Finding Structural Variants in Short Read, Paired-end Sequence Data with R and Bioconductor

Sean Davis

National Cancer Institute,
National Institutes of Health
Bethesda, MD, USA

sdavis2@mail.nih.gov

November 19, 2009

Abstract

Second-generation sequencing technologies, when used to sequence genomic DNA with paired-end reads, are extremely powerful and sensitive tools for discovering structural variation within a genome. Insertions and deletions as well as translocations (and inversions–not discussed directly here), if present in the genomic DNA under investigation, may be captured by examining paired-end reads that are "unusual". This little vignette lays out the concepts for finding these structural variants in a simulated genome (taken from human chromosome 17).

1 Introduction

In both "normal" and "disease" states, the structure of the genome is a potentially important factor in determining biology. Insertions of DNA can alter the regulatory regions or coding sequence of genes by introducing foreign regulatory elements or by disrupting coding sequence or splice sites. Deletions can have similar effects by removing functional elements in DNA. Translocations can place one functional element (such as a transcription-factor binding site) into proximity of another functional element such as a miRNA or proteing-coding gene, thus affecting the normal regulatory pathways. Furthermore, translocations may act more directly by producing fusion transcripts or chimeras of two protein-coding genes; this particular phenomenon has been described in numerous types of cancer.

1.1 Paired-end reads

Paired-end reads can encode information about the presence and nature of structural variants in the genome. Medvedev et al. provide an excellent review in a recent manuscript ([4]). Paired-end reads are generated in a relatively straightforward way. As review, the paired-end sequencing process at a very high level looks like:

1. Genomic DNA is sheared into fragments
2. The sheared fragments of DNA are size-selected (the target size is based on the particular sequencing technology), maniupulated to make them suitable for sequencing (library preparation), and finally placed on the sequencer
3. One end of the pair (the primary read) is read up to the read length (often between 30 and 100 bp, for Illumina)
4. The other end of the pair (the secondary read) is read, again up to the read length
5. The paired ends are aligned to the reference genome resulting in a map of where the ends of the original genomic fragments originated

Note that in point 2 above, the fragments are size-selected. Therefore, if all the DNA that was finally prepared for paired-end sequencing was without structural variation, *the distances between the two paired-ends–the* **insert size***–should recapitulate the distribution of fragment sizes that were selected.* If a pair exhibits an insert size that is an extreme outlier–an outlier pair–it could be evidence for a structural variation.

1.2 Insert size and structural variation

The insert size should be systematically affected by structural variation (Table 1, Figure 1).

Table 1: Effects of structural variants on apparent insert size (the insert size as measured by alignment back to the reference genome).

Structural Variant	Affect on Apparent Insert Size
Small insertion	Decrease
Small deletion	Increase
Large deletion	Increase
Intrachromosomal translocation	Increase
Interchromosomal translocation	Change mate chromosome
Large insertion	Increase or lack of mapped mate

