An Introduction to *intansv*

Wen Yao

April 27, 2014

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':        913 obs. of  4 variables:
  ..$ chromosome: chr [1:913] "chr05" "chr05" "chr05" "chr05" ...
  ..$ pos1      : num [1:913] 65586 86442 120288 153694 201845 ...
  ..$ pos2      : num [1:913] 65938 87430 127670 153801 201959 ...
  ..$ size      : num [1:913] 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 6942424 12014529 15092390 18770925 ...
  ..$ pos2      : num [1:17] 1291678 6944610 12015863 15157462 18771377 ...
  ..$ 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':        336 obs. of  4 variables:
  ..$ chromosome: chr [1:336] "chr05" "chr05" "chr05" "chr05" ...
  ..$ pos1      : num [1:336] 231601 348301 610501 1089301 1524301 ...
  ..$ pos2      : num [1:336] 247800 350100 630600 1103100 1531500 ...
  ..$ size      : num [1:336] 16199 1799 20099 13799 7199 ...
 $ dup:'data.frame':        108 obs. of  4 variables:
  ..$ chromosome: chr [1:108] "chr05" "chr05" "chr05" "chr05" ...
  ..$ pos1      : num [1:108] 405001 973201 1346401 5036101 5190001 ...
  ..$ pos2      : num [1:108] 408900 977700 1359000 5055300 5209500 ...
  ..$ size      : num [1:108] 3899 4499 12599 19199 19499 ...
 - 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':        1322 obs. of  4 variables:
  ..$ chromosome: chr [1:1322] "chr05" "chr05" "chr05" "chr05" ...
  ..$ pos1      : num [1:1322] 65580 86458 153671 172303 201850 ...
  ..$ pos2      : num [1:1322] 65949 87419 153829 172439 201959 ...
  ..$ size      : num [1:1322] 369 961 158 136 109 331 104 128 129 143 ...
 $ dup:'data.frame':        52 obs. of  4 variables:
  ..$ chromosome: chr [1:52] "chr05" "chr05" "chr05" "chr05" ...
  ..$ pos1      : num [1:52] 120803 910613 957201 1133978 1615002 ...
  ..$ pos2      : num [1:52] 128008 911328 957324 1136611 1616030 ...
  ..$ size      : num [1:52] 7205 715 123 2633 1028 ...
 $ inv:'data.frame':        35 obs. of  4 variables:
  ..$ chromosome: chr [1:35] "chr05" "chr05" "chr05" "chr05" ...
  ..$ pos1      : num [1:35] 1374701 3049731 5550226 5678183 6372412 ...
  ..$ pos2      : num [1:35] 1390659 3557307 5550342 5709463 6453728 ...
  ..$ size      : num [1:35] 15958 507576 116 31280 81316 ...
 - 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. You should put all these files in the same directory and feed the path of this directory to `readDelly`. DELLY produces two output files for all kind of SVs. One file with the paired-end predictions and one file with the split-read predictions. The files contain the paired-end predictions of deletions, inversions and duplication in this directory should be named with suffix of *.del*, *.inv* and *.dup* respectively. The files contain the split-read predictions of deletions, inversions and duplication in this directory should be named with suffix of *.del.br*, *.inv.br* and *.dup.br* respectively. See the example dataset for more details.

