Comprehensive evaluation and guidance of structural variation detection tools in chicken whole genome sequence data

Cheng Ma[1,2,†], Xian Shi[1,3,†], Xuzhen Li[4,5,†], Ya-Ping Zhang[1,3,6,7,*], Min-Sheng Peng[1,3,7,*]

1. Key Laboratory of Genetic Evolution & Animal Models and Yunnan Key Laboratory of Molecular Biology of Domestic Animals, Kunming Institute of Zoology, Chinese Academy of Sciences, Kunming 650223, China.
2. Department of Medical Biochemistry and Microbiology, Uppsala University, BMC, Uppsala SE-75123, Sweden.
3. University of Chinese Academy of Sciences, Beijing 100049, China.
4. State Key Laboratory for Conservation and Utilization of Bio-Resources in Yunnan, Yunnan Agricultural University, Kunming 650201, China.
5. College of Biological Big Data, Yunnan Agriculture University, Kunming 650201, China.
6. State Key Laboratory for Conservation and Utilization of Bio-Resources in Yunnan, Yunnan University, Kunming 650091, China.
7. KIZ-CUHK Joint Laboratory of Bioresources and Molecular Research in Common Diseases, Kunming Institute of Zoology, Chinese Academy of Sciences, Kunming 650223, China.

† These authors contributed equally to this work.

* Correspondence: Tel: +86 871-68526519; Fax: +86 871-68526519; Email: pengminsheng@mail.kiz.ac.cn or zhangyp@mail.kiz.ac.cn

Supplementary Materials and Methods

SV calling

For each sample, ten selected tools were used to perform SV calling, and for each tool, the data processing process was consistent, the bam file was used as input, and the vcf file was generated as the final output results. For MetaSV [1], which merges SV calling results from other multiple tools, four tools were used as sources for integration, including Pindel [2], Manta [3], Lumpy [4] and Wham [5]. The detailed usage and parameter settings for each software are provided.

In the script below, "${ID}" represents each sample ID; "${ref6}" represents the chicken reference genome of GRCg6a, "$t" represents the number of threads used in data processing, and "./" represents the current directory. "*.bam" represents all the bam files, and "Control" represents the control group. Here, we used the ancestor of domestic chicken, *Gallus gallus spadiceus* subspecies of Red junglefowl (n=20), as a background control for these case–control-based tools (CNVkit [6], and Delly [7]). The meaning of the parameters of each software can be checked through the website (listed below) and the manual of the software, and the parameters were set by referring to the article of [8].

1. BreakDancer [9] (https://github.com/genome/breakdancer)

```
## Step 1. Create a configuration file
bam2cfg.pl -v 2.5 ${ID}.bam > ${ID}.config.txt

## Step 2. SV calling
breakdancer-max -y 20 -x 200 -r 3 -m 10000000 ${ID}.config.txt > ${ID}.breakdancer.sv

## step 3. Convert file format
java -jar breakdancer2vcf.jar -R ${ref6} ${ID}.breakdancer.sv --out ${ID}.breakdancer.vcf
```

2. CNVkit [6] **(https://github.com/etal/cnvkit)**

```
#for each breed, run the following two-step program once.
#"${pop}" represents different breed IDs.
##step 1. access
cnvkit.py access ${ref6} -o ${pop}.access.bed

##step 2. batch
#${refFlat} represents the flat format of the chicken annotation file.
cnvkit.py batch --method wgs -p $t ${pop}.*.bam --normal control. *.bam --fasta ${ref6} --annotate
${refFlat} --access ${pop}.access.bed --output-reference ${pop}_reference_ggs.cnn --output-dir ./
--scatter --diagram

#for each sample, run the following programs.
##step 3. coverage
```

```
cnvkit.py coverage -p $t ${ID}.bam ${pop}.access.target.bed -o ${ID}.target_coverage.cnn

cnvkit.py coverage -p $t ${ID}.bam ${pop}.access.antitarget.bed -o
${ID}.antitarget_coverage.cnn

##step 4. fix
cnvkit.py fix ${ID}.target_coverage.cnn ${ID}.antitarget_coverage.cnn
${pop}_reference_ggs.cnn -o ${ID}.cnr

##step 5. segment
cnvkit.py segment -p $t --method cbs ${ID}.cnr -o ${ID}.cns

##step 6. SV call
cnvkit.py call ${ID}.cns --drop-low-coverage --sample-id ${ID} --filter cn -o ${ID}.call.cns

##step 7. export vcf
cnvkit.py export vcf ${ID}.call.cns -i ${ID} -o ${ID}.call.vcf
```

