An overview of package *girafe*

Joern Toedling

April 22, 2010

1 Introduction

The intent of package *girafe* is to facilitate the functional exploration of the alignments of multiple reads[1] from next-generation sequencing data to a genome.

It extends the functionality of the Bioconductor (Gentleman et al., 2004) packages *ShortRead* (Morgan et al., 2009) and *genomeIntervals*.

```
> library("girafe")
> library("RColorBrewer")
```

2 Workflow

We present the functionality of the package *girafe* using example data that was downloaded from the Gene Expression Omnibus database (Edgar et al., 2002, GSE10364). The example data are Solexa reads of 26 nt length derived from small RNA profiling of mouse oocytes. The data has previously been described in Tam et al. (2008).

```
> exDir <- system.file("extdata", package="girafe")
> load(file.path(exDir, "anno_mm_genint.RData"))
```

2.1 Adapter trimming

We load reads that include parts of the adapter sequence.

```
> ra23.wa  <- readFastq(dirPath=exDir, pattern=
+                       "aravinSRNA_23_plus_adapter_excerpt.fastq")
```

[1] The package has been developed for analysing single-end reads (fragment libraries) and does not support mate-pair reads yet.

```
> show(ra23.wa)
```

```
class: ShortReadQ
length: 1000 reads; width: 26 cycles
```

For removing the adapter sequence, we are using the function `trimAdapter`, which relies on the function `pairwiseAlignment` from package *Biostrings*. The adapter sequence was obtained from the GEO page of the data.

```
> adapter <- "CTGTAGGCACCATCAAT"
> ra23.na  <- trimAdapter(ra23.wa, adapter)
> show(ra23.na)
```

```
class: ShortReadQ
length: 1000 reads; width: 23 cycles
```

2.2 Importing aligned reads

The reads have been mapped to the mouse genome (assembly *mm9*) using the alignment tool *Bowtie* (Langmead et al., 2009).

The resulting tab-delimited map file can be read into an object of class *AlignedRead* using the function `readAligned`. Both, this class and this function, are defined in the Bioconductor package *ShortRead*.

```
> exA   <- readAligned(dirPath=exDir, type="Bowtie",
+    pattern="aravinSRNA_23_no_adapter_excerpt_mm9_unmasked.bwtmap")
> show(exA)
```

```
class: AlignedRead
length: 1689 reads; width: 23 cycles
chromosome: chr14 chr17 ... chr3 chr14
position: 115443405 13011859 ... 68813840 62250772
strand: + + ... + -
alignQuality: NumericQuality
alignData varLabels: similar mismatch
```

The object of class *AlignedRead* can be converted into an object of class *AlignedGenomeIntervals*, which is the main class of package *girafe*.

```
> exAI <- as(exA, "AlignedGenomeIntervals")
> organism(exAI) <- "Mm"
```

2.3 Exploring the aligned reads

```
> show(exAI)
```

```
1286 genome intervals with aligned reads on 22 chromosome(s).
Organism: Mm
```

Which chromosomes are the intervals located on?

```
> table(seq_name(exAI))
```

```
 chr1 chr10 chr11 chr12 chr13 chr14 chr15 chr16 chr17 chr18 chr19  chr2  chr3
  112    50    60    53    65    74    59    48    47    43    19    87    62
 chr4  chr5  chr6  chr7  chr8  chr9 chrMT  chrX  chrY
   57    52    82    51    57    69     5   132     2
```

A subset of the intervals on a specific chromosome can be obtained using subsetting via `'['`.

```
> detail(exAI[seq_name(exAI)=="chrMT"])
```

```
  start   end seq_name strand reads matches                sequence
1   964   986    chrMT      +     1       1 GTTTATGAGAGGAGATAAGTTGT
2 11613 11635    chrMT      +    10       2 AAGAAAGATTGCAAGAACTGCTA
3 11613 11635    chrMT      +     1       2 AAGCAAGATTGCAAGAACTGCTA
4 11613 11635    chrMT      +     1       2 AAGAACGATTGCAAGAACTGCTA
5 11613 11635    chrMT      +     1       3 AAGAAAGATTGCAAGAACTGTTA
```

2.4 Processing the aligned intervals

Sometimes, users may wish to combine certain aligned intervals. One intention could be to combine aligned reads at exactly the same position, which only differ in their sequence due to sequencing errors. Another objective could be to combine overlapping short reads that may be (degradation) products of the same primary transcript. The function `reduce` combines a set of aligned intervals into a single aligned interval if the intervals

