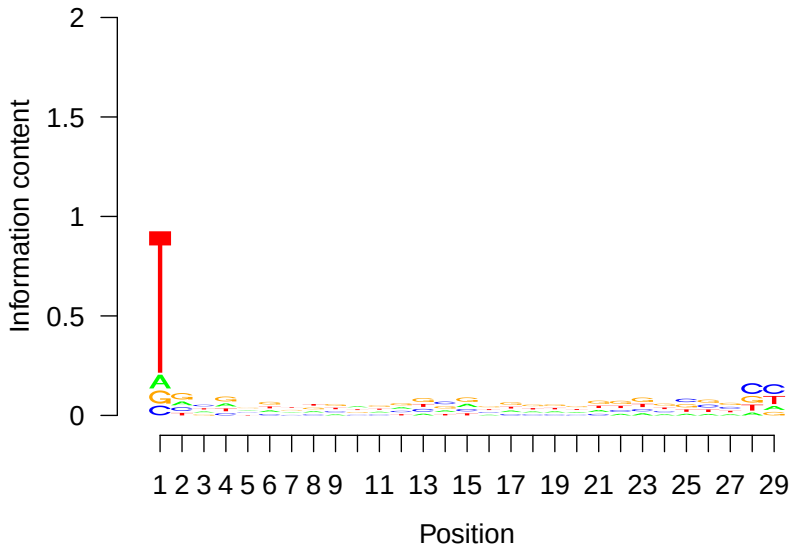

```
> pwm.heatmap(pwm2, col.fun=colorRampPalette(c("green","red"), space="rgb"))
```



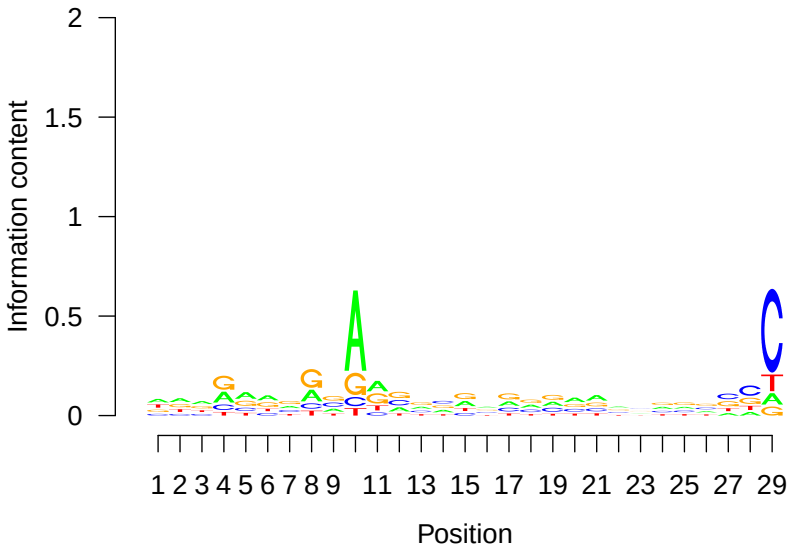
Using a green-to-red colour scheme, we can clearly see the T1 and A10 biases.

We can also visualise the data as sequence logos, using the (third party) seqLogo package:

```
> library(seqLogo) # must have this installed!
> seqLogo(pwm1)
```

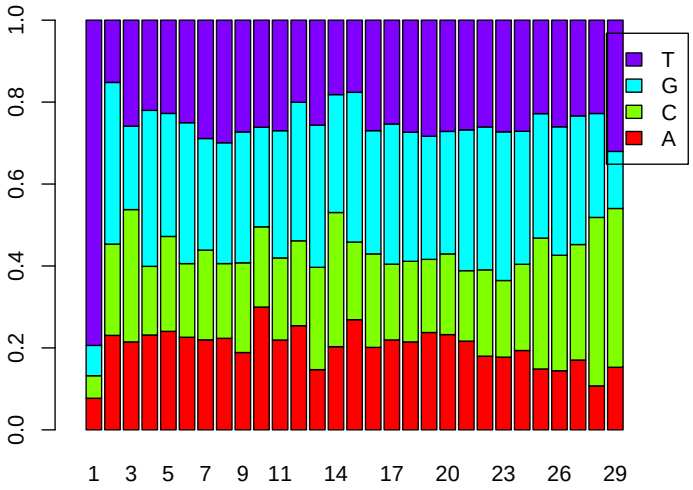


```
> library(seqLogo) # must have this installed!
> seqLogo(pwm2)
```