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

```
List of 3
 $ del:'data.frame':        452 obs. of  4 variables:
  ..$ chromosome: chr [1:452] "chr05" "chr05" "chr05" "chr05" ...
  ..$ pos1      : num [1:452] 76477 334346 366428 405038 498623 ...
  ..$ pos2      : num [1:452] 76913 334613 367941 408729 505449 ...
```

```
  ..$ size      : num [1:452] 435 266 1512 3690 6825 ...
 $ inv:'data.frame':        19 obs. of  4 variables:
  ..$ chromosome: chr [1:19] "chr05" "chr05" "chr05" "chr05" ...
  ..$ pos1      : num [1:19] 3062624 7562060 12014414 13771739 18770853 ...
  ..$ pos2      : num [1:19] 3062746 7562294 12015867 13771852 18771407 ...
  ..$ size      : num [1:19] 121 233 1452 112 553 ...
 $ dup:'data.frame':        40 obs. of  4 variables:
  ..$ chromosome: chr [1:40] "chr05" "chr05" "chr05" "chr05" ...
  ..$ pos1      : num [1:40] 910587 957199 1018140 1614997 3268243 ...
  ..$ pos2      : num [1:40] 911326 957324 1018273 1616027 3268350 ...
  ..$ size      : num [1:40] 738 124 132 1029 106 ...
 - 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':        116 obs. of  4 variables:
  ..$ chromosome: chr [1:116] "chr05" "chr05" "chr05" "chr05" ...
  ..$ pos1      : num [1:116] 516368 1166862 1779896 1839176 2405030 ...
  ..$ pos2      : num [1:116] 516644 1168026 1780996 1839708 2416382 ...
  ..$ size      : num [1:116] 276 1164 1101 531 11353 ...
 $ dup:'data.frame':        10 obs. of  4 variables:
  ..$ chromosome: chr [1:10] "chr05" "chr05" "chr10" "chr10" ...
  ..$ pos1      : num [1:10] 1133974 23363932 1425876 2944164 3624818 ...
  ..$ pos2      : num [1:10] 1136644 23372578 1426506 2960648 3629856 ...
  ..$ size      : num [1:10] 2671 8647 631 16485 5039 ...
 $ inv:'data.frame':        6 obs. of  4 variables:
  ..$ chromosome: chr [1:6] "chr05" "chr05" "chr05" "chr10" ...
  ..$ pos1      : num [1:6] 6942390 18737110 29431750 2152578 18176816 ...
  ..$ pos2      : num [1:6] 6944776 18739194 29432374 2240846 18178542 ...
  ..$ size      : num [1:6] 2387 2085 625 88269 1727 ...
 - 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':        3 obs. of  4 variables:
  ..$ chromosome: chr [1:3] "chr05" "chr10" "chr10"
  ..$ pos1      : num [1:3] 3106237 2435097 7020239
  ..$ pos2      : num [1:3] 3106523 2486815 7020927
  ..$ size      : num [1:3] 286 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':        897 obs. of  8 variables:
  ..$ chromosome : chr [1:897] "chr05" "chr05" "chr05" "chr05" ...
  ..$ pos1       : num [1:897] 65583 86450 201848 208219 230523 ...
  ..$ pos2       : num [1:897] 65944 87424 201959 208562 230655 ...
  ..$ BreakDancer: chr [1:897] "Y" "Y" "Y" "Y" ...
  ..$ Pindel     : chr [1:897] "N" "N" "N" "N" ...
  ..$ CNVnator   : chr [1:897] "N" "N" "N" "N" ...
  ..$ DELLY      : chr [1:897] "Y" "Y" "Y" "Y" ...
  ..$ SVseq2     : chr [1:897] "N" "N" "N" "N" ...
 $ dup:'data.frame':        13 obs. of  8 variables:
  ..$ chromosome : chr [1:13] "chr05" "chr05" "chr05" "chr05" ...
  ..$ pos1       : num [1:13] 910600 957200 1615000 5757284 16420176 ...
  ..$ pos2       : num [1:13] 911327 957324 1616028 5758092 16425276 ...
  ..$ BreakDancer: chr [1:13] "N" "N" "N" "N" ...
  ..$ Pindel     : chr [1:13] "Y" "Y" "Y" "Y" ...
  ..$ CNVnator   : chr [1:13] "N" "N" "N" "N" ...
  ..$ DELLY      : chr [1:13] "Y" "Y" "Y" "Y" ...
  ..$ SVseq2     : chr [1:13] "N" "N" "N" "N" ...
 $ inv:'data.frame':        12 obs. of  8 variables:
  ..$ chromosome : chr [1:12] "chr05" "chr05" "chr05" "chr05" ...
  ..$ pos1       : num [1:12] 6942396 12014454 15092362 18770874 2152650 ...
  ..$ pos2       : num [1:12] 6944670 12015870 15157532 18771397 2240852 ...
  ..$ BreakDancer: chr [1:12] "Y" "Y" "Y" "Y" ...
  ..$ Pindel     : chr [1:12] "N" "Y" "N" "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':        862 obs. of  7 variables:
  ..$ chromosome : chr [1:862] "chr05" "chr05" "chr05" "chr05" ...
  ..$ pos1       : num [1:862] 65583 86450 201848 208219 230523 ...
  ..$ pos2       : num [1:862] 65944 87424 201959 208562 230655 ...
  ..$ BreakDancer: chr [1:862] "Y" "Y" "Y" "Y" ...
  ..$ CNVnator   : chr [1:862] "N" "N" "N" "N" ...
  ..$ DELLY      : chr [1:862] "Y" "Y" "Y" "Y" ...
  ..$ SVseq2     : chr [1:862] "N" "N" "N" "N" ...
 $ dup:'data.frame':        2 obs. of  7 variables:
  ..$ chromosome : chr [1:2] "chr05" "chr05"
  ..$ pos1       : num [1:2] 16420176 29657896
  ..$ pos2       : num [1:2] 16425276 29662746
  ..$ BreakDancer: chr [1:2] "N" "N"
  ..$ CNVnator   : chr [1:2] "Y" "Y"
  ..$ DELLY      : chr [1:2] "Y" "Y"
  ..$ SVseq2     : chr [1:2] "N" "N"
 $ inv:'data.frame':        11 obs. of  7 variables:
  ..$ chromosome : chr [1:11] "chr05" "chr05" "chr05" "chr05" ...
  ..$ pos1       : num [1:11] 6942396 12014474 15092362 18770885 2152650 ...
  ..$ pos2       : num [1:11] 6944670 12015871 15157532 18771392 2240852 ...
  ..$ BreakDancer: 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" ...
```