- are on the same strand,
- are overlapping (or contained in each other) or directly adjacent to each other AND
- have the same *read match specificity* (see below)

Read match specificity By the *read match specificity* $r(I_i)$ of an interval I_i, we refer to the total number of valid alignments of reads that have been aligned to I_i, *i.e.* the total numbers of intervals with the same reads aligned in the whole genome (or other set of reference sequences). If $r(I_i) = 1$, the reads that were aligned to the interval I_i have no

other valid alignments in the whole genome, i.e. the interval I_i is the unique match position of these reads. If $r(I_i) = 2$, the reads that were aligned to the interval I_i have exactly one other valid alignment to another interval I_j, $j \neq i$. The match specificity is stored in the `matches` slot of objects of the class *AlignedGenomeIntervals*.

We first demonstrate the `reduce` using a toy example data object.

```
> D <- AlignedGenomeIntervals(
+      start=c(1,3,4,5,8,10,11), end=c(5,5,6,8,9,11,13),
+      chromosome=rep(c("chr1","chr2","chr3"), c(2,2,3)),
+      strand=c("-","-","+","+","+","+","+"),
+      sequence=c("ACATT","ACA","CGT","GTAA","AG","CT","TTT"),
+      reads=rep(1,7), matches=c(rep(1,6),3))
> detail(D)
```

```
  start end seq_name strand reads matches sequence
1     1   5     chr1      -     1       1    ACATT
2     3   5     chr1      -     1       1      ACA
3     4   6     chr2      +     1       1      CGT
4     5   8     chr2      +     1       1     GTAA
5     8   9     chr3      +     1       1       AG
6    10  11     chr3      +     1       1       CT
7    11  13     chr3      +     1       3      TTT
```

Calling the `reduce` method on these example data results in the following:

```
> detail(reduce(D))
```

```
  start end seq_name strand reads matches sequence
1     1   5     chr1      -     2       1    ACATT
2     4   8     chr2      +     2       1    CGTAA
3     8  11     chr3      +     2       1     AGCT
4    11  13     chr3      +     1       3      TTT
```

Note that the two last intervals still show overlap. However, the last interval is a non-unique match position of the respective reads (`matches`= 3), in contrast to the other intervals.

Here is another example using the data introduced above.

```
> S <- exAI[seq_name(exAI)=="chrX" & exAI@matches==1 & exAI[,1]>1e8]
> detail(S)
```

```
      start       end seq_name strand reads matches                sequence
1 100768450 100768472     chrX      -     1       1 ATATAATACAACCTGCTAACTGT
2 101311567 101311589     chrX      -    18       1 TGAGGTTGGTGTACTGTGTGTGG
3 101311567 101311589     chrX      -    12       1 TGAGGTTGGTGTACTGTGTGTGA
4 101311567 101311589     chrX      -     2       1 TGAGGTTGGTGTACTGTGTGTGT
5 101311567 101311589     chrX      -     1       1 TGACGTTGGTGTACTGTGTGTGA
6 101311567 101311589     chrX      -     1       1 TGAGGTTGGTGTACTGTGTGCGG
7 148346896 148346918     chrX      +     4       1 TGAGGTAGTAGATTGTATAGTTT
```

Calling the `reduce` method on these data leads to the following result.

```
> detail(reduce(S))
```

```
      start       end seq_name strand reads matches                 sequence
1 100768450 100768472     chrX      -     1       1 ATATAATACAACCTGCTAACTGT
2 101311567 101311589     chrX      -    34       1 TGAGGTTGGTGTACTGTGTGTGN
3 148346896 148346918     chrX      +     4       1 TGAGGTAGTAGATTGTATAGTTT
```

Notice that the reads that match the same segment of the X chromosome differ in their last base. This ambiguity is represented in the combined aligned interval having an 'N' as the last letter.