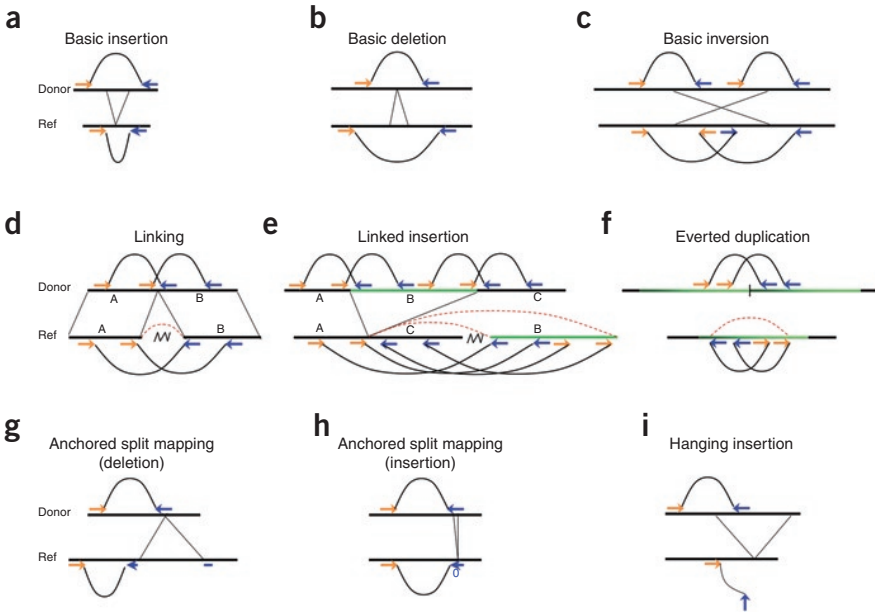

Figure 1: Illustrations of PEM signatures. Mate pairs are sampled from the donor, where they are ordered with opposite orientation (the blue mate follows the orange), and are mapped to the reference (ref). Basic signatures include (a) insertions and (b) deletions, where the mapped distance is different from the insert size, as well as (c) inversions, where the order of the two mates is preserved but one of them changes orientation. (d) The linking signature has several discordant mate pairs with similar mapped distances identifying adjacency in the donor (dashed orange arrows) of two distal segments of the reference. The orientation and order of the mapped mate pairs depends on the orientation and order of the two segments in the reference; here, these are unchanged. (e) A linked insertion signature is composed of two linking signatures and arises when the inserted sequence (green) is copied from another location in the genome. (f) A tandem duplication will create an everted duplication linking signature, with mates mapping out of order but with proper orientations. These mate pairs link the end of the duplicated region to its beginning. (g,h) In the anchored split mapping signature, one mate has a good mapping, whereas the other has a split mapping. For a deletion (g) the prefix and suffix surround the deletion, whereas for an insertion (h) the split read has the prefix and suffix mapped to adjacent locations, while a middle part does not map. (i) When a novel genomic segment is inserted, a hanging insertion signature is created, in which only one of the mates has a good mapping. From [4], figure 1

2 Finding some structural variants

2.1 Data description

In order to keep the exercises manageable and to have a "gold standard" against which to check the results, I have generated an "abnormal" segment of human chromosome 17 that includes several structural variants (see Table 2). The code to generate the artificial sequence is given in this vignette. The MAQ package ([2]) was used to simulate 100000 pairs of reads (so 200000 reads) with insert sizes with mean of 200 and a standard deviation of 20. Some errors were allowed (error rates modeled after typical 36-basepair reads from Illumina). The reads were aligned to the human genome using the BWA software package ([3]) with default settings. The samtools package ([1]) was used to convert the output from BWA into BAM format.

Table 2: Description of insertions and deletions included in the simulated sequence. In addition, a segment of chromosome 17 between 40400000 and 40500000 was "translocated" from its original location to be attached between 40040000 and 40070000. Finally, five copies of the region between 40100000 and 401100000 were attached tandemly to create a copy number gain region.

Variant	Start	End	Size
deletion	40020000	40020020	20
deletion	40022000	40022040	40
deletion	40024000	40024060	60
deletion	40026000	40026080	80
deletion	40028000	40028100	100
deletion	40030000	40030200	200
deletion	40032000	40034000	2000
insertion	40600000	NA	20
insertion	40602000	NA	100
insertion	40604000	NA	1000

```
> a = BSgenome::getSeq(Hsapiens, "chr17", start = 4e+07, width = 1e+06)
> b = paste(sample(c("A", "C", "T", "G"), 10000, replace = TRUE),
+     collapse = "")
> newseq = paste(substr(a, 1, 20000), substr(a, 20020, 22000),
+     substr(a, 22040, 24000), substr(a, 24060, 26000), substr(a,
+         26080, 28000), substr(a, 28100, 30000), substr(a, 30200,
+         32000), substr(a, 34000, 40000), substr(a, 4e+05, 5e+05),
+     substr(a, 70000, 1e+05), substr(a, 100001, 110000), substr(a,
+         100001, 110000), substr(a, 100001, 110000), substr(a,
+         100001, 110000), substr(a, 100001, 110000), substr(a,
+         110001, 399999), substr(a, 500001, 6e+05), substr(b,
```

```
+         1, 20), substr(a, 600001, 602000), substr(b, 1, 100),
+     substr(a, 602001, 604000), substr(b, 1, 1000), substr(a,
+         604001, 1e+06))
> f = file("/tmp/sample.sequence.fa", "w")
> writeLines(">sample", f)
> writeLines(newseq, f)
> close(f)
```

