An Introduction to *intansv*

Wen Yao

June 1, 2017

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

```
[1] "/tmp/RtmpRXAIWT/Rinst48b0444b4c52/intansv/extdata/ZS97.breakdancer.sv"
```

```
> breakdancer <- readBreakDancer(breakdancer.file.path)
> str(breakdancer)
```

```
List of 2
 $ del:'data.frame':        1007 obs. of  5 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 ...
  ..$ info      : chr [1:1007] "score=88;PE=5" "score=99;PE=20" "score=83;PE=6" "score=80;PE=4" ...
 $ inv:'data.frame':        17 obs. of  5 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 ...
  ..$ info      : chr [1:17] "score=99;PE=3" "score=99;PE=5" "score=99;PE=6" "score=99;PE=4" ...
 - attr(*, "method")= chr "BreakDancer"
```

BreakDancer is able to predict deletions and inversions. The prediction of all type 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.dir.path <- system.file("extdata/cnvnator", package="intansv")
> cnvnator.dir.path
```

```
[1] "/tmp/RtmpRXAIWT/Rinst48b0444b4c52/intansv/extdata/cnvnator"
```

```
> cnvnator <- readCnvnator(cnvnator.dir.path)
> str(cnvnator)
```

```
List of 2
 $ del:'data.frame':        369 obs. of  5 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 ...
  ..$ info      : chr [1:369] "eval1=0.0100681;eval2=1386200000;eval3=10.1189;eval4=1690690000" "eval1=8.
 $ dup:'data.frame':        113 obs. of  5 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 ...
  ..$ info      : chr [1:113] "eval1=0.0169421;eval2=0;eval3=0.00545604;eval4=0" "eval1=0.0118874;eval2=3
 - 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.dir.path <- system.file("extdata/svseq2", package="intansv")
> svseq.dir.path

[1] "/tmp/RtmpRXAIWT/Rinst48b0444b4c52/intansv/extdata/svseq2"

> svseq <- readSvseq(svseq.dir.path)
> str(svseq)

List of 1
 $ del:'data.frame':        135 obs. of  5 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 ...
  ..$ info      : chr [1:135] "SR=3" "SR=3" "SR=4" "SR=4" ...
 - 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 type 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.dir.path <- system.file("extdata/delly", package="intansv")
> delly.dir.path

[1] "/tmp/RtmpRXAIWT/Rinst48b0444b4c52/intansv/extdata/delly"

> delly <- readDelly(delly.dir.path)
> str(delly)

List of 3
 $ del:'data.frame':        869 obs. of  5 variables:
  ..$ chromosome: chr [1:869] "chr05" "chr05" "chr05" "chr05" ...
  ..$ pos1      : int [1:869] 65578 86458 120268 208220 232721 280422 340472 348421 366440 379196 ...
  ..$ pos2      : num [1:869] 65949 87419 127693 208551 245063 ...
  ..$ size      : num [1:869] 371 961 7425 331 12342 ...
  ..$ info      : chr [1:869] "PE=5;SR=0" "PE=18;SR=0" "PE=7;SR=0" "PE=17;SR=0" ...
 $ dup:'data.frame':        37 obs. of  5 variables:
  ..$ chromosome: chr [1:37] "chr05" "chr05" "chr05" "chr05" ...
  ..$ pos1      : int [1:37] 120788 910613 1133976 1614998 3087194 5696565 5757285 8175735 14472319 17377
  ..$ pos2      : num [1:37] 128007 911328 1136614 1616026 3090871 ...
```