The additional argument `exact=TRUE` can be specified if you want to solely combine intervals that have exactly the same start and stop position (but may have reads of slightly different sequence aligned to them). Consider the following example:

```
> S2 <- exAI[seq_name(exAI)=="chr11" & exAI@matches==1 & exAI[,1]>8e7]
> detail(S2)
```

```
      start       end seq_name strand reads matches                 sequence
1  86397621  86397643    chr11      -    20       1 TAGCTTATCAGACTGATGTTTAC
2  86397621  86397643    chr11      -     1       1 TAGATTATCAGACTGATGTTTAC
3  86397621  86397643    chr11      -     2       1 TAGCTTATCAGACTGATGTTCAC
4  88515338  88515360    chr11      -     1       1 GGTGCAGGGAGCGCCAGTGTCTC
5  96178500  96178522    chr11      +     2       1 TACCCTGTAGATCCGAATTTTTG
6  96178501  96178523    chr11      +     1       1 ACCCTGTAGATCCGAATTTGTGA
7 108873196 108873218    chr11      -     1       1 AGTGCGGTAACGCGACCGCTACC
```

```
> detail(reduce(S2, exact=TRUE))
```

```
      start       end seq_name strand reads matches                 sequence
1  86397621  86397643    chr11      -    23       1 TAGNTTATCAGACTGATGTTNAC
2  88515338  88515360    chr11      -     1       1 GGTGCAGGGAGCGCCAGTGTCTC
3  96178500  96178522    chr11      +     2       1 TACCCTGTAGATCCGAATTTTTG
4  96178501  96178523    chr11      +     1       1 ACCCTGTAGATCCGAATTTGTGA
5 108873196 108873218    chr11      -     1       1 AGTGCGGTAACGCGACCGCTACC
```

Notice that the 6th aligned interval in `S2` is only shifted by 1 nt from the 5th one. By default, the function `reduce` would merge them into one aligned genome interval. However, when `exact=TRUE` is specified, these two intervals are not merged since they are not at exactly the same position.

2.5 Visualizing the aligned genome intervals

The package *girafe* contains functions for visualising genome regions with aligned reads.

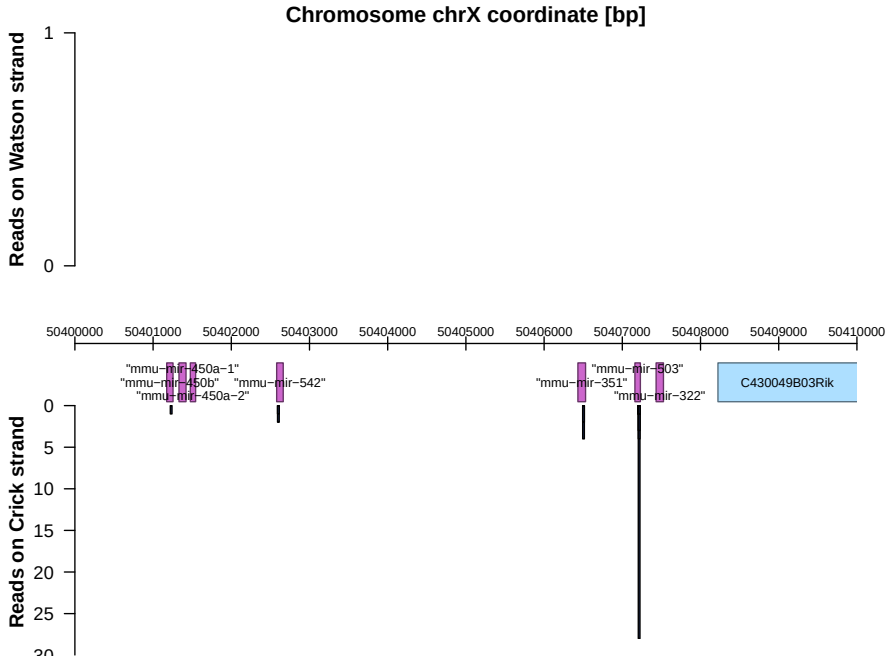

Figure 1: *A 10-kb region on mouse chromosome X. Reads aligned to the Watson strand in this region would be shown above the chromosome coordinate axis with the annotation of genome elements in this region, while reads aligned to the Crick strand are shown below. In the shown region, there only are intervals with aligned reads on the Crick strand, and these four intervals overlap with annotated micro-RNA positions.*

```
> plot(exAI, mm.gi, chr="chrX", start=50400000, end=50410000)
```

See the result in Figure 1.

Note that the annotation of genome elements (as shown in Figure 1) has to be supplied to the function. Here the object `mm.gi` contains most annotated genes and ncRNAs for the mouse genome (assembly: *mm9*). This object has been created beforehand[2] and it is of class *Genome_intervals_stranded*, a class defined in package *genomeIntervals*.