2.2 Data import and preprocessing

Since the data are in BAM format, which is a binary version of the SAM format, the *Rsamtools* package will be used for reading in the data. The SAM format specification provides for storage of paired-end reads. Both ends of the pair are stored, regardless of whether or not they are mapped. In addition, *each* read record has information about the *mapping* of its mate. Since we are typically going to be interested in operating on intervals with data attached to each interval, a natural choice of data structure is the *RangedData* class. A little convenience function, `scanBamToRangedData`, does the necessary reading and minor preprocessing for our BAM file of reads.

```
> require(Rsamtools)
> scanBamToRangedData <- function(...) {
+     tmp = scanBam(...)[[1]]
+     toKeep = TRUE
+     ranges = IRanges(start = tmp$pos[toKeep], width = nchar(tmp$seq[toKeep]))
+     rangeddata = RangedData(ranges, isize = tmp$isize[toKeep],
+         mrnm = tmp$mrnm[toKeep], strand = tmp$strand[toKeep],
+         space = tmp$rname[toKeep], flag = tmp$flag[toKeep])
+     return(rangeddata)
+ }

> fname = system.file("extdata/reads.sorted.bam", package = "StructuralVariant")
> totalRD = scanBamToRangedData(fname, param = ScanBamParam(flag = scanBamFlag(isUnmappedQue
> totalRD

RangedData with 199403 rows and 4 value columns across 25 spaces
         space               ranges |     isize     mrnm   strand      flag
   <character>            <IRanges> | <integer> <factor> <factor> <integer>
1        chr10 [  894815,   894849] |         0    chr14        +       161
2        chr10 [ 4256528,  4256562] |         0    chr17        -        81
3        chr10 [ 5954230,  5954264] |         0    chr17        -       113
4        chr10 [ 7816566,  7816600] |         0    chr17        -        81
5        chr10 [ 8160489,  8160523] |         0    chr17        -       177
6        chr10 [ 8302659,  8302693] |         0    chr17        -       113
7        chr10 [11553012, 11553046] |         0    chr13        -       177
8        chr10 [11561646, 11561680] |         0     chr1        +        65
9        chr10 [11710832, 11710866] |         0    chr15        -       113
```

```
10       chr10 [11770586, 11770620] |         0     chr8        +       161
...
<199393 more rows>
```

The first point to note is that, while all the sequence in the sample sequence file originated from the chr17 human reference sequence, there are mappings to other chromosomes. Since the simulation allows for errors, such errors can induce enough entropy into the original sequence to facilitate alignment to another region of the genome. The effect is small, but non-trivial.

```
> table(space(totalRD))
```

```
  chr1  chr10  chr11  chr12  chr13  chr14  chr15  chr16  chr17  chr18  chr19
   330    157    118    178    104    129    123    140 196093     68    151
  chr2  chr20  chr21  chr22   chr3   chr4   chr5   chr6   chr7   chr8   chr9
   236     80     35     65    211    174    193    156    222    139    132
  chrX   chrY
   146     23
```

Using the information contained in the "flag" column of the *c*olumn, it is possible to get an overview of the mapping characteristics of the pairs; for example, which reads have an unmapped mate?

```
> require(bitops)
> summary(bitAnd(totalRD$flag, 8) == 8)
```

```
   Mode   FALSE    TRUE    NA's
logical  199012     391       0
```

These reads with unmapped mates may (among other possibilities) represent insertions of foreign sequence relative to the reference.