3. CNVnator [10] (https://github.com/abyzovlab/CNVnator)

```
##step 1. extract read mapping
cnvnator -root ${ID}.root -tree ${ID}.bam -chrom `seq 1 28` 30 31 32 33 Z W –unique

##step 2. generate histogram
#${GRCg6a_dir} represents the folder path of the chicken reference file.
cnvnator -root ${ID}.root -his 1000 -d ${GRCg6a_dir}

##step 3. statistics
cnvnator -root ${ID}.root -chrom `seq 1 28` 30 31 32 33 Z W -stat 1000

##step 4. partition
cnvnator -root ${ID}.root -chrom `seq 1 28` 30 31 32 33 Z W -partition 1000

##step 5. call CNVs
cnvnator -root ${ID}.root -chrom `seq 1 28` 30 31 32 33 Z W -call 1000 > ${ID}.cnvnator.cnv

##step 6. cnv2vcf
cnvnator2VCF.pl -prefix ${ID} -reference GRCg6a ${ID}.cnvnator.cnv ${GRCg6a_dir} >
${ID}.cnvnator.vcf
```

4. Delly [7] (https://github.com/dellytools/delly)

```
##step 1. SV call
delly call --svtype ALL -g ${ref6} -o ${ID}.delly.all.bcf ${ID}.bam Control.*.bam
```

```
##step 2. SV prefiltering
#"${ID}_and_control_list.tsv" is a list of sample IDs and their groups.
delly filter -f somatic -o ${ID}.delly.all.pre.bcf -s ${ID}_and_control_list.tsv ${ID}.delly.all.bcf

##step 3. SV genotyping
delly call -g ${ref6} -v ${ID}.delly.all.pre.bcf -o ${ID}.delly.all.pre.geno.bcf ${ID}.bam
Control.*.bam

##step 4. SV filtering
delly filter -f somatic -o ${ID}.delly.all.pre.geno.filtered.bcf -s ${ID}_and_control_list.tsv
${ID}.delly.all.pre.geno.bcf

##step 5. bcf2vcf
bcftools view ${ID}.delly.all.pre.geno.filtered.bcf > ${ID}.delly.all.pre.geno.filtered.vcf
```

5. GRIDSS [11] (https://github.com/PapenfussLab/gridss)

```
##step 1. SV call
gridss.sh -t $t -r ${ref6} -o ${ID}.gridss.vcf -a ${ID}.gridss.assembly.bam -j grids.jar --jvmheap 5g
${ID}.bam

##step 2. vcf filter
vcftools --vcf ${ID}.gridss.vcf --recode --recode-INFO-all --remove-filtered-all --out
${ID}.gridss.filtered
```

6. Lumpy [4] (https://github.com/brentp/smoove)

```
##step 1. SV call
smoove call --outdir ./ --name ${ID} --fasta ${ref6} -p $t --genotype ${ID}.bam

##step 2. SV call
#"${gff_file}" is the chicken annotation file in gff file format.
smoove annotate --gff ${gff_file} ${ID}-smoove.genotyped.vcf | bgzip -c >
${ID}.smoove.genotyped.anno.vcf.gz
```

7. Manta [3] (https://github.com/Illumina/manta)

```
## step 1. config
configManta.py --bam ${ID}.bam --referenceFasta ${ref6} --runDir ./

## step 2. Call SVs
./runWorkflow.py -j $t -g 5
```

8. MetaSV [1] (http://bioinform.github.io/metasv)

```
##step 1. call SVs
run_metasv.py--reference ${ref6} --bam ${ID}.bam --outdir ./ --sample ${ID} --num_threads $t --
mean_read_length 150 --minsvlen 1 --maxsvlen 100000000 --min_support_ins 3 --
mean_read_coverage 10 --chromosomes
{1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,30,31,32,33,Z,W} --
spades /$Path_to_MetaSV/bin/ --age /$Path_to_MetaSV/bin/ --workdir ./ --pindel_vcf
${ID}.Pindel.vcf --manta_vcf ${ID}.Manta.vcf.gz --lumpy_vcf ${ID}.smoove.vcf.gz --wham_vcf
${ID}.Wham.vcf --disable_assembly
```