2.6 Summarising the data using sliding windows

The data can be searched for regions of defined interest using a sliding-window approach implemented in the function `perWindow`. For each window, the number of intervals with aligned reads, the total number of reads aligned, the number of unique reads aligned, the fraction of intervals on the Plus-strand, and the higher number of aligned reads at a single interval within the window are reported.

```
> exPX  <- perWindow(exAI, chr="chrX", winsize=1e5, step=0.5e5)
> head(exPX[order(exPX$n.overlap, decreasing=TRUE),])
```

[2]See the script `prepareAnnotation.R` in the package scripts directory for an example of how to create such an object.

```
      chr     start       end n.overlap n.reads n.unique frac.plus max.reads
942  chrX  50341103  50441102        18      55       18         0        28
943  chrX  50391103  50491102        18      55       18         0        28
1960 chrX 101241103 101341102         5      34        5         0        18
1961 chrX 101291103 101391102         5      34        5         0        18
1216 chrX  64041103  64141102         4       5        4         0         2
1215 chrX  63991103  64091102         3       4        3         0         2
```

2.7 Exporting the data

The package *girafe* also contains methods for exporting the data into tab-delimited text files, which can be uploaded to the UCSC genome browser[3] as 'custom tracks'.

Currently, there are methods for exporting the data in 'bed' format and 'bedGraph' format, either writing intervals from both strands into one file or into two separate files. Details about these track formats can be found at the UCSC genome browser web pages.

```
> export(exAI, con="export.bed",
+        format="bed", name="example_reads",
+        description="Example reads",
+        color="100,100,255", visibility="pack")
```

Additional arguments to the export function, besides `object`, `con`, and `format`, are treated as attributes for the track definition line, which specifies details about how the data should be visualised in the genome browser.

Users may also want to have a look at the Bioconductor package *rtracklayer* for data transfer and direct interaction between R and the UCSC genome browser.

2.8 Overlap with annotated genome features

Next, we determine the overlap of the aligned reads with annotated genome elements. In this example, the annotated genome elements are provided as an object of class *Genome_intervals_stranded*[4]. This objects needs to be created beforehand. See the script `prepareAnnotation.R` in the package scripts directory[5] for an example of how to create such an object.

```
> exOv <- interval_overlap(exAI, mm.gi)
```

How many elements do read match positions generally overlap?

```
> table(listLen(exOv))
```

[3] http://genome.ucsc.edu

[4] Objects of class *Genome_intervals* and *AlignedGenomeIntervals* are also allowed.

[5] `system.file('scripts', package='girafe')`

```
  0   1   2   3  12
797 359 122   7   1
```

What are the 12 elements overlapped by a single match position?

```
> getGffAttribute(mm.gi[exOv[[which(listLen(exOv)==12)]]],"Alias")[,1]
```

```
 [1] "Pcdha1"  "Pcdha2"  "Pcdha3"  "Pcdha4"  "Pcdha5"  "Pcdha6"  "Pcdha7"
 [8] "Pcdha8"  "Pcdha9"  "Pcdha10" "Pcdha11" "Pcdha12"
```

And in general, what kinds of annotated genome elements are overlapped by reads?

```
> (tabOv <- table(as.character(mm.gi$type)[unlist(exOv)]))
```

```
      gene      miRNA      ncRNA pseudogene       rRNA     snoRNA       tRNA
       290        296          1         25          9          1         14
```

We display these overlap classes using a pie chart.

```
> my.cols <- brewer.pal(length(tabOv), "Set3")
> pie(tabOv, col=my.cols, radius=0.95)
```

See the result in Figure 2.

Note that function `interval.overlap` only determines whether two intervals are overlapping by at least one base. For restricting the result to intervals overlapping by at least a certain number of bases or by a fraction of the interval's length, see the function `fracOverlap`.

3 A word about speed

For improving the run time on machines with multiple processors, some of the functions in package *girafe* have been implemented to make use of the functionality in package *multicore*. If the package *multicore* has been attached and initialised before calling these functions, the functions will make use of `mclapply` instead of the normal `lapply`.

