An Introduction to *intansv*

Wen Yao

April 16, 2015

Contents

1 Introduction

Currently, dozens of programs have been developped to predict structural variations (SV) utilizing next-generation sequencing data. However, the prediction of multiple methods have to be integrated to get relatively reliable results (Zichner et al., 2013). The *intansv* package is designed for integrating results of different methods, annotating effects caused by SVs to genes and its elements, displaying the genomic distribution of SVs and visualizing SVs in specific genomic region. In this vignette, we will rely on a simple, illustrative example dataset to explain the usage of *intansv*.

The *intansv* package is available at bioconductor.org and can be downloaded via `biocLite`:

```
> source("http://bioconductor.org/biocLite.R")
> biocLite("intansv")

> library(intansv)
```

2 Usage of *intansv* with example dataset

The example dataset was obtained by running seven different programs with the alignment results of a public available NGS dataset (NCBI SRA accession number SRA012177) to the rice reference genome (`http://rice.plantbiology.msu.edu/`). These seven programs including BreakDancer (Chen et al., 2009), Pindel (Ye et al., 2009), CNVnator (Abyzov et al., 2011), DELLY (Rausch et al., 2012), Lumpy (Ryan M. et al., 2012), softSearch (Steven N. et al., 2013), and SVseq2 (Zhang et al., 2012) were run with default parameters. The predicted SVs of chromosomes, *chr05* and *chr10*, were provided in the example dataset.

2.1 Read in predictions of different programs

Most of the predictions of programs are tedious and need to be formatted for further analysis. *intansv* provides functions to read in the predictions of five popular programs, BreakDancer, Pindel, CNVnator, DELLY and SVseq2. During this process, the SV predictions with low quality would be filtered out and overlapped predictions would be resolved.

We begin our discussion by reading in some example data.

```
> breakdancer <- readBreakDancer(system.file("extdata/ZS97.breakdancer.sv",
+                                 package="intansv"))
> str(breakdancer)

List of 2
 $ del:'data.frame':        1007 obs. of  4 variables:
  ..$ chromosome: chr [1:1007] "chr05" "chr05" "chr05" "chr05" ...
  ..$ pos1      : num [1:1007] 65586 86442 120288 153694 201845 ...
  ..$ pos2      : num [1:1007] 65938 87430 127670 153801 201959 ...
  ..$ size      : num [1:1007] 353 988 7382 107 114 ...
 $ inv:'data.frame':        17 obs. of  4 variables:
  ..$ chromosome: chr [1:17] "chr05" "chr05" "chr05" "chr05" ...
  ..$ pos1      : num [1:17] 1291574 6942451 12014581 15092428 18770962 ...
  ..$ pos2      : num [1:17] 1291678 6944637 12015915 15157500 18771414 ...
  ..$ size      : num [1:17] 105 2186 1334 65072 452 ...
 - attr(*, "method")= chr "BreakDancer"
```

BreakDancer is able to predict deletions and inversions. The prediction of all kind of SVs were provided as a single file by BreakDancer. You can feed this file to `readBreakdancer` and it will do all the tedious work for you.

```
> cnvnator <- readCnvnator(system.file("extdata/cnvnator",package="intansv"))
> str(cnvnator)

List of 2
 $ del:'data.frame':        369 obs. of  4 variables:
  ..$ chromosome: chr [1:369] "chr05" "chr05" "chr05" "chr05" ...
  ..$ pos1      : num [1:369] 104101 230401 348301 1089301 1524301 ...
  ..$ pos2      : num [1:369] 110700 248400 350100 1103100 1529700 ...
  ..$ size      : num [1:369] 6599 17999 1799 13799 5399 ...
 $ dup:'data.frame':        113 obs. of  4 variables:
  ..$ chromosome: chr [1:113] "chr05" "chr05" "chr05" "chr05" ...
  ..$ pos1      : num [1:113] 405001 973201 1346401 5036101 6419101 ...
  ..$ pos2      : num [1:113] 408900 977700 1359000 5055600 6423300 ...
  ..$ size      : num [1:113] 3899 4499 12599 19499 4199 ...
 - attr(*, "method")= chr "CNVnator"
```