```
  ..$ size      : num [1:37] 7219 715 2638 1028 3677 ...
  ..$ info      : chr [1:37] "PE=5;SR=0" "PE=9;SR=0" "PE=11;SR=0" "PE=14;SR=5" ...
 $ inv:'data.frame':        32 obs. of  5 variables:
  ..$ chromosome: chr [1:32] "chr05" "chr05" "chr05" "chr05" ...
  ..$ pos1      : int [1:32] 5550225 6372412 6942366 8422112 10402721 10940594 11062213 12014418 13291663
  ..$ pos2      : num [1:32] 5550342 6453740 6944731 8834987 10402842 ...
  ..$ size      : num [1:32] 117 81328 2365 412875 121 ...
  ..$ info      : chr [1:32] "PE=11;SR=0" "PE=6;SR=0" "PE=5;SR=0" "PE=3;SR=0" ...
 - attr(*, "method")= chr "DELLY"
```

DELLY is able to predict deletions, inversions and duplications. The final prediction of different type of SVs given by DELLY were written to different files. DELLY v0.7.6 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.dir.path <- system.file("extdata/pindel", package="intansv")
> pindel.dir.path

[1] "/tmp/RtmpRXAIWT/Rinst48b0444b4c52/intansv/extdata/pindel"

> pindel <- readPindel(pindel.dir.path)
> str(pindel)

List of 3
 $ del:'data.frame':        631 obs. of  5 variables:
  ..$ chromosome: chr [1:631] "chr05" "chr05" "chr05" "chr05" ...
  ..$ pos1      : num [1:631] 76477 86438 120523 334346 366428 ...
  ..$ pos2      : num [1:631] 76913 87419 127917 334613 367941 ...
  ..$ size      : num [1:631] 435 980 7393 266 1512 ...
  ..$ info      : chr [1:631] "SR=4;score=8" "SR=11;score=42" "SR=4;score=5" "SR=9;score=30" ...
 $ inv:'data.frame':        208 obs. of  5 variables:
  ..$ chromosome: chr [1:208] "chr05" "chr05" "chr05" "chr05" ...
  ..$ pos1      : num [1:208] 616253 777580 917524 1072380 1421491 ...
  ..$ pos2      : num [1:208] 616376 777810 917629 1072496 1421596 ...
  ..$ size      : num [1:208] 122 229 118 115 116 140 570 160 111 5050 ...
  ..$ info      : chr [1:208] "SR=4;score=9" "SR=6;score=7" "SR=6;score=15" "SR=7;score=14" ...
 $ dup:'data.frame':        72 obs. of  5 variables:
  ..$ chromosome: chr [1:72] "chr05" "chr05" "chr05" "chr05" ...
  ..$ pos1      : num [1:72] 97139 120110 910586 957199 1018140 ...
  ..$ pos2      : num [1:72] 97308 127734 911325 957324 1018273 ...
  ..$ size      : num [1:72] 168 7623 738 124 132 ...
  ..$ info      : chr [1:72] "SR=6;score=12" "SR=7;score=14" "SR=3;score=4" "SR=11;score=36" ...
 - 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 type 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.file.path <- system.file("extdata/ZS97.lumpy.pesr.bedpe",
+                    package="intansv")
> lumpy.file.path
```

```
[1] "/tmp/RtmpRXAIWT/Rinst48b0444b4c52/intansv/extdata/ZS97.lumpy.pesr.bedpe"
```

```
> lumpy <- readLumpy(lumpy.file.path)
> str(lumpy)
```

```
List of 3
 $ del:'data.frame':        652 obs. of  5 variables:
  ..$ chromosome: chr [1:652] "chr05" "chr05" "chr05" "chr05" ...
  ..$ pos1      : int [1:652] 86439 201770 208089 280376 334242 348326 366352 379143 405022 484624 ...
  ..$ pos2      : num [1:652] 87534 202010 208559 281233 334635 ...
  ..$ size      : num [1:652] 1095 240 470 857 393 ...
  ..$ info      : chr [1:652] "SU=4" "SU=8" "SU=4" "SU=5" ...
 $ dup:'data.frame':        26 obs. of  5 variables:
  ..$ chromosome: chr [1:26] "chr05" "chr05" "chr05" "chr05" ...
  ..$ pos1      : int [1:26] 120805 910641 1134012 1615101 5757330 16420453 18771458 22744738 23911718 24
  ..$ pos2      : num [1:26] 127911 911277 1136594 1616008 5758050 ...
  ..$ size      : num [1:26] 7106 636 2582 907 720 ...
  ..$ info      : chr [1:26] "SU=4" "SU=5" "SU=6" "SU=5" ...
 $ inv: NULL
 - attr(*, "method")= chr "Lumpy"
```

Lumpy is able to predict deletions, inversions and duplications. The prediction of all type 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.file.path <- system.file("extdata/ZS97.softsearch", package="intansv")
> softSearch.file.path
```