For example, if *multicore* works on your machine, there should be an obvious speed improvement in the following code example.

```
> library("multicore")
> options("cores"=4) # adjust for your machine
> covAI <- coverage(exAI, byStrand=TRUE)
```

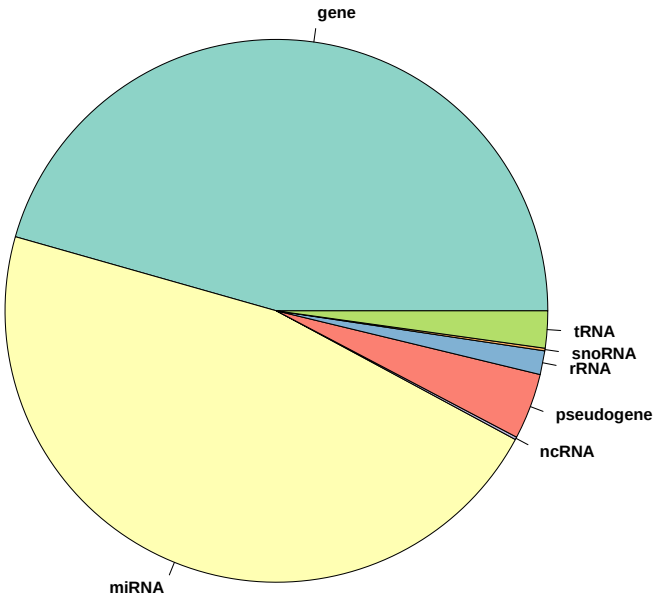

Figure 2: *Pie chart showing what kinds of genome elements are overlapped by aligned reads. Note that the proportions of the pie chart are given by the proportions among all annotated genome elements that have geq*1 *reads mapped to them and not by the total numbers of reads mapped to elements of that class, in which case the proportion of the miRNA class would be significantly larger.*

Package versions

This vignette was generated using the following package versions:

- R version 2.11.0 (2010-04-22), `x86_64-unknown-linux-gnu`
- Base packages: base, datasets, graphics, grDevices, grid, methods, stats, tools, utils
- Other packages: AnnotationDbi 1.10.0, Biobase 2.8.0, Biostrings 2.16.0, DBI 0.2-5, genomeIntervals 1.4.0, GenomicRanges 1.0.0, girafe 1.0.0, intervals 0.13.1, IRanges 1.6.0, lattice 0.18-5, org.Mm.eg.db 2.4.1, RColorBrewer 1.0-2, Rsamtools 1.0.0, RSQLite 0.8-4, ShortRead 1.6.0
- Loaded via a namespace (and not attached): BSgenome 1.16.0, hwriter 1.2

Acknowledgements

Many thanks to Nicolas Servant, Emmanuel Barillot, Constance Ciaudo, Edith Heard, Valérie Cognat, Nicolas Delhomme, and especially Patrick Aboyoun for suggestions and feedback on the package. Special thanks to Julien Gagneur and Richard Bourgon for writing *genomeIntervals* and for rapidly answering all my questions regarding the package.
The plotting functions in package *girafe* are largely based on the function `plotAlongChrom` and its auxiliary functions from package *tilingArray*, most of which were written by Wolfgang Huber.

References

R. Edgar, M. Domrachev, and A. E. Lash. Gene Expression Omnibus: NCBI gene expression and hybridization array data repository. *Nucleic Acids Res*, 30(1):207–210, Jan 2002.

R. C. Gentleman, V. J. Carey, D. J. Bates, B. M. Bolstad, M. Dettling, S. Dudoit, B. Ellis, L. Gautier, Y. Ge, J. Gentry, K. Hornik, T. Hothorn, W. Huber, S. Iacus, R. Irizarry, F. Leisch, C. Li, M. Maechler, A. J. Rossini, G. Sawitzki, C. Smith, G. K. Smyth, L. Tierney, Y. H. Yang, and J. Zhang. Bioconductor: Open software development for computational biology and bioinformatics. *Genome Biology*, 5:R80, 2004.

B. Langmead, C. Trapnell, M. Pop, and S. L. Salzberg. Ultrafast and memory-efficient alignment of short DNA sequences to the human genome. *Genome Biology*, 10(3):R25, 2009.

M. Morgan, S. Anders, M. Lawrence, P. Aboyoun, H. Pages, and R. Gentleman. ShortRead: a Bioconductor package for input, quality assessment and exploration of high-throughput sequence data. *Bioinformatics*, 25(19):2607–2608, Oct 2009.

O. H. Tam, A. A. Aravin, P. Stein, A. Girard, E. P. Murchison, S. Cheloufi, E. Hodges, M. Anger, R. Sachidanandam, R. M. Schultz, and G. J. Hannon. Pseudogene-derived small interfering RNAs regulate gene expression in mouse oocytes. *Nature*, 453(7194):534–538, May 2008.