CNVnator is able to predict deletions and duplications. The final output of CNVnator usually contains several files and each file is the output for a specific chromosome. You should put all these files in the same directory and feed the path of this directory to `readCnvnator`. Then it will do all the jobs for you. However, the directory given to `readCnvnator` should only contain the final output files of CNVnator. See the example dataset for more details.

```
> svseq <- readSvseq(system.file("extdata/svseq2",package="intansv"))
> str(svseq)
```

```
List of 1
 $ del:'data.frame':        135 obs. of  4 variables:
  ..$ chromosome: chr [1:135] "chr10" "chr10" "chr10" "chr10" ...
  ..$ pos1      : num [1:135] 84242 93888 288998 316662 337456 ...
  ..$ pos2      : num [1:135] 87392 94156 289244 318112 337668 ...
  ..$ size      : num [1:135] 3150 268 246 1450 212 ...
 - attr(*, "method")= chr "SVseq2"
```

SVseq2 is able to predict deletions. The final output of SVseq2 contains several files and each file is the output for a specific chromosome. What's more, different kind of SVs were written to different files. You should put all these files in the same directory and feed the path of this directory to `readSvseq`. However, the files contain the predicted deletions in this directory should be named with suffix of *.del*. See the example dataset for more details.

```
> delly <- readDelly(system.file("extdata/delly",package="intansv"))
> str(delly)
```

```
List of 3
 $ del:'data.frame':        1208 obs. of  4 variables:
  ..$ chromosome: chr [1:1208] "chr05" "chr05" "chr05" "chr05" ...
  ..$ pos1      : int [1:1208] 65580 86457 153671 172303 201850 208219 226809 230526 276053 276807 ...
  ..$ pos2      : num [1:1208] 65949 87419 153829 172439 201959 ...
  ..$ size      : num [1:1208] 369 962 158 136 109 358 104 128 129 143 ...
 $ dup:'data.frame':        29 obs. of  4 variables:
  ..$ chromosome: chr [1:29] "chr05" "chr05" "chr05" "chr05" ...
  ..$ pos1      : int [1:29] 120788 910613 1133978 1614997 5696565 5757285 14472343 17377712 21630340 225
  ..$ pos2      : num [1:29] 128008 911328 1136610 1616026 5699335 ...
  ..$ size      : num [1:29] 7220 715 2632 1029 2770 ...
 $ inv:'data.frame':        23 obs. of  4 variables:
  ..$ chromosome: chr [1:23] "chr05" "chr05" "chr05" "chr05" ...
  ..$ pos1      : int [1:23] 5550226 6942367 8700896 10402721 11062213 12014418 13291663 15092334 1873713
  ..$ pos2      : num [1:23] 5550352 6944731 8702504 10402842 11631416 ...
  ..$ size      : num [1:23] 126 2364 1608 121 569203 ...
 - attr(*, "method")= chr "DELLY"
```

DELLY is able to predict deletions, inversions and duplications. The final prediction of different kind of SVs given by DELLY were written to different files. DELLY v0.6.1 outputs vcf files now. You should put all these files in the same directory and feed the path of this directory to `readDelly`. See the example dataset for more details.

```
> pindel <- readPindel(system.file("extdata/pindel",package="intansv"))
> str(pindel)
```

```
List of 3
 $ del:'data.frame':        677 obs. of  4 variables:
  ..$ chromosome: chr [1:677] "chr05" "chr05" "chr05" "chr05" ...
  ..$ pos1      : num [1:677] 76477 86438 120431 334346 366428 ...
  ..$ pos2      : num [1:677] 76913 87419 127842 334613 367941 ...
  ..$ size      : num [1:677] 435 980 7410 266 1512 ...
 $ inv:'data.frame':        52 obs. of  4 variables:
  ..$ chromosome: chr [1:52] "chr05" "chr05" "chr05" "chr05" ...
  ..$ pos1      : num [1:52] 777580 1385804 2580617 3062624 6298321 ...
```

```
  ..$ pos2      : num [1:52] 777810 1390029 2583765 3062746 6311292 ...
  ..$ size      : num [1:52] 229 4224 3147 121 12970 ...
 $ dup:'data.frame':        68 obs. of  4 variables:
  ..$ chromosome: chr [1:68] "chr05" "chr05" "chr05" "chr05" ...
  ..$ pos1      : num [1:68] 97139 120110 910586 957199 1018140 ...
  ..$ pos2      : num [1:68] 97308 127734 911325 957324 1018273 ...
  ..$ size      : num [1:68] 168 7623 738 124 132 ...
 - attr(*, "method")= chr "Pindel"
```