```
[1] "/tmp/RtmpRXAIWT/Rinst48b0444b4c52/intansv/extdata/ZS97.softsearch"
```

```
> softSearch <- readSoftSearch(softSearch.file.path)
> str(softSearch)
```

```
List of 3
 $ del:'data.frame':        47 obs. of  5 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 ...
  ..$ info      : chr [1:47] "nr=18;sc=5" "nr=30;sc=6" "nr=24;sc=5" "nr=15;sc=5" ...
 $ dup: NULL
 $ inv:'data.frame':        4 obs. of  5 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
  ..$ info      : chr [1:4] "nr=4;sc=7" "nr=10;sc=6" "nr=17;sc=6" "nr=15;sc=5"
 - attr(*, "method")= chr "softSearch"
```

softSearch is able to predict deletions, inversions and duplications. The prediction of all type 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, 2017) 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':        869 obs. of  5 variables:
  ..$ chromosome: chr [1:869] "chr05" "chr05" "chr05" "chr05" ...
  ..$ pos1      : num [1:869] 65582 86446 120360 208219 280432 ...
  ..$ pos2      : num [1:869] 65944 87423 127760 208562 281179 ...
  ..$ methods   : chr [1:869] "BreakDancer:DELLY" "BreakDancer:Pindel:DELLY" "BreakDancer:Pindel:DELLY" "
  ..$ info      : chr [1:869] "score=88;PE=5:PE=5;SR=0" "score=99;PE=20:SR=11;score=42:PE=18;SR=0" "score
 $ dup:'data.frame':        13 obs. of  5 variables:
  ..$ chromosome: chr [1:13] "chr05" "chr05" "chr05" "chr05" ...
  ..$ pos1      : num [1:13] 120449 910600 1614998 3087200 5696556 ...
  ..$ pos2      : num [1:13] 127870 911326 1616026 3090903 5699328 ...
  ..$ methods   : chr [1:13] "Pindel:DELLY" "Pindel:DELLY" "Pindel:DELLY" "Pindel:DELLY" ...
  ..$ info      : chr [1:13] "SR=7;score=14:PE=5;SR=0" "SR=3;score=4:PE=9;SR=0" "SR=10;score=27:PE=14;SR=
 $ inv:'data.frame':        13 obs. of  5 variables:
  ..$ chromosome: chr [1:13] "chr05" "chr05" "chr05" "chr05" ...
  ..$ pos1      : num [1:13] 6372420 6942408 12014500 15090018 18770907 ...
  ..$ pos2      : num [1:13] 6453749 6944684 12015907 15158197 18771410 ...
  ..$ methods   : chr [1:13] "Pindel:DELLY" "BreakDancer:DELLY" "BreakDancer:DELLY" "BreakDancer:Pindel:D
  ..$ info      : chr [1:13] "SR=11;score=12:PE=6;SR=0" "score=99;PE=5:PE=5;SR=0" "score=99;PE=6:PE=8;SR=
```

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':        738 obs. of  5 variables:
  ..$ chromosome: chr [1:738] "chr05" "chr05" "chr05" "chr05" ...
  ..$ pos1      : num [1:738] 65582 86450 120278 208219 280432 ...
  ..$ pos2      : num [1:738] 65944 87424 127682 208562 281179 ...
  ..$ methods   : chr [1:738] "BreakDancer:DELLY" "BreakDancer:DELLY" "BreakDancer:DELLY" "BreakDancer:DE
  ..$ info      : chr [1:738] "score=88;PE=5:PE=5;SR=0" "score=99;PE=20:PE=18;SR=0" "score=83;PE=6:PE=7;S
 $ dup:'data.frame':        1 obs. of  5 variables:
  ..$ chromosome: chr "chr05"
  ..$ pos1      : num 29657896
  ..$ pos2      : num 29662746
  ..$ methods   : chr "CNVnator:DELLY"
  ..$ info      : chr "eval1=0.000194584;eval2=1.3107e-14;eval3=0.0408135;eval4=2.64522e-17:PE=22;SR=0"
 $ inv:'data.frame':        11 obs. of  5 variables:
  ..$ chromosome: chr [1:11] "chr05" "chr05" "chr05" "chr05" ...
  ..$ pos1      : num [1:11] 6942408 12014500 15092381 18770907 2152662 ...
  ..$ pos2      : num [1:11] 6944684 12015907 15157552 18771410 2240864 ...
  ..$ methods   : chr [1:11] "BreakDancer:DELLY" "BreakDancer:DELLY" "BreakDancer:DELLY" "BreakDancer:DEL
  ..$ info      : chr [1:11] "score=99;PE=5:PE=5;SR=0" "score=99;PE=6:PE=8;SR=0" "score=99;PE=4:PE=3;SR=0
```

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. To avoid confusion, this file should be a plain text file with 6 columns, the chromosome ID, tag of each genome element, start coordinate, end coordinate, strand, ID of each genome element. An example genomic annotation file has been storaged in the example dataset of *intansv*.

