An Introduction to *intansv*

Wen Yao

October 27, 2020

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 from bioconductor

```
> if (!requireNamespace("BiocManager", quietly=TRUE))
+     install.packages("BiocManager")
> BiocManager::install("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/RtmpTNVj6Y/Rinst4ae7115d3df6/intansv/extdata/ZS97.breakdancer.sv"

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

List of 2
 $ del:'data.frame':        1034 obs. of  5 variables:
  ..$ chromosome: chr [1:1034] "chr05" "chr05" "chr05" "chr05" ...
  ..$ pos1      : int [1:1034] 65575 86452 120264 153666 201845 208214 230521 276048 280404 334347 ...
  ..$ pos2      : int [1:1034] 65949 87419 127693 153829 201959 208577 230654 276182 281218 334632 ...
  ..$ size      : num [1:1034] 353 988 7382 107 114 ...
  ..$ info      : chr [1:1034] "score=88;PE=5" "score=99;PE=20" "score=83;PE=6" "score=80;PE=4" ...
 $ inv:'data.frame':        24 obs. of  5 variables:
  ..$ chromosome: chr [1:24] "chr05" "chr05" "chr05" "chr05" ...
  ..$ pos1      : int [1:24] 1291603 6942362 12014477 18770973 20583909 29431819 2152645 2460953 3357135
  ..$ pos2      : int [1:24] 1291649 6944726 12016019 18771402 20584709 29432511 2240933 2512834 3393347
  ..$ size      : num [1:24] -105 2186 1334 452 320 ...
  ..$ info      : chr [1:24] "score=99;PE=3" "score=99;PE=5" "score=99;PE=6" "score=99;PE=5" ...
 - 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/RtmpTNVj6Y/Rinst4ae7115d3df6/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/RtmpTNVj6Y/Rinst4ae7115d3df6/intansv/extdata/svseq2"

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

List of 1
 $ del:'data.frame':        4 obs. of  5 variables:
  ..$ chromosome: chr [1:4] "chr10" "chr10" "chr10" "chr10"
  ..$ pos1      : num [1:4] 84242 93888 22588070 22678617
  ..$ pos2      : num [1:4] 87392 94156 22591496 22679241
  ..$ size      : num [1:4] 3150 268 3426 624
  ..$ info      : chr [1:4] "SR=3" "SR=3" "SR=5" "SR=10"
 - 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.file.path <- system.file("extdata/ZS97.DELLY.vcf", package="intansv")
> delly.file.path

[1] "/tmp/RtmpTNVj6Y/Rinst4ae7115d3df6/intansv/extdata/ZS97.DELLY.vcf"

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

List of 3
 $ del:'data.frame':        826 obs. of  5 variables:
  ..$ chromosome: chr [1:826] "chr05" "chr05" "chr05" "chr05" ...
  ..$ pos1      : int [1:826] 65578 86458 120270 208220 232721 280422 340472 348421 366440 379196 ...
  ..$ pos2      : num [1:826] 65949 87419 127693 208551 245063 ...
  ..$ size      : num [1:826] 371 961 7423 331 12342 ...
  ..$ info      : chr [1:826] "SU=5" "SU=21" "SU=8" "SU=17" ...
 $ dup:'data.frame':        36 obs. of  5 variables:
  ..$ chromosome: chr [1:36] "chr05" "chr05" "chr05" "chr05" ...
  ..$ pos1      : int [1:36] 120788 910613 1133976 1615002 3087194 5696565 5757285 8175735 14472319 17377
  ..$ pos2      : num [1:36] 128007 911328 1136614 1616030 3090871 ...
```

```
  ..$ size      : num [1:36] 7219 715 2638 1028 3677 ...
  ..$ info      : chr [1:36] "SU=6" "SU=12" "SU=13" "SU=14" ...
 $ inv:'data.frame':        67 obs. of  5 variables:
  ..$ chromosome: chr [1:67] "chr05" "chr05" "chr05" "chr05" ...
  ..$ pos1      : int [1:67] 2088869 2317728 3801772 5550225 5899550 6372412 6482296 6942366 8422112 9146
  ..$ pos2      : num [1:67] 2089034 2990494 3802074 5550342 5899750 ...
  ..$ size      : num [1:67] 165 672766 302 117 200 ...
  ..$ info      : chr [1:67] "SU=3" "SU=3" "SU=3" "SU=12" ...
 - attr(*, "method")= chr "Delly"
```

DELLY is able to predict deletions, inversions and duplications. The prediction of all type of SVs were provided as a single VCF file by DELLY. You can feed this file to to `readDelly` and it will do all the tedious work for you. See the example dataset for more details.