Pindel is able to predict deletions, inversions and duplications. The final prediction for different chromosomes given by Pindel were written to different files. And different kind of SVs were written to different files. You should put all these files in the same directory and feed the path of this directory to `readPindel`. However, the files contain the predicted deletions, inversions and duplication in this directory should be named with suffix of *_D*, *_INV* and *_TD* respectively. See the example dataset for more details.

```
> lumpy <- readLumpy(system.file("extdata/ZS97.lumpy.pesr.bedpe",
+                     package="intansv"))
> str(lumpy)

List of 3
 $ del:'data.frame':        1047 obs. of  4 variables:
  ..$ chromosome: chr [1:1047] "chr05" "chr05" "chr05" "chr05" ...
  ..$ pos1      : num [1:1047] 65582 86450 201842 208220 230514 ...
  ..$ pos2      : num [1:1047] 65936 87424 201976 208548 230656 ...
  ..$ size      : num [1:1047] 355 975 134 329 142 ...
 $ dup:'data.frame':        37 obs. of  4 variables:
  ..$ chromosome: chr [1:37] "chr05" "chr05" "chr05" "chr05" ...
  ..$ pos1      : num [1:37] 910606 957196 1133966 1615004 5757302 ...
  ..$ pos2      : num [1:37] 911322 957324 1136616 1616026 5758086 ...
  ..$ size      : num [1:37] 717 127 2650 1023 784 ...
 $ inv:'data.frame':        47 obs. of  4 variables:
  ..$ chromosome: chr [1:47] "chr05" "chr05" "chr05" "chr05" ...
  ..$ pos1      : num [1:47] 2088840 3277686 5899568 6372408 6613076 ...
  ..$ pos2      : num [1:47] 2089006 3278456 5899762 6453738 6645398 ...
  ..$ size      : num [1:47] 166 770 194 81331 32323 ...
 - attr(*, "method")= chr "Lumpy"
```

Lumpy is able to predict deletions, inversions and duplications. The prediction of all kind of SVs were provided as a single file by Lumpy. You can feed this file to `readLumpy` and it will do all the tedious work for you.

```
>  softSearch <- readSoftSearch(system.file("extdata/ZS97.softsearch",package="intansv"))
> str(softSearch)

List of 3
 $ del:'data.frame':        47 obs. of  4 variables:
  ..$ chromosome: chr [1:47] "chr05" "chr05" "chr05" "chr05" ...
  ..$ pos1      : num [1:47] 366291 1491176 1802871 2986918 14116786 ...
  ..$ pos2      : num [1:47] 367937 1491595 1803476 2987435 14129776 ...
  ..$ size      : num [1:47] 1646 419 605 517 12990 ...
 $ dup: NULL
 $ inv:'data.frame':        4 obs. of  4 variables:
```

```
  ..$ chromosome: chr [1:4] "chr05" "chr05" "chr10" "chr10"
  ..$ pos1      : num [1:4] 3106094 18770855 2460956 7020583
  ..$ pos2      : num [1:4] 3106380 19669803 2512674 7021271
  ..$ size      : num [1:4] 286 898948 51718 688
 - attr(*, "method")= chr "softSearch"
```

softSearch is able to predict deletions, inversions and duplications. The prediction of all kind of SVs were provided as a single file by softSearch. You can feed this file to `readSoftSearch` and it will do all the tedious work for you.

The results of these seven programs have now been read into R and storaged as R data structure of lists with different compoents representing different types of SVs. We only care about three types of SVs, deletions, duplications and inversions.