```
> anno.file.path <- system.file("extdata/chr05_chr10.anno.txt", package="intansv")
> anno.file.path

[1] "/tmp/RtmpRXAIWT/Rinst48b0444b4c52/intansv/extdata/chr05_chr10.anno.txt"

> msu_gff_v7 <- read.table(anno.file.path, head=TRUE, as.is=TRUE)
> head(msu_gff_v7)

    chr  tag start  end strand                      ID
1 chr05 gene  4003 4356      +          LOC_Os05g00988
2 chr05 mRNA  4003 4356      +        LOC_Os05g00988.1
3 chr05 exon  4003 4356      + LOC_Os05g00988.1:exon_1
4 chr05  CDS  4003 4356      +  LOC_Os05g00988.1:cds_1
5 chr05 gene  6935 9099      +          LOC_Os05g00990
6 chr05 mRNA  6935 9099      +        LOC_Os05g00990.1

> 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                 methods
1      chr05  65582  65944        BreakDancer:DELLY
2      chr05  86446  87423 BreakDancer:Pindel:DELLY
3      chr05 120360 127760 BreakDancer:Pindel:DELLY
4      chr05 208219 208562        BreakDancer:DELLY
5      chr05 208219 208562        BreakDancer:DELLY
6      chr05 280432 281179        BreakDancer:DELLY
                                    info  tag  start    end strand
1                 score=88;PE=5:PE=5;SR=0 <NA>     NA     NA   <NA>
```

```
2 score=99;PE=20:SR=11;score=42:PE=18;SR=0 <NA>     NA     NA   <NA>
3     score=83;PE=6:SR=4;score=5:PE=7;SR=0 <NA>     NA     NA   <NA>
4                 score=99;PE=9:PE=17;SR=0 gene 207853 209693      -
5                 score=99;PE=9:PE=17;SR=0 mRNA 207853 209693      -
6                 score=99;PE=7:PE=10;SR=0 <NA>     NA     NA   <NA>
                ID
1             <NA>
2             <NA>
3             <NA>
4   LOC_Os05g01330
5 LOC_Os05g01330.1
6             <NA>
```

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.file.path <- system.file("extdata/chr05_chr10.genome.txt", package="intansv")
> genome.file.path

[1] "/tmp/RtmpRXAIWT/Rinst48b0444b4c52/intansv/extdata/chr05_chr10.genome.txt"

> genome <- read.table(genome.file.path, head=TRUE, as.is=TRUE)
> 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.

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

    chr  tag start  end strand                      ID
1 chr05 gene  4003 4356      +          LOC_Os05g00988
2 chr05 mRNA  4003 4356      +        LOC_Os05g00988.1
3 chr05 exon  4003 4356      + LOC_Os05g00988.1:exon_1

> 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.

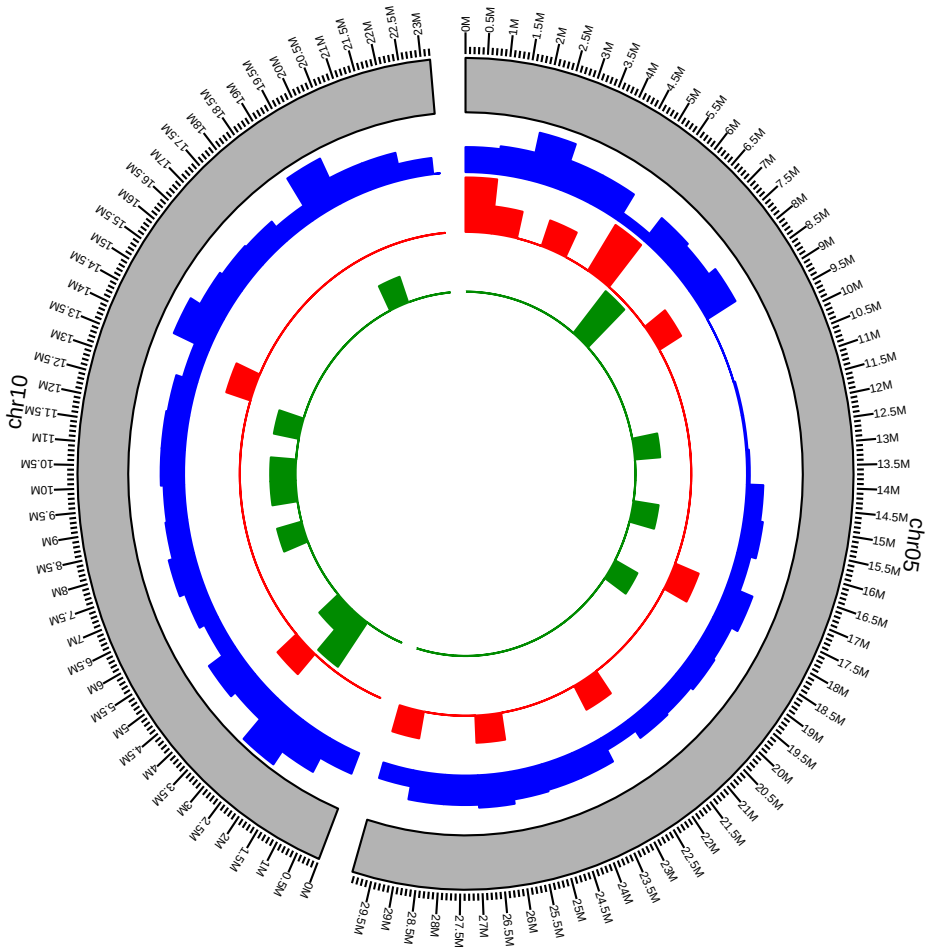

Figure 1: Visualization of SVs distribution in the whole genome. Blue: deletions, Red: duplications, Green: inversions.

3 Session Information

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