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*.

```
> 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 with 2 ranges and 0 metadata columns:
      seqnames        ranges strand
         <Rle>     <IRanges>  <Rle>
  [1]    chr05 [1, 29958434]      *
  [2]    chr10 [1, 23207287]      *
  ---
  seqlengths:
      chr05    chr10
   29958434 23207287
```

```
> 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)
```

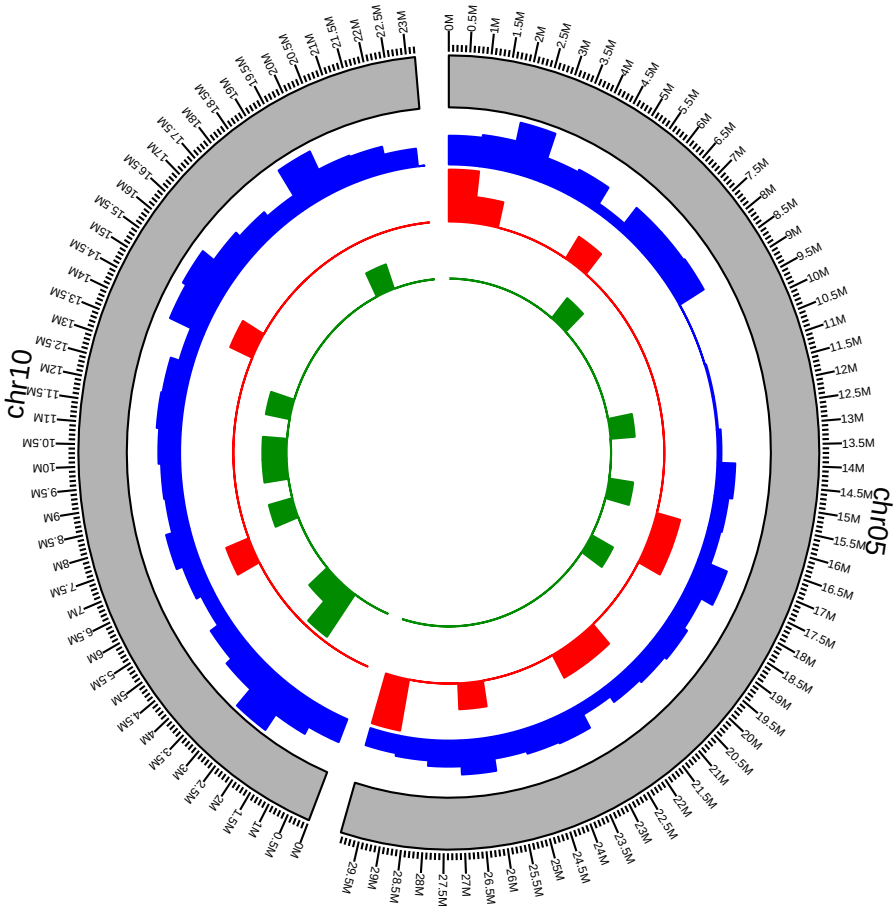

Figure 1: Visualization of SVs distribution in the whole genome. Blue: deletions, Red: duplications, Green: inversions.

```
GRanges 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
  ---
  seqlengths:
   chr05 chr10
      NA    NA
```