9. Pindel [2] (https://github.com/genome/pindel)

```
##step 1. call SVs
echo "${ID}.bam 300 ${ID}" > ${ID}_config.txt

pindel --fasta ${ref6} --config-file ${ID}_config.txt --chromosome ALL --number_of_threads $t -
-window_size 10 --min_num_matched_bases 50 --min_inversion_size 100 --
minimum_support_for_event 3 --sensitivity 0.92 --max_range_index 2 --
MIN_DD_MAP_DISTANCE 3000 --output-prefix ${ID}

##step 2. Pindel2vcf
pindel2vcf -P ${ID} -r ${ref6} -R GRCg6a -d 201803 --min_coverage 10 --min_supporting_reads
4 --gatk_compatible --compact_output_limit 100 -v ${ID}.vcf
```

10. Wham [5] (https://github.com/zeeev/wham)

```
##step 1. call SVs
whamg -f ${ID}.bam -a ${ref6} -x $t -c
1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,30,31,32,33,Z,W,MT
> ${ID}.Wham.vcf
```

References

1. Mohiyuddin M, Mu JC, Li J, Bani Asadi N, Gerstein MB, Abyzov A, Wong WH, Lam HY: **MetaSV: an accurate and integrative structural-variant caller for next generation sequencing**. *Bioinformatics* 2015, **31**(16):2741-2744.
2. Ye K, Schulz MH, Long Q, Apweiler R, Ning Z: **Pindel: a pattern growth approach to detect break points of large deletions and medium sized insertions from paired-end short reads**. *Bioinformatics* 2009, **25**(21):2865-2871.
3. Chen X, Schulz-Trieglaff O, Shaw R, Barnes B, Schlesinger F, Källberg M, Cox AJ, Kruglyak S, Saunders CT: **Manta: rapid detection of structural variants and indels for germline**

and cancer sequencing applications. *Bioinformatics* 2016, **32**(8):1220-1222.

4. Layer RM, Chiang C, Quinlan AR, Hall IM: **LUMPY: a probabilistic framework for structural variant discovery**. *Genome Biology* 2014, **15**(6):R84.
5. Kronenberg ZN, Osborne EJ, Cone KR, Kennedy BJ, Domyan ET, Shapiro MD, Elde NC, Yandell M: **Wham: Identifying Structural Variants of Biological Consequence**. *PLoS Computational Biology* 2015, **11**(12):e1004572.
6. Talevich E, Shain AH, Botton T, Bastian BC: **CNVkit: Genome-Wide Copy Number Detection and Visualization from Targeted DNA Sequencing**. *PLoS Computational Biology* 2016, **12**(4):e1004873.
7. Rausch T, Zichner T, Schlattl A, Stütz AM, Benes V, Korbel JO: **DELLY: structural variant discovery by integrated paired-end and split-read analysis**. *Bioinformatics* 2012, **28**(18):i333-i339.
8. Kosugi S, Momozawa Y, Liu X, Terao C, Kubo M, Kamatani Y: **Comprehensive evaluation of structural variation detection algorithms for whole genome sequencing**. *Genome Biology* 2019, **20**(1):117.
9. Chen K, Wallis JW, McLellan MD, Larson DE, Kalicki JM, Pohl CS, McGrath SD, Wendl MC, Zhang Q, Locke DP *et al*: **BreakDancer: an algorithm for high-resolution mapping of genomic structural variation**. *Nature Methods* 2009, **6**(9):677-681.
10. Abyzov A, Urban AE, Snyder M, Gerstein M: **CNVnator: an approach to discover, genotype, and characterize typical and atypical CNVs from family and population genome sequencing**. *Genome Research* 2011, **21**(6):974-984.
11. Cameron DL, Schröder J, Penington JS, Do H, Molania R, Dobrovic A, Speed TP, Papenfuss AT: **GRIDSS: sensitive and specific genomic rearrangement detection using positional de Bruijn graph assembly**. *Genome Research* 2017, **27**(12):2050-2060.