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

[1] "/tmp/RtmpTNVj6Y/Rinst4ae7115d3df6/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':        113 obs. of  5 variables:
  ..$ chromosome: chr [1:113] "chr10" "chr10" "chr10" "chr10" ...
  ..$ pos1      : num [1:113] 451780 650354 1386326 1794264 2460952 ...
  ..$ pos2      : num [1:113] 451886 650500 1387907 1794599 2512838 ...
  ..$ size      : num [1:113] 127 145 1580 334 51885 ...
  ..$ info      : chr [1:113] "SR=3;score=6" "SR=3;score=6" "SR=3;score=6" "SR=4;score=5" ...
 $ dup:'data.frame':        33 obs. of  5 variables:
  ..$ chromosome: chr [1:33] "chr05" "chr05" "chr05" "chr05" ...
  ..$ pos1      : num [1:33] 97139 120110 910586 957199 1018140 ...
  ..$ pos2      : num [1:33] 97308 127734 911325 957324 1018273 ...
  ..$ size      : num [1:33] 168 7623 738 124 132 ...
  ..$ info      : chr [1:33] "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.vcf",
+                    package="intansv")
> lumpy.file.path
```

```
[1] "/tmp/RtmpTNVj6Y/Rinst4ae7115d3df6/intansv/extdata/ZS97.LUMPY.vcf"
```

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

```
List of 3
 $ del:'data.frame':        1010 obs. of  5 variables:
  ..$ chromosome: chr [1:1010] "chr05" "chr05" "chr05" "chr05" ...
  ..$ pos1      : int [1:1010] 65576 86456 120268 208218 230524 276033 280420 334350 348419 366438 ...
  ..$ pos2      : num [1:1010] 65948 87418 127692 208550 230674 ...
  ..$ size      : num [1:1010] 372 962 7424 332 150 ...
  ..$ info      : chr [1:1010] "SU=6" "SU=22" "SU=8" "SU=17" ...
 $ dup:'data.frame':        38 obs. of  5 variables:
  ..$ chromosome: chr [1:38] "chr05" "chr05" "chr05" "chr05" ...
  ..$ pos1      : int [1:38] 120787 910612 1133975 1615001 1790511 5696564 5757284 14472318 17377711 2163
  ..$ pos2      : num [1:38] 128005 911326 1136612 1616028 1790644 ...
  ..$ size      : num [1:38] 7218 714 2637 1027 133 ...
  ..$ info      : chr [1:38] "SU=6" "SU=12" "SU=13" "SU=14" ...
 $ inv:'data.frame':        14 obs. of  5 variables:
  ..$ chromosome: chr [1:14] "chr05" "chr05" "chr05" "chr05" ...
  ..$ pos1      : int [1:14] 5550224 11480998 19750934 21005616 28356770 577811 3019800 7020548 7552445 7
  ..$ pos2      : num [1:14] 5550341 11481149 19751474 21005857 28392270 ...
  ..$ size      : num [1:14] 117 151 540 241 35500 ...
  ..$ info      : chr [1:14] "SU=13" "SU=4" "SU=6" "SU=4" ...
 - 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 VCF 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/RtmpTNVj6Y/Rinst4ae7115d3df6/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 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':        832 obs. of  5 variables:
  ..$ chromosome: chr [1:832] "chr05" "chr05" "chr05" "chr05" ...
  ..$ pos1      : num [1:832] 65576 86449 120352 208217 280413 ...
  ..$ pos2      : num [1:832] 65949 87419 127768 208564 281198 ...
  ..$ methods   : chr [1:832] "BreakDancer:Delly" "BreakDancer:Pindel:Delly" "BreakDancer:Pindel:Delly" "
  ..$ info      : chr [1:832] "score=88;PE=5:SU=5" "score=99;PE=20:SR=11;score=42:SU=21" "score=83;PE=6:S
 $ dup:'data.frame':        11 obs. of  5 variables:
  ..$ chromosome: chr [1:11] "chr05" "chr05" "chr05" "chr05" ...
  ..$ pos1      : num [1:11] 120449 910600 1615000 3087200 5696556 ...
  ..$ pos2      : num [1:11] 127870 911326 1616028 3090903 5699328 ...
  ..$ methods   : chr [1:11] "Pindel:Delly" "Pindel:Delly" "Pindel:Delly" "Pindel:Delly" ...
  ..$ info      : chr [1:11] "SR=7;score=14:SU=6" "SR=3;score=4:SU=12" "SR=10;score=27:SU=14" "SR=5;score
 $ inv:'data.frame':        14 obs. of  5 variables:
  ..$ chromosome: chr [1:14] "chr05" "chr05" "chr05" "chr05" ...
  ..$ pos1      : num [1:14] 6942364 12014448 15092332 29431780 2152617 ...
  ..$ pos2      : num [1:14] 6944728 12015959 15157600 29432466 2240909 ...
  ..$ methods   : chr [1:14] "BreakDancer:Delly" "BreakDancer:Delly" "BreakDancer:Delly" "BreakDancer:Del
  ..$ info      : chr [1:14] "score=99;PE=5:SU=5" "score=99;PE=6:SU=8" "score=99;PE=4:SU=4" "score=99;PE=
```

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':        686 obs. of  5 variables:
  ..$ chromosome: chr [1:686] "chr05" "chr05" "chr05" "chr05" ...
  ..$ pos1      : num [1:686] 65576 86455 120267 208217 280413 ...
  ..$ pos2      : num [1:686] 65949 87419 127693 208564 281198 ...
  ..$ methods   : chr [1:686] "BreakDancer:Delly" "BreakDancer:Delly" "BreakDancer:Delly" "BreakDancer:De
  ..$ info      : chr [1:686] "score=88;PE=5:SU=5" "score=99;PE=20:SU=21" "score=83;PE=6:SU=8" "score=99;
 $ 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:SU=23"
 $ inv:'data.frame':        13 obs. of  5 variables:
  ..$ chromosome: chr [1:13] "chr05" "chr05" "chr05" "chr05" ...
  ..$ pos1      : num [1:13] 6942364 12014448 15092332 29431780 2152617 ...
  ..$ pos2      : num [1:13] 6944728 12015959 15157600 29432466 2240909 ...
  ..$ methods   : chr [1:13] "BreakDancer:Delly" "BreakDancer:Delly" "BreakDancer:Delly" "BreakDancer:Del
  ..$ info      : chr [1:13] "score=99;PE=5:SU=5" "score=99;PE=6:SU=8" "score=99;PE=4:SU=4" "score=99;PE=
```