2.2 Integrate predictions of different programs

To get more reliable results, we need to integrate the predictions of different programs. See our paper (Yao and Xie, 2013) for more details on how *intansv* integrate the predictions of different programs.

```
> sv_all_methods <- methodsMerge(breakdancer,pindel,cnvnator,delly,svseq)
> str(sv_all_methods)
```

```
List of 3
 $ del:'data.frame':        947 obs. of  8 variables:
  ..$ chromosome : chr [1:947] "chr05" "chr05" "chr05" "chr05" ...
  ..$ pos1       : num [1:947] 65583 86446 120360 201848 208218 ...
  ..$ pos2       : num [1:947] 65944 87423 127756 201959 208574 ...
  ..$ BreakDancer: chr [1:947] "Y" "Y" "Y" "Y" ...
  ..$ Pindel     : chr [1:947] "N" "Y" "Y" "N" ...
  ..$ CNVnator   : chr [1:947] "N" "N" "N" "N" ...
  ..$ DELLY      : chr [1:947] "Y" "Y" "N" "Y" ...
  ..$ SVseq2     : chr [1:947] "N" "N" "N" "N" ...
 $ dup:'data.frame':        11 obs. of  8 variables:
  ..$ chromosome : chr [1:11] "chr05" "chr05" "chr05" "chr05" ...
  ..$ pos1       : num [1:11] 120449 910600 1614997 5696556 5757284 ...
  ..$ pos2       : num [1:11] 127871 911326 1616026 5699328 5758092 ...
  ..$ BreakDancer: chr [1:11] "N" "N" "N" "N" ...
  ..$ Pindel     : chr [1:11] "Y" "Y" "Y" "Y" ...
  ..$ CNVnator   : chr [1:11] "N" "N" "N" "N" ...
  ..$ DELLY      : chr [1:11] "Y" "Y" "Y" "Y" ...
  ..$ SVseq2     : chr [1:11] "N" "N" "N" "N" ...
 $ inv:'data.frame':        12 obs. of  8 variables:
  ..$ chromosome : chr [1:12] "chr05" "chr05" "chr05" "chr05" ...
  ..$ pos1       : num [1:12] 6942409 12014500 15090018 29431740 2152662 ...
  ..$ pos2       : num [1:12] 6944684 12015915 15158197 29432418 2240864 ...
  ..$ BreakDancer: chr [1:12] "Y" "Y" "Y" "N" ...
  ..$ Pindel     : chr [1:12] "N" "N" "Y" "Y" ...
  ..$ CNVnator   : chr [1:12] "N" "N" "N" "N" ...
  ..$ DELLY      : chr [1:12] "Y" "Y" "Y" "Y" ...
  ..$ SVseq2     : chr [1:12] "N" "N" "N" "N" ...
```

The integrated results contain 897 deletions, 13 duplications and 12 inversions. However, it's not necessary for you to feed *intansv* the output of all five programs. The following example shows the integration of four programs: BreakDancer, CNVnator, DELLY and SVseq2.

```
> sv_all_methods.nopindel <- methodsMerge(breakdancer,cnvnator,delly,svseq)
> str(sv_all_methods.nopindel)

List of 3
 $ del:'data.frame':        901 obs. of  7 variables:
  ..$ chromosome : chr [1:901] "chr05" "chr05" "chr05" "chr05" ...
  ..$ pos1       : num [1:901] 65583 86450 201848 208218 230523 ...
  ..$ pos2       : num [1:901] 65944 87424 201959 208574 230655 ...
  ..$ BreakDancer: chr [1:901] "Y" "Y" "Y" "Y" ...
  ..$ CNVnator   : chr [1:901] "N" "N" "N" "N" ...
  ..$ DELLY      : chr [1:901] "Y" "Y" "Y" "Y" ...
  ..$ SVseq2     : chr [1:901] "N" "N" "N" "N" ...
 $ dup:'data.frame':        1 obs. of  7 variables:
  ..$ chromosome : chr "chr05"
  ..$ pos1       : num 29657896
  ..$ pos2       : num 29662746
  ..$ BreakDancer: chr "N"
  ..$ CNVnator   : chr "Y"
  ..$ DELLY      : chr "Y"
  ..$ SVseq2     : chr "N"
 $ inv:'data.frame':        10 obs. of  7 variables:
  ..$ chromosome : chr [1:10] "chr05" "chr05" "chr05" "chr10" ...
  ..$ pos1       : num [1:10] 6942409 12014500 15092381 2152662 2460999 ...
  ..$ pos2       : num [1:10] 6944684 12015915 15157552 2240864 2512789 ...
  ..$ BreakDancer: chr [1:10] "Y" "Y" "Y" "Y" ...
  ..$ CNVnator   : chr [1:10] "N" "N" "N" "N" ...
  ..$ DELLY      : chr [1:10] "Y" "Y" "Y" "Y" ...
  ..$ SVseq2     : chr [1:10] "N" "N" "N" "N" ...
```

intansv is also able to integrate SV predictions of other programs. However, predictions of other programs should be provided as a data frame with six columns:

- chromosome the chromosome ID of a SV
- pos1 the start coordinate of a SV
- pos2 the end coordinate of a SV
- size the size of a SV
- type the type of a SV
- methods the program used to get this SV prediction

2.3 Annotate the effects of SVs

To annotate the effects of SVs to genes, we need the genomic annotation file, which is usually a *.gff3* file. This file could be read into R by the *rtracklayer* package and storaged as genomic ranges data. An example genomic annotation file has been storaged in the example dataset of *intansv*.

The follow code shows how to read a *.gff3* file into R.

```
> library(rtracklayer)
> msu_gff_v7 <- import.gff(system.file("extdata/chr05_chr10.gff3", package="intansv"),
+                          asRangedData=FALSE)
> head(msu_gff_v7)
```

```
GRanges object with 6 ranges and 8 metadata columns:
      seqnames       ranges strand |     source     type     score
         <Rle>    <IRanges>  <Rle> |   <factor> <factor> <numeric>
  [1]    chr05 [4003, 4356]      + | MSU_osa1r7     gene      <NA>
  [2]    chr05 [4003, 4356]      + | MSU_osa1r7     mRNA      <NA>
  [3]    chr05 [4003, 4356]      + | MSU_osa1r7     exon      <NA>
  [4]    chr05 [4003, 4356]      + | MSU_osa1r7      CDS      <NA>
  [5]    chr05 [6935, 9099]      + | MSU_osa1r7     gene      <NA>
  [6]    chr05 [6935, 9099]      + | MSU_osa1r7     mRNA      <NA>
          phase                        ID             Name
      <integer>             <character>      <character>
  [1]      <NA>          LOC_Os05g00988   LOC_Os05g00988
  [2]      <NA>        LOC_Os05g00988.1 LOC_Os05g00988.1
  [3]      <NA> LOC_Os05g00988.1:exon_1             <NA>
  [4]      <NA>  LOC_Os05g00988.1:cds_1             <NA>
  [5]      <NA>          LOC_Os05g00990   LOC_Os05g00990
  [6]      <NA>        LOC_Os05g00990.1 LOC_Os05g00990.1
                                                      Note
                                                <CharacterList>
  [1]                                    hypothetical protein
  [2]
  [3]
  [4]
  [5] retrotransposon protein, putative, unclassified, expressed
  [6]
               Parent
       <CharacterList>
  [1]
  [2]   LOC_Os05g00988
  [3] LOC_Os05g00988.1
  [4] LOC_Os05g00988.1
  [5]
  [6]   LOC_Os05g00990
  -------
  seqinfo: 2 sequences from an unspecified genome; no seqlengths

> load(system.file("extdata/genome.anno.RData",package="intansv"))
> sv_all_methods.anno <- llply(sv_all_methods,svAnnotation,
+                              genomeAnnotation=msu_gff_v7)
> names(sv_all_methods.anno)

[1] "del" "dup" "inv"

> head(sv_all_methods.anno$del)

  chromosome  pos1  pos2 overlap      annotation
1      chr05 65583 65944   0.066            gene
2      chr05 65583 65944   0.094            mRNA
3      chr05 65583 65944   0.094            mRNA
4      chr05 65583 65944   0.077            mRNA
5      chr05 65583 65944   0.317            exon
6      chr05 65583 65944   0.377 three_prime_UTR
```

```
                        id           parent
1          LOC_Os05g01040   LOC_Os05g01040
2        LOC_Os05g01040.1   LOC_Os05g01040
3        LOC_Os05g01040.2   LOC_Os05g01040
4        LOC_Os05g01040.3   LOC_Os05g01040
5 LOC_Os05g01040.1:exon_8 LOC_Os05g01040.1
6  LOC_Os05g01040.1:utr_1 LOC_Os05g01040.1
```

Since the function `svAnnotation` only accept structural variations of the data frame format, we need to apply this function to each component of *sv_all_methods*, which is a list.