2.3 Clustering outlier read pairs

As discussed above, the insert size is expected to change in systematic ways when structural variants are present. Those read pairs that show a larger or smaller insert size or have an unmapped mate represent "outlier read pairs". For small insertions, there may be bridging reads that show a smaller-than-expected insert size. To test the feasibility of this thinking, I first restrict the *RangedData* object `totalRD` to come from chr17.

```
> chr17RD = totalRD["chr17"]
```

Using the robust estimates of the mean and standard deviation, the median and median absolute deviation (mad), it is possible to choose a range of "normal" insert sizes. Of course, the isize column of the `totalRD` contains this information, but it only makes sense to compute on those reads that have a mapped mate and are on the same chromosome.

```
> median(abs(totalRD$isize), na.rm = TRUE)

[1] 200

> mad(abs(totalRD$isize), na.rm = TRUE)

[1] 20.7564
```

Note that these agree quite well with the parameters from the simulation. So, filtering the reads to those with isizes smaller than 137 (approximately μ - 3σ), we can find potential pairs associated with small insertions. More specifically, if there is clustering of reads that show smaller-than-expected insert sizes, there may be evidence for a deletion in the vicinity of the clustering. To calculate clustering, it is fairly straightforward to use ideas similar to those demonstrated for ChIP-seq analysis; the little function, `extendReadsRD` takes as input a *RangedData* object and returns a modified *RangedData* object with the start and end values adjusted to have the specified width after accounting for the read strand (which *must* be in the *RangedData* object in the strand column).

```
> extendReadsRD <- function(rangeddata, width = 100) {
+     s = start(rangeddata)
+     e = end(rangeddata)
+     w = width - width(rangeddata)
+     strand = rangeddata$strand
+     s[strand == "-"] = s[strand == "-"] - w[strand == "-"]
+     e[strand == "+"] = e[strand == "+"] + w[strand == "+"]
+     start(rangeddata) <- s
+     end(rangeddata) <- e
+     return(rangeddata)
+ }
```

To look for the reads that have a mapped mate that also maps to chr17 and have insert sizes less than 137 bases long, we use the following incantation. Basically, the first command finds reads that have a mapped mate where the mate is mapped to chromosome 17 and the insert size is smaller than expected. The reads are then extended to half of the expected insert size. Coverage is calculated on the extended reads and finally the coverage vector is sliced (thresholded) to find regions that are somewhat enriched in short-insert-size pairs. The last step would require some statistical modeling to determine the best threshold; here, a convenient threshold was chosen empirically.

```
> smallInsertRD = chr17RD[(bitAnd(chr17RD$flag, 8) != 8) & (chr17RD$mrnm ==
+     "chr17") & (abs(chr17RD$isize) < 137), ]
> smallInsertRD = extendReadsRD(smallInsertRD, 135)
> cvgSI = coverage(smallInsertRD)
> sliceSI = slice(cvgSI, 10)
```

Reads with insert sizes larger than expected (263 bp, approximately $\mu + 3\sigma$) may be associated with regions of the genome that are physically attached but on the reference sequence are actually far apart. Deletions and translocations would be expected to show reads that have larger-than-expected insert sizes. Again, we find these reads and then cluster them to find putative regions of deletion or translocation. Using an analogous set of functions to short-insert-size pairs, it is possible to cluster the long-insert-size pairs.

```
> largeInsertRD = chr17RD[(bitAnd(chr17RD$flag, 8) != 8) & (chr17RD$mrnm ==
+     "chr17") & (abs(chr17RD$isize) > 263), ]
> largeInsertRD = extendReadsRD(largeInsertRD, 135)
> cvgLI = coverage(largeInsertRD)
> sliceLI = slice(cvgLI, 4)
```