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/RtmpTNVj6Y/Rinst4ae7115d3df6/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  65576  65949        BreakDancer:Delly
2      chr05  86449  87419 BreakDancer:Pindel:Delly
3      chr05 120352 127768 BreakDancer:Pindel:Delly
4      chr05 208217 208564        BreakDancer:Delly
5      chr05 208217 208564        BreakDancer:Delly
6      chr05 280413 281198        BreakDancer:Delly
                                 info  tag  start    end strand
1                  score=88;PE=5:SU=5 <NA>     NA     NA   <NA>
2 score=99;PE=20:SR=11;score=42:SU=21 <NA>     NA     NA   <NA>
3     score=83;PE=6:SR=4;score=5:SU=8 <NA>     NA     NA   <NA>
4                 score=99;PE=9:SU=17 gene 207853 209693      -
5                 score=99;PE=9:SU=17 mRNA 207853 209693      -
6                 score=99;PE=7:SU=10 <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/RtmpTNVj6Y/Rinst4ae7115d3df6/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.

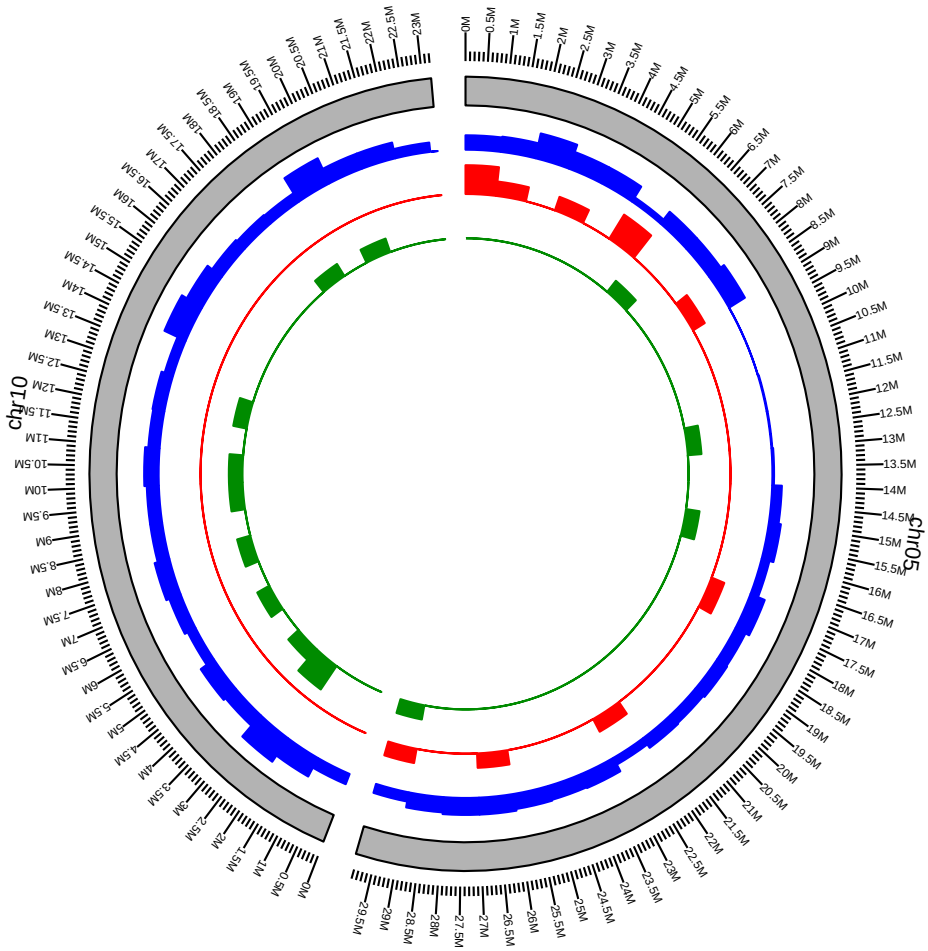

Figure 1: Visualization of SVs distribution in the whole genome. Blue: deletions, Red: duplications, Green: inversions.

3 Session Information

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

```
R version 4.0.3 (2020-10-10)
Platform: x86_64-pc-linux-gnu (64-bit)
Running under: Ubuntu 18.04.5 LTS

Matrix products: default
BLAS:   /home/biocbuild/bbs-3.12-bioc/R/lib/libRblas.so
LAPACK: /home/biocbuild/bbs-3.12-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
```