```
R version 3.4.0 (2017-04-21)
Platform: x86_64-pc-linux-gnu (64-bit)
Running under: Ubuntu 16.04.2 LTS

Matrix products: default
BLAS: /home/biocbuild/bbs-3.6-bioc/R/lib/libRblas.so
LAPACK: /home/biocbuild/bbs-3.6-bioc/R/lib/libRlapack.so

locale:
 [1] LC_CTYPE=en_US.UTF-8       LC_NUMERIC=C
 [3] LC_TIME=en_US.UTF-8        LC_COLLATE=C
```

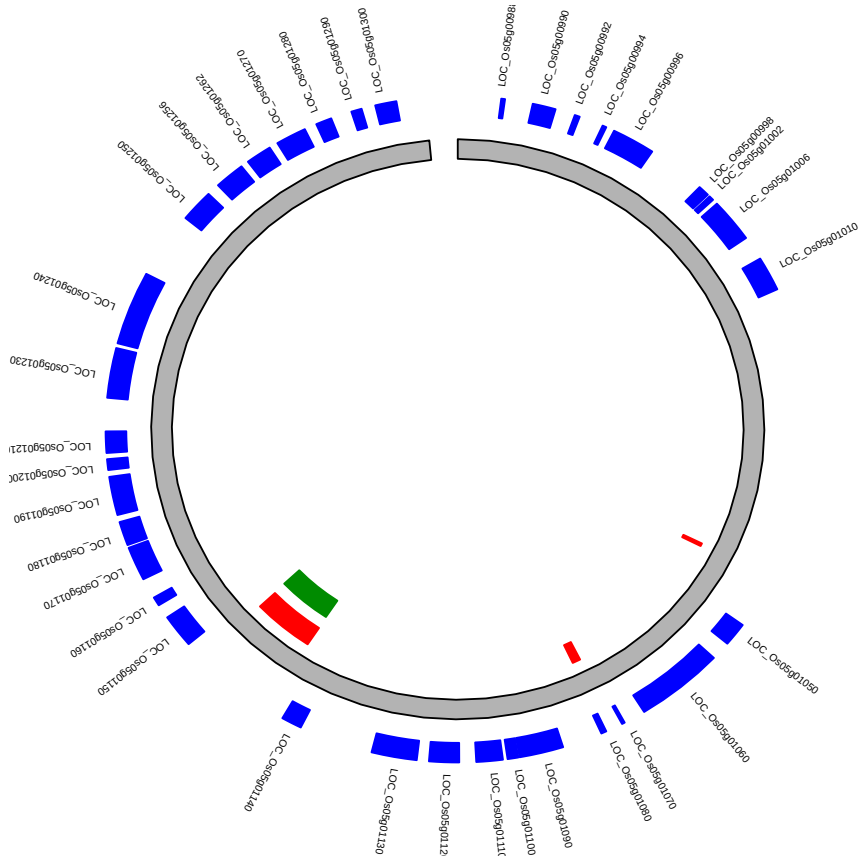

Figure 2: SVs in genomic region chr05:1-200000. Blue: genes, Red: deletions, Green: duplications, Purple: inversions.