2.4 Display the genomic distribution of SVs

Now, let's get a genomic view of SVs. We first split chromosomes into windows of 1 Mb and then display the number of SVs in each window as circular barplot. We also need the length of each chromosome, which was storaged in the example dataset of *intansv*.

```
> genome
```

```
GRanges object with 2 ranges and 0 metadata columns:
      seqnames        ranges strand
         <Rle>     <IRanges>  <Rle>
  [1]    chr05 [1, 29958434]      *
  [2]    chr10 [1, 23207287]      *
  -------
  seqinfo: 2 sequences from an unspecified genome
```

```
> plotChromosome(genome,sv_all_methods,1000000)
```

2.5 Visualize SVs in specific genomic region

We could also visualize SVs in specific genomic region. Here, we also need the genomic annotation file (named with suffix *.gff3* or *.gff*).

```
> head(msu_gff_v7,n=3)
```

```
GRanges object with 3 ranges and 8 metadata columns:
      seqnames       ranges strand |     source     type     score
         <Rle>    <IRanges>  <Rle> |   <factor> <factor> <numeric>
  [1]    chr05 [4003, 4356]      + | MSU_osa1r7     gene      <NA>
  [2]    chr05 [4003, 4356]      + | MSU_osa1r7     mRNA      <NA>
  [3]    chr05 [4003, 4356]      + | MSU_osa1r7     exon      <NA>
          phase                      ID             Name
      <integer>             <character>      <character>
  [1]      <NA>          LOC_Os05g00988   LOC_Os05g00988
  [2]      <NA>        LOC_Os05g00988.1 LOC_Os05g00988.1
  [3]      <NA> LOC_Os05g00988.1:exon_1             <NA>
                     Note           Parent
          <CharacterList>  <CharacterList>
  [1] hypothetical protein
  [2]                        LOC_Os05g00988
  [3]                      LOC_Os05g00988.1
  -------
  seqinfo: 2 sequences from an unspecified genome; no seqlengths
```

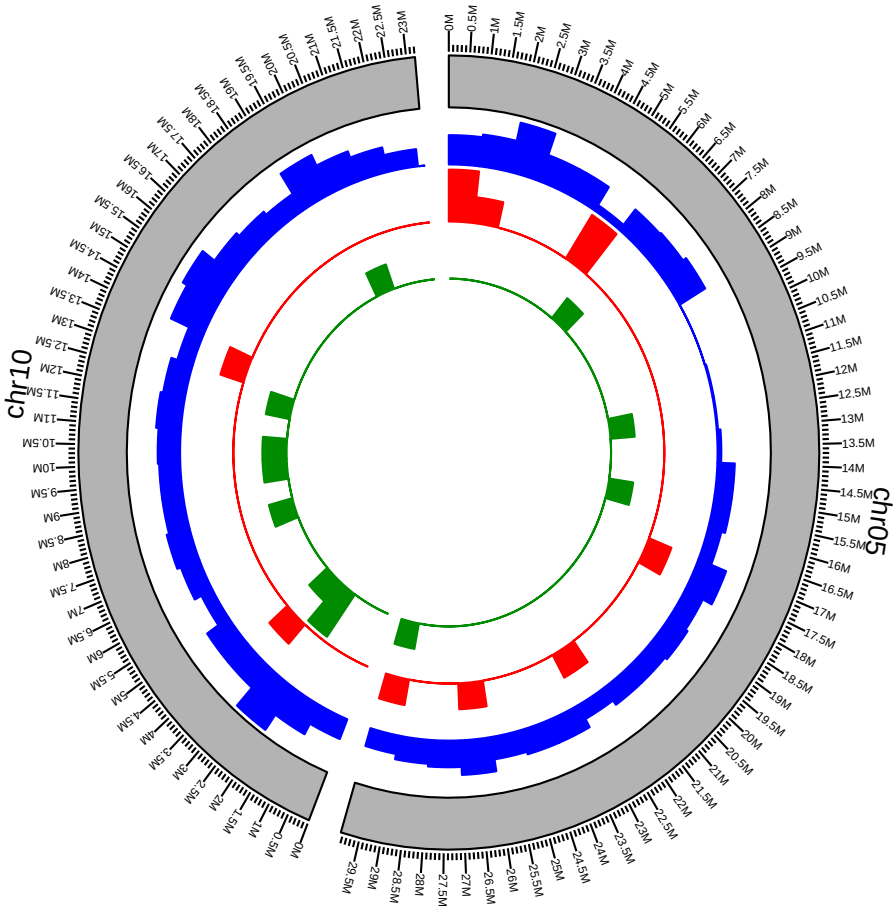

Figure 1: Visualization of SVs distribution in the whole genome. Blue: deletions, Red: duplications, Green: inversions.

```
> plotRegion(sv_all_methods,msu_gff_v7,"chr05",1,200000)
```

This command showed the SVs in the genomic region *chr05:1-200000*. The genes and SVs were shown as circular rectangles with different color.