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

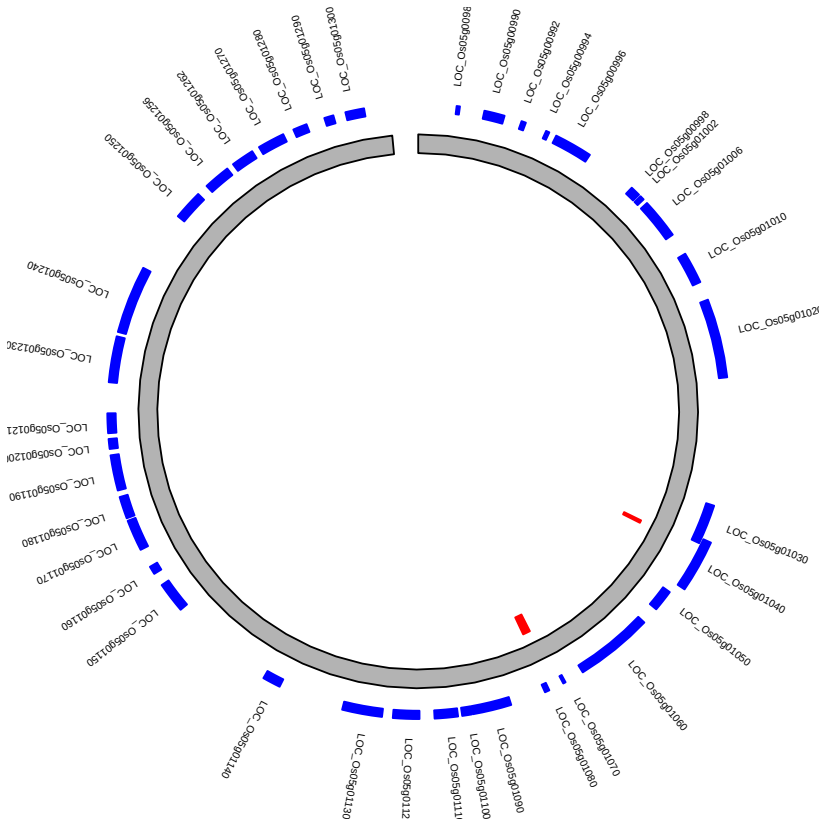

Figure 2: SVs in genomic region chr05:1-200000. Blue: genes, Red: deletions, Green: duplications, Purple: inversions.

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

3 Session Information

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

```
R version 3.1.0 RC (2014-04-02 r65358)
Platform: i386-w64-mingw32/i386 (32-bit)

locale:
[1] LC_COLLATE=C
[2] LC_CTYPE=English_United States.1252
[3] LC_MONETARY=English_United States.1252
[4] LC_NUMERIC=C
[5] LC_TIME=English_United States.1252

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

other attached packages:
[1] intansv_1.4.1        GenomicRanges_1.16.2 GenomeInfoDb_1.0.2
[4] IRanges_1.22.4       ggbio_1.12.2         ggplot2_0.9.3.1
[7] BiocGenerics_0.10.0  plyr_1.8.1

loaded via a namespace (and not attached):
 [1] AnnotationDbi_1.26.0     BBmisc_1.6
 [3] BSgenome_1.32.0          BatchJobs_1.2
 [5] Biobase_2.24.0           BiocParallel_0.6.0
 [7] Biostrings_2.32.0        DBI_0.2-7
 [9] Formula_1.1-1            GenomicAlignments_1.0.0
[11] GenomicFeatures_1.16.0   Hmisc_3.14-4
[13] MASS_7.3-31              RColorBrewer_1.0-5
[15] RCurl_1.95-4.1           RSQLite_0.11.4
[17] Rcpp_0.11.1              Rsamtools_1.16.0
[19] VariantAnnotation_1.10.0 XML_3.98-1.1
[21] XVector_0.4.0            biomaRt_2.20.0
[23] biovizBase_1.12.1        bitops_1.0-6
[25] brew_1.0-6               cluster_1.15.2
[27] codetools_0.2-8          colorspace_1.2-4
[29] dichromat_2.0-0          digest_0.6.4
[31] fail_1.2                 foreach_1.4.2
[33] grid_3.1.0               gridExtra_0.9.1
[35] gtable_0.1.2             iterators_1.0.7
[37] labeling_0.2             lattice_0.20-29
[39] latticeExtra_0.6-26      munsell_0.4.2
[41] proto_0.3-10             reshape2_1.4
[43] rtracklayer_1.24.0       scales_0.2.4
[45] sendmailR_1.1-2          splines_3.1.0
[47] stats4_3.1.0             stringr_0.6.2
[49] survival_2.37-7          tools_3.1.0
[51] zlibbioc_1.10.0
```

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 structural 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. Stĺźtz, Enrico Cannavĺő, 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.