Of course, it also possible to use the humble barplot function to more accurately visualise the proportions:

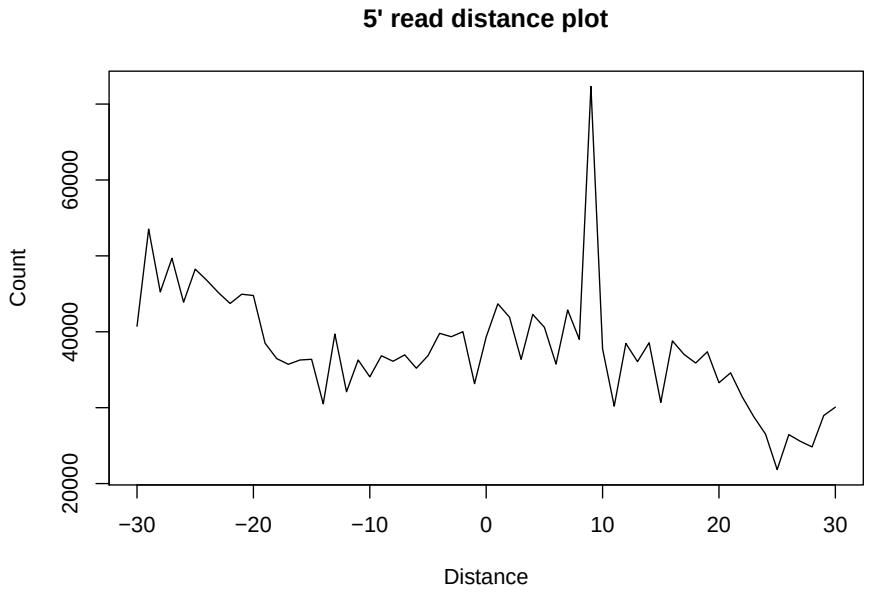
```
> barplot(pwm1, col=rainbow(4), legend.text=rownames(pwm1),
+         args.legend=list(x=39), xlim=c(0,45))
```



14 5' read-distance plots

Finally, we plot the distance between the 5' ends of overlapping reads on opposite strands. A peak at 10bp has been shown to be a signature of the piRNA response. The read.dist.plot function takes every read mapped to a given strand and within a given size range, counts the frequency of that read, and then counts the frequency of reads mapping to every location on the opposite strand within a given window. A range of summary functions are offered by the function. Despite the report of this 10bp distance in the literature, this paper represents the first full and accurate description of how the data may be summarized and plotted, and viRome represents the first software implementation.

```
> rdp <- read.dist.plot(sr, minlen=25, maxlen=29, method="add")
```



Again, the data are returned to the user, so if you choose to you can plot the data using a different package/function.

```
> rdp
```

```
   loc count
38 -30 40729
27 -29 53529
28 -28 45248
29 -27 49689
30 -26 43893
31 -25 48253
32 -24 46775
33 -23 45149
34 -22 43729
57 -21 44951
55 -20 44775
54 -19 38501
52 -18 36468
48 -17 35718
49 -16 36281
58 -15 36375
56 -14 30510
43 -13 39701
44 -12 32100
```

```
50 -11 36286
39 -10 34060
40  -9 36841
35  -8 36114
36  -7 36960
53  -6 35202
51  -5 36850
41  -4 39794
45  -3 39349
37  -2 40009
46  -1 33164
42   0 39319
59   1 43683
19   2 41931
20   3 36368
21   4 42287
22   5 40591
23   6 35741
24   7 42869
25   8 39009
26   9 72316
11  10 37753
12  11 30199
13  12 38470
14  13 36065
15  14 38557
16  15 30694
17  16 38798
18  17 37019
61  18 35882
60  19 37366
47  20 33286
10  21 34592
9   22 31420
8   23 28792
1   24 26520
2   25 21840
3   26 26450
4   27 25562
5   28 24828
6   29 28966
7   30 30066
```

15 Closing comments

And there we have it!

viRome is under active development and a manuscript is under review. Please check back regularly for updates.

Comments, suggestions etc to mick.watson@roslin.ed.ac.uk

`http://www.ark-genomics.org/services-bioinformatics/virome`