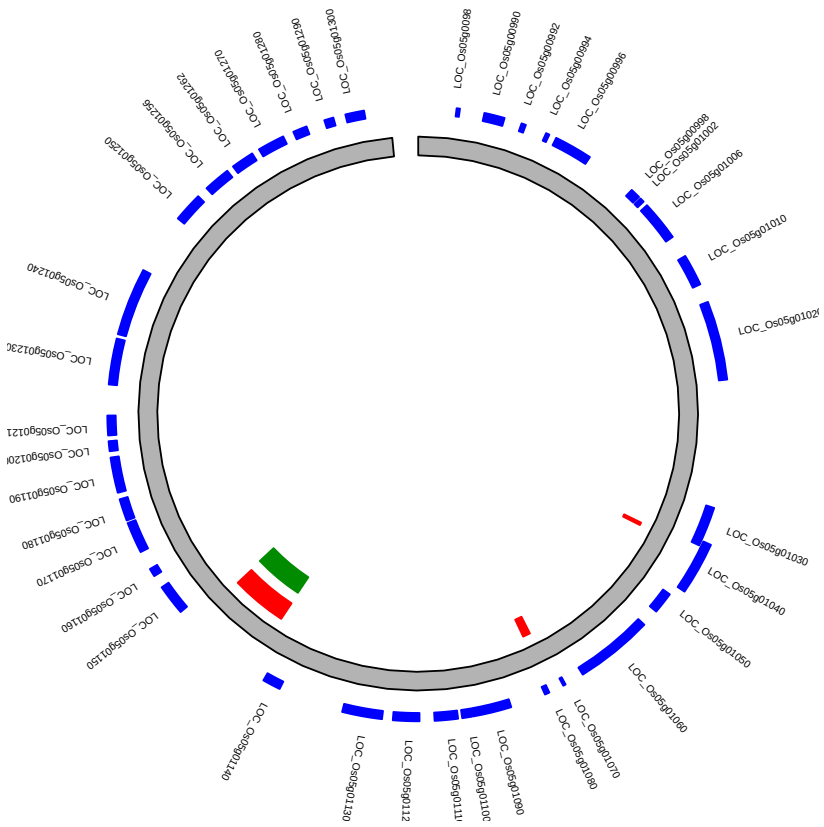

Figure 2: SVs in genomic region chr05:1-200000. Blue: genes, Red: deletions, Green: duplications, Purple: inversions.