Finally, it is possible to do the same with reads with unmapped mates.

```
> unmappedRD = chr17RD[(bitAnd(chr17RD$flag, 8) == 8), ]
> unmappedRD = extendReadsRD(unmappedRD, 135)
> cvgUM = coverage(unmappedRD)
> sliceUM = slice(cvgUM, 5)
```

2.4 Interpretation of results

Currently, there is little infrastructure in place to facilitate the interpretation of the results of clustering outlier read pairs. Using the large-insert-pairs clustering result as an example is still instructive, though, for gaining insight into the structural variation present. Examining the text output of the sliced coverage is informative, but leaves a sour feeling in the lack of insight it provides.

```
> sliceLI[["chr17"]]

Views on a 78092744-length Rle subject

views:
        start      end width
 [1] 40021994 40021997     4 [4 4 4 4]
 [2] 40022013 40022013     1 [4]
 [3] 40023968 40024137   170 [4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 ...]
 [4] 40025888 40026055   168 [4 4 4 4 4 5 5 5 5 5 5 5 5 5 5 5 5 5 6 7 7 7 ...]
 [5] 40028049 40028225   177 [4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 5 5 5 5 5 5 5 5 ...]
 [6] 40029880 40030011   132 [4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 5 5 5 5 5 5 5 5 ...]
 [7] 40030161 40030275   115 [4 4 4 4 4 4 4 4 4 4 4 4 4 5 5 5 5 5 5 5 5 5 5 ...]
 [8] 40031937 40032066   130 [4 4 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 6 6 ...]
 [9] 40034005 40034119   115 [4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 5 5 5 5 5 ...]
[10] 40039872 40040055   184 [ 4  4  4  4  4  4  5  5  6  6  7  7  7  7 ...]
[11] 40069925 40070083   159 [ 4  4  4  4  4  5  5  5  5  5  5  6  6  6 ...]
[12] 40099911 40100155   245 [ 4  6  6  6  6  6  6  6  6  6  6  6  6  6 ...]
```

```
[13] 40109839 40110096   258 [ 4  4  4  4  4  4  4  4  4  5  5  6  7  7 ...]
[14] 40399874 40400150   277 [4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 5 ...]
[15] 40417415 40417487    73 [4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 ...]
[16] 40417813 40417862    50 [4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 ...]
[17] 40499863 40500143   281 [4 4 4 4 5 5 5 5 5 5 5 6 7 7 7 7 7 7 7 7 8 8 ...]
```

However, some simple plots can be very informative about the fine- and large-scale structure of this artificial segment of chromosome 17. A little plotting function, `plotSV`, generates a plot showing the regions captured by the clustering process and then overlays the connections as encoded in the paired-end reads that overlap with the clusters. The resulting plots can be instructive (indeed, they led me to a bug in the generation of the original sequence).

```
> plotSV <- function(rangeddata, cvgslice, space = "chr17", ...) {
+     xl = min(start(rangeddata)[space(rangeddata) == space])
+     xr = max(end(rangeddata)[space(rangeddata) == space])
+     par(las = 2)
+     plot(c(xl, xr), c(0, 3), pch = "", xlab = "genomic coordinates",
+         ylab = "", axes = FALSE, ...)
+     for (i in 1:length(cvgslice[[space]])) {
+         rect(start(cvgslice[[space]])[i], 0, end(cvgslice[[space]])[i],
+             0.5, col = "red", border = "red")
+     }
+     subrd = rangeddata[rangeddata %in% RangedData(IRanges(start = start(cvgslice[[space]]),
+         end = end(cvgslice[[space]])), space = rep(space, length(cvgslice[[space]]))),
+         ]
+     for (i in 1:length(start(subrd))) {
+         strandq = ifelse(bitAnd(subrd$flag[i], 16) == 16, "-",
+             "+")
+         strandmate = ifelse(bitAnd(subrd$flag[i], 32) == 32,
+             "-", "+")
+         if (strandq == "+" & strandmate == "-")
+             col = "red"
+         if (strandq == "-" & strandmate == "+")
+             col = "green"
+         if (strandq == "+" & strandmate == "+")
+             col = "blue"
+         if (strandq == "-" & strandmate == "-")
+             col = "orange"
+         lines(list(x = c(start(subrd)[i], start(subrd)[i] + subrd$isize[i]),
+             y = c(0.5, 2.5)), col = col)
+     }
+     axis(side = 1)
+     box()
+ }
```