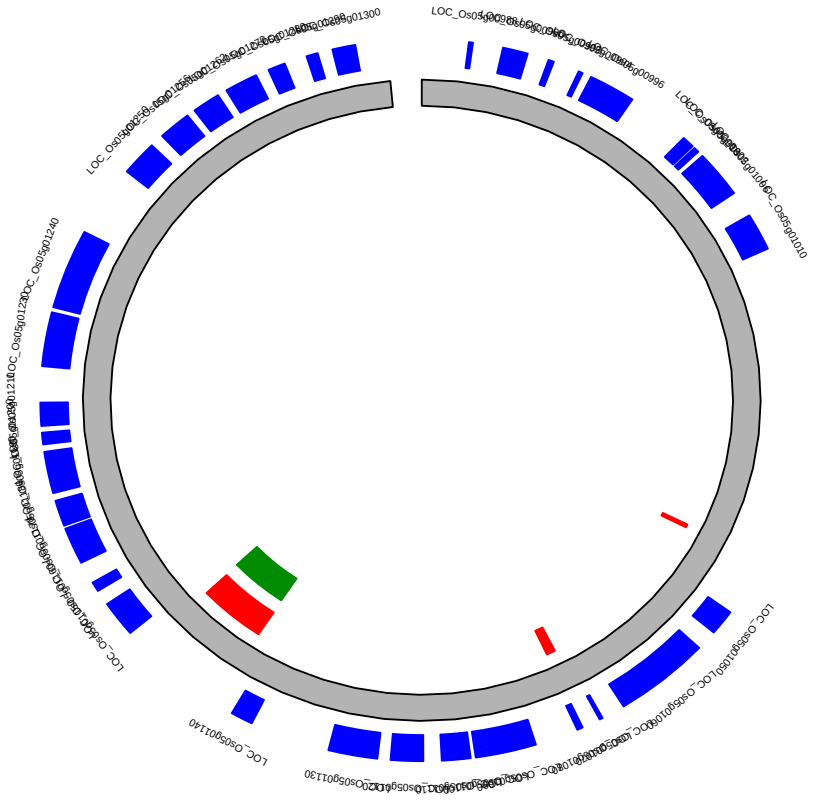

Figure 2: SVs in genomic region chr05:1-200000. Blue: genes, Red: deletions, Green: duplications, Purple: inversions.

```
 [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_1.30.0       GenomicRanges_1.42.0 GenomeInfoDb_1.26.0
[4] IRanges_2.24.0       S4Vectors_0.28.0     ggbio_1.38.0
[7] ggplot2_3.3.2        BiocGenerics_0.36.0  plyr_1.8.6

loaded via a namespace (and not attached):
  [1] ProtGenerics_1.22.0         bitops_1.0-6
  [3] matrixStats_0.57.0          bit64_4.0.5
  [5] RColorBrewer_1.1-2          progress_1.2.2
```

```
 [7] httr_1.4.2                  tools_4.0.3
 [9] backports_1.1.10            R6_2.4.1
[11] rpart_4.1-15                Hmisc_4.4-1
[13] DBI_1.1.0                   lazyeval_0.2.2
[15] colorspace_1.4-1            nnet_7.3-14
[17] withr_2.3.0                 tidyselect_1.1.0
[19] gridExtra_2.3               prettyunits_1.1.1
[21] GGally_2.0.0                bit_4.0.4
[23] curl_4.3                    compiler_4.0.3
[25] graph_1.68.0                Biobase_2.50.0
[27] htmlTable_2.1.0             xml2_1.3.2
[29] DelayedArray_0.16.0         labeling_0.4.2
[31] rtracklayer_1.50.0          scales_1.1.1
[33] checkmate_2.0.0             RBGL_1.66.0
[35] askpass_1.1                 rappdirs_0.3.1
[37] stringr_1.4.0               digest_0.6.27
[39] Rsamtools_2.6.0             foreign_0.8-80
[41] XVector_0.30.0              base64enc_0.1-3
[43] dichromat_2.0-0             jpeg_0.1-8.1
[45] pkgconfig_2.0.3             htmltools_0.5.0
[47] MatrixGenerics_1.2.0        ensembldb_2.14.0
[49] BSgenome_1.58.0             dbplyr_1.4.4
[51] htmlwidgets_1.5.2           rlang_0.4.8
[53] rstudioapi_0.11             RSQLite_2.2.1
[55] farver_2.0.3                generics_0.0.2
[57] BiocParallel_1.24.0         dplyr_1.0.2
[59] VariantAnnotation_1.36.0    RCurl_1.98-1.2
[61] magrittr_1.5                GenomeInfoDbData_1.2.4
[63] Formula_1.2-4               Matrix_1.2-18
[65] Rcpp_1.0.5                  munsell_0.5.0
[67] lifecycle_0.2.0             stringi_1.5.3
[69] SummarizedExperiment_1.20.0 zlibbioc_1.36.0
[71] BiocFileCache_1.14.0        grid_4.0.3
[73] blob_1.2.1                  crayon_1.3.4
[75] lattice_0.20-41             Biostrings_2.58.0
[77] splines_4.0.3               GenomicFeatures_1.42.0
[79] hms_0.5.3                   knitr_1.30
[81] pillar_1.4.6                reshape2_1.4.4
[83] biomaRt_2.46.0              XML_3.99-0.5
[85] glue_1.4.2                  biovizBase_1.38.0
[87] latticeExtra_0.6-29         BiocManager_1.30.10
[89] data.table_1.13.2           png_0.1-7
[91] vctrs_0.3.4                 gtable_0.3.0
[93] openssl_1.4.3               purrr_0.3.4
[95] reshape_0.8.8               assertthat_0.2.1
[97] xfun_0.18                   AnnotationFilter_1.14.0
[99] survival_3.2-7              OrganismDbi_1.32.0
[101] tibble_3.0.4               GenomicAlignments_1.26.0
[103] AnnotationDbi_1.52.0       memoise_1.1.0
[105] cluster_2.1.0              ellipsis_0.3.1
```

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. St??tz, 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/`.

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. St??tz, Enrico Cannav??, 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.