3 Session Information

The version number of R and packages loaded for generating the vignette were:

```
R version 3.2.0 RC (2015-04-08 r68161)
Platform: x86_64-apple-darwin13.4.0 (64-bit)
Running under: OS X 10.9.5 (Mavericks)

locale:
[1] C/en_US.UTF-8/en_US.UTF-8/C/en_US.UTF-8/en_US.UTF-8

attached base packages:
[1] stats4    parallel  stats     graphics  grDevices utils
[7] datasets  methods   base

other attached packages:
 [1] rtracklayer_1.28.0   intansv_1.8.0        GenomicRanges_1.20.0
 [4] GenomeInfoDb_1.4.0   IRanges_2.2.0        S4Vectors_0.6.0
 [7] ggbio_1.16.0         ggplot2_1.0.1        BiocGenerics_0.14.0
```

```
[10] plyr_1.8.1

loaded via a namespace (and not attached):
 [1] VariantAnnotation_1.14.0 reshape2_1.4.1
 [3] splines_3.2.0            lattice_0.20-31
 [5] colorspace_1.2-6         GenomicFeatures_1.20.0
 [7] RBGL_1.44.0              survival_2.38-1
 [9] XML_3.98-1.1             foreign_0.8-63
[11] DBI_0.3.1                BiocParallel_1.2.0
[13] RColorBrewer_1.1-2       lambda.r_1.1.7
[15] stringr_0.6.2            zlibbioc_1.14.0
[17] Biostrings_2.36.0        munsell_0.4.2
[19] gtable_0.1.2             futile.logger_1.4
[21] OrganismDbi_1.10.0       labeling_0.3
[23] latticeExtra_0.6-26      Biobase_2.28.0
[25] GGally_0.5.0             biomaRt_2.24.0
[27] AnnotationDbi_1.30.0     proto_0.3-10
[29] Rcpp_0.11.5              acepack_1.3-3.3
[31] scales_0.2.4             BSgenome_1.36.0
[33] graph_1.46.0             Hmisc_3.15-0
[35] XVector_0.8.0            Rsamtools_1.20.0
[37] gridExtra_0.9.1          digest_0.6.8
[39] biovizBase_1.16.0        grid_3.2.0
[41] tools_3.2.0              bitops_1.0-6
[43] RCurl_1.95-4.5           RSQLite_1.0.0
[45] dichromat_2.0-0          Formula_1.2-1
[47] cluster_2.0.1            futile.options_1.0.0
[49] MASS_7.3-40              reshape_0.8.5
[51] rpart_4.1-9              GenomicAlignments_1.4.0
[53] nnet_7.3-9
```

References

Alexej Abyzov, Alexander E. Urban, Michael Snyder, and Mark Gerstein. Cnvnator: An approach to discover, genotype, and characterize typical and atypical cnvs from family and population genome sequencing. *Genome Research*, 21(6):974–984, 2011. URL `http://sv.gersteinlab.org/cnvnator/`.

Ken Chen, John W. Wallis, Michael D. McLellan, David E. Larson, Joelle M. Kalicki, Craig S. Pohl, Sean D. McGrath, Michael C. Wendl, Qunyuan Zhang, Devin P. Locke, Xiaoqi Shi, Robert S. Fulton, Timothy J. Ley, Richard K. Wilson, Li Ding, and Elaine R. Mardis. Breakdancer: an algorithm for high-resolution mapping of genomic structural variation. *Nat Meth*, 6(9):677–681, 2009. URL `http://gmt.genome.wustl.edu/breakdancer/1.2/index.html`.

Tobias Rausch, Thomas Zichner, Andreas Schlattl, Adrian M. Stĺźtz, Vladimir Benes, and Jan O. Korbel. Delly: structural variant discovery by integrated paired-end and split-read analysis. *Bioinformatics*, 28 (18):i333–i339, 2012. URL `http://www.korbel.embl.de/software.html`.

Layer Ryan M., Hall Ira M., and Quinlan Aaron R. Lumpy: A probabilistic framework for structural variant discovery. *arxiv.org*, 2012. URL `https://github.com/arq5x/lumpy-sv`.

Hart Steven N., Sarangi Vivekananda, Moore Raymond, Baheti Saurabh, Bhavsar Jaysheel D., Couch Fergus J., and Kocher Jean-Pierre A. An improved approach for accurate and efficient calling of struc-

tural variations with low-coverage sequence data. *PLoS One*, 2013. URL `http://code.google.com/p/softsearch/`.

Wen Yao and Weibo Xie. intansv: an r package for integrative analysis of structural variations. 2013.

K. Ye, M. H. Schulz, Q. Long, R. Apweiler, and Z. Ning. Pindel: a pattern growth approach to detect break points of large deletions and medium sized insertions from paired-end short reads. *Bioinformatics*, 21(6): 974–984, 2009. URL `http://gmt.genome.wustl.edu/pindel/0.2.4/index.html`.

Jin Zhang, Jiayin Wang, and Yufeng Wu. An improved approach for accurate and efficient calling of structural variations with low-coverage sequence data. *BMC Bioinformatics*, 13:S6, 2012. URL `http://www.engr.uconn.edu/~jiz08001/svseq2.html`.

Thomas Zichner, David A. Garfield, Tobias Rausch, Adrian M. StÍźtz, Enrico CannavÍő, Martina Braun, Eileen E.M. Furlong, and Jan O. Korbel. Impact of genomic structural variation in drosophila melanogaster based on population-scale sequencing. *Genome Research*, 23(3):568–579, 2013.