```
 [5] LC_MONETARY=en_US.UTF-8    LC_MESSAGES=en_US.UTF-8
 [7] LC_PAPER=en_US.UTF-8       LC_NAME=C
 [9] LC_ADDRESS=C               LC_TELEPHONE=C
[11] LC_MEASUREMENT=en_US.UTF-8 LC_IDENTIFICATION=C

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

other attached packages:
[1] intansv_1.17.1       GenomicRanges_1.29.4 GenomeInfoDb_1.13.3
[4] IRanges_2.11.3       S4Vectors_0.15.2     ggbio_1.25.0
[7] ggplot2_2.2.1        BiocGenerics_0.23.0  plyr_1.8.4

loaded via a namespace (and not attached):
 [1] httr_1.2.1                   Biobase_2.37.2
 [3] AnnotationHub_2.9.5          splines_3.4.0
 [5] shiny_1.0.3                  Formula_1.2-1
```

```
 [7] assertthat_0.2.0             interactiveDisplayBase_1.15.0
 [9] latticeExtra_0.6-28          RBGL_1.53.0
[11] BSgenome_1.45.1              GenomeInfoDbData_0.99.0
[13] Rsamtools_1.29.0             yaml_2.1.14
[15] progress_1.1.2               RSQLite_1.1-2
[17] backports_1.1.0              lattice_0.20-35
[19] biovizBase_1.25.1            digest_0.6.12
[21] RColorBrewer_1.1-2           XVector_0.17.0
[23] checkmate_1.8.2              colorspace_1.3-2
[25] httpuv_1.3.3                 htmltools_0.3.6
[27] Matrix_1.2-10                OrganismDbi_1.19.0
[29] XML_3.98-1.7                 biomaRt_2.33.1
[31] zlibbioc_1.23.0              xtable_1.8-2
[33] scales_0.4.1                 BiocParallel_1.11.1
[35] htmlTable_1.9                tibble_1.3.3
[37] AnnotationFilter_1.1.1       SummarizedExperiment_1.7.4
[39] GenomicFeatures_1.29.5       nnet_7.3-12
[41] lazyeval_0.2.0               mime_0.5
[43] survival_2.41-3              magrittr_1.5
[45] memoise_1.1.0                GGally_1.3.0
[47] foreign_0.8-68               graph_1.55.0
[49] BiocInstaller_1.27.2         tools_3.4.0
[51] data.table_1.10.4            prettyunits_1.0.2
[53] matrixStats_0.52.2           stringr_1.2.0
[55] munsell_0.4.3                cluster_2.0.6
[57] DelayedArray_0.3.8           AnnotationDbi_1.39.1
[59] ensembldb_2.1.7              Biostrings_2.45.2
[61] compiler_3.4.0               rlang_0.1.1
[63] grid_3.4.0                   RCurl_1.95-4.8
[65] dichromat_2.0-0              VariantAnnotation_1.23.2
[67] htmlwidgets_0.8              labeling_0.3
[69] bitops_1.0-6                 base64enc_0.1-3
[71] gtable_0.2.0                 curl_2.6
[73] DBI_0.6-1                    reshape_0.8.6
[75] reshape2_1.4.2               R6_2.2.1
[77] GenomicAlignments_1.13.2     gridExtra_2.2.1
[79] knitr_1.16                   rtracklayer_1.37.1
[81] Hmisc_4.0-3                  ProtGenerics_1.9.0
[83] stringi_1.1.5                Rcpp_0.12.11
[85] rpart_4.1-11                 acepack_1.4.1
```

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. 2017.

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.