Using this function, it is possible to generate some interesting plots. Figures 2 and 3 show the entire 1MB region and a zoomed-in view of the region with a number of small deletions, respectively.

```
> pdf("fullRegionLI.pdf", width = 9, height = 7)
> plotSV(chr17RD, sliceLI, xlim = c(4e+07, 4.1e+07))
> dev.off()
```

```
null device
          1
```

```
> pdf("zoomDeletionsLI.pdf", width = 9, height = 7)
> plotSV(chr17RD, sliceLI, xlim = c(40020000, 40045000))
> dev.off()
```

```
null device
          1
```

3 Conclusions and future work

While multiple software packages for finding structural variants exist (see table 1 in [4]), the R and Bioconductor tools offer an opportunity to build on existing infrastructure for short-read data analysis, to experiment with methodologies for extending existing methods, and to capitalize on the extremely powerful graphics and data handling capabilities inherent in R. There is a need to improve handling of paired-end reads (and related mate-pair reads) in a memory- and computationally-efficient manner. Also, clustering of reads could be done using other models or statistical or computational techniques. After the processing of reads down to clusters is complete, a mechanism to help to truly understand the variants that lead to those clustering results, such as a graph-theoretic approach to linking clusters, needs to be developed. Finally, and perhaps most difficult, is to come to some new biologic understanding or knowledge (are fusion genes formed, regulatory regions disrupted, etc.) based on the results.

4 sessionInfo()

```
> toLatex(sessionInfo())
```

- R version 2.11.0 Under development (unstable) (2009-11-13 r50424), `i386-apple-darwin10.2.0`
- Locale: `en_US/en_US/C/C/en_US/en_US`
- Base packages: base, datasets, graphics, grDevices, methods, stats, utils
- Other packages: Biostrings 2.15.6, bitops 1.0-4.1, BSgenome 1.15.2, BSgenome.Hsapiens.UCSC.hg18 1.3.15, IRanges 1.5.6, Rsamtools 0.1.19
- Loaded via a namespace (and not attached): Biobase 2.7.0

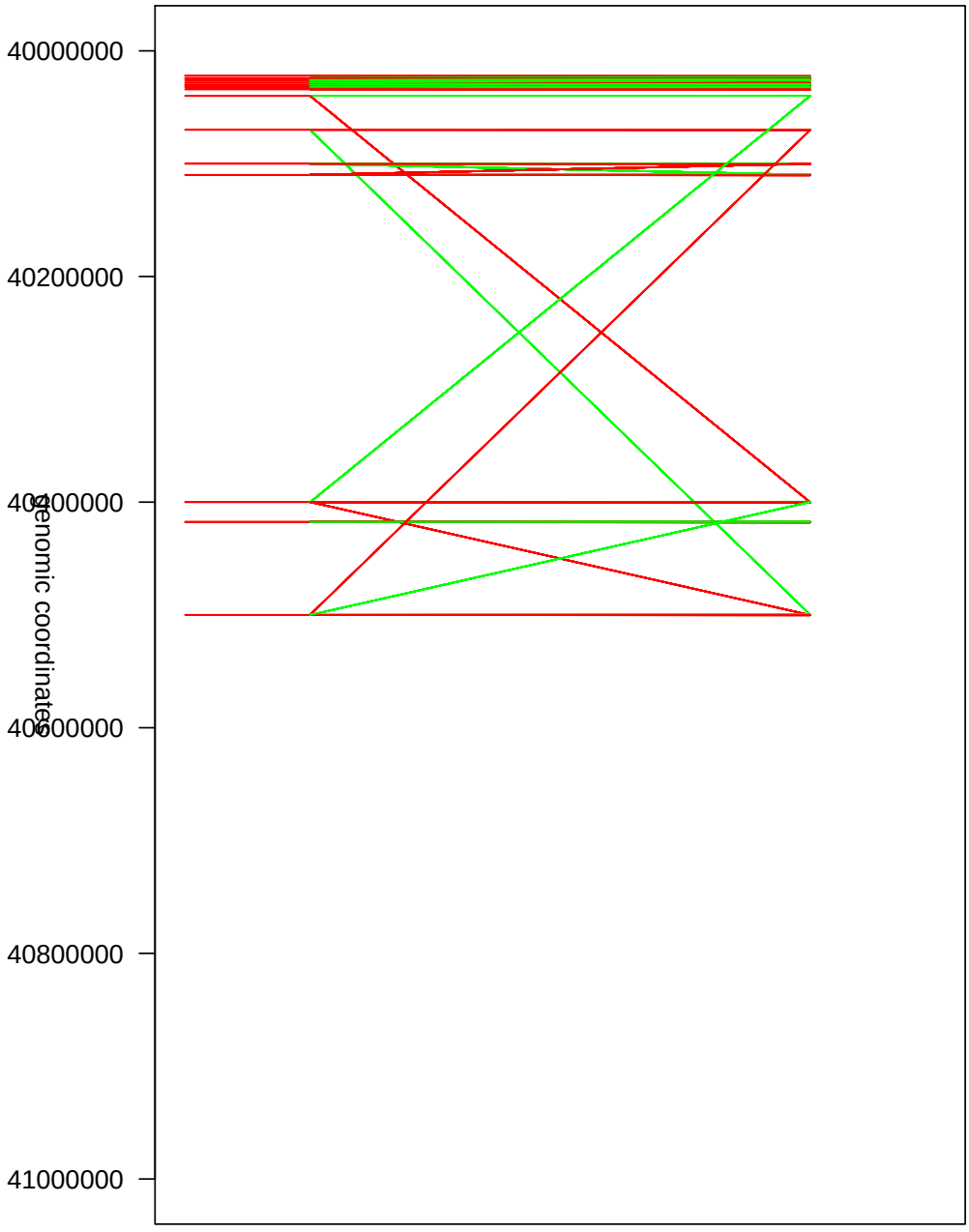

Figure 2: A view of the the entire 1MB region showing some large-scale rearrangements. The red bars represent the locations of the clusters of abnormal read pairs. The lines lines demonstrate how the reads that map to the locations covered by the clusters connect to their mates. The colors of the lines correspond to the combinations of strand information in the two ends of the pair; read means the first pair is + and the second is - strand while green is the opposite. All the lines are drawn from the first pair location on the lower part of the plot to the second pair of the read on the upper part of the plot. A translocation is clearly visible as is the region of copy-number gain (which is a

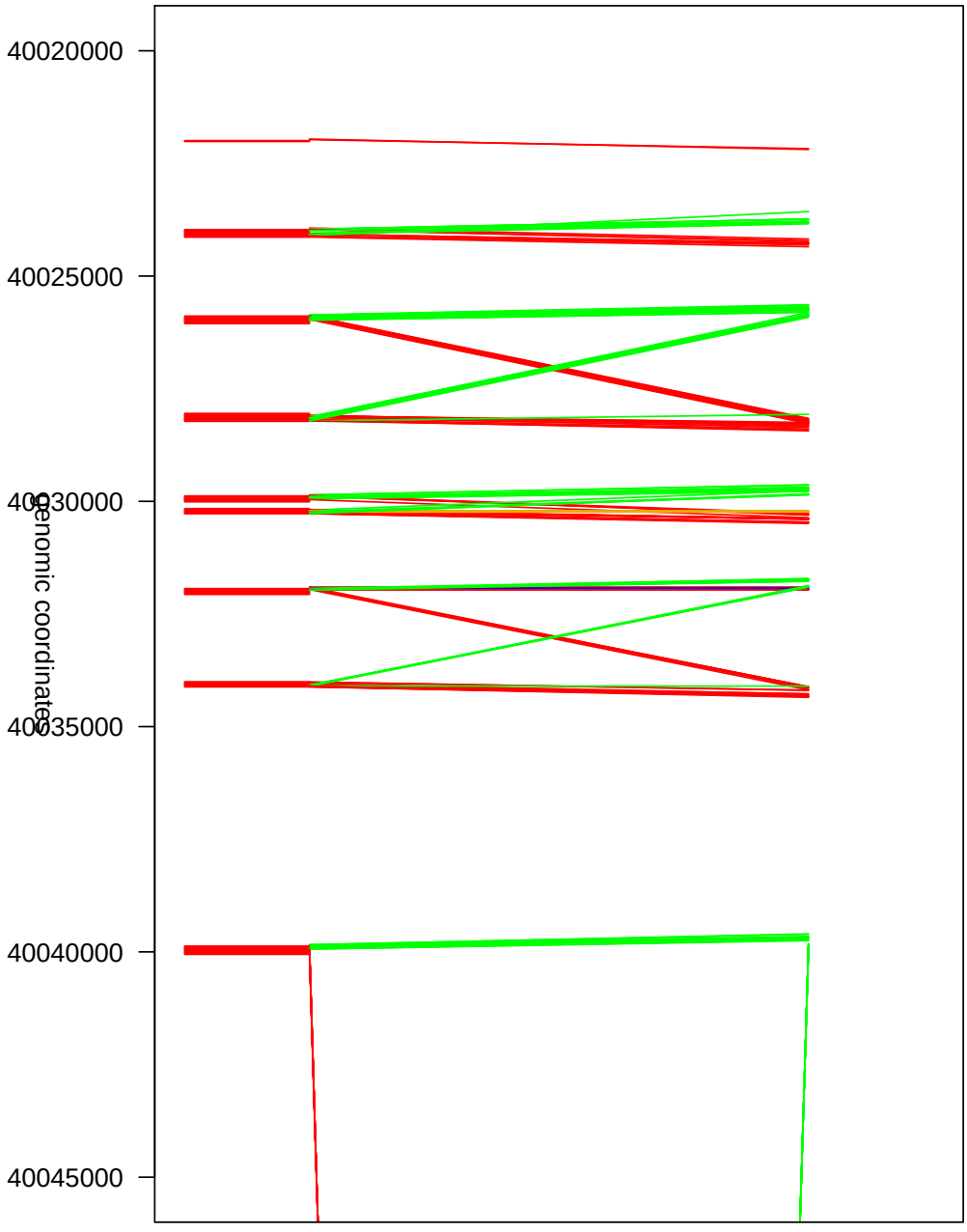

Figure 3: A zoomed-in view of some small deletions. The markup of the figure is as for Figure 2. Refer to Table 2 for details of the locations of the deletions. Note that there is what appears to be a "extra" large deletion–this was a mistake in the code for generating the artificial sequence that I caught only after making the plots.

References

[1] H Li, B Handsaker, A Wysoker, T Fennell, J Ruan, N Homer, G Marth, G Abecasis, R Durbin, and 1000 Genome Project Data Processing Subgroup. The sequence alignment/map format and samtools. *Bioinformatics*, 25(16):2078–2079, Aug 2009.

[2] H Li, J Ruan, and R Durbin. Mapping short dna sequencing reads and calling variants using mapping quality scores. *Genome Res*, 18(11):1851–1858, Nov 2008.

[3] Heng Li and Richard Durbin. Fast and accurate short read alignment with Burrows-Wheeler transform. *Bioinformatics*, 25(14):1754–1760, 2009.

[4] P Medvedev, M Stanciu, and M Brudno. Computational methods for discovering structural variation with next-generation sequencing. *Nat Methods*, 6(11 Suppl):13–20, Nov 2009.