NucBreak: Location of structural errors in a genome assembly by using paired-end Illumina reads

Supplementary materials

Ksenia Khelik, Geir Kjetil Sandve, Alexander Johan Nederbragt, Torbjørn Rognes

Supplementary methods and results

1. Fragment size estimation

Only read pairs satisfying the following conditions are used for fragment size estimation:

1. Each read in a pair is uniquely aligned
2. Both reads are mapped to the same genome sequence
3. The reads have different orientations relative to the genome sequence
4. The read with the reverse orientation is located at the same position or further down on the sequence compared to the mapping locations of the forward-oriented read
5. The forward- and reverse-oriented reads are not soft-clipped at both sides. However, the alignments of properly mapped reads may contain short substitutions, insertions and deletions.

The fragment size is calculated by the formula:

$frag\_size = ref\_end_2 - ref\_st_1 + 1$, where

ref_end_2 - the location of reverse-oriented read end at the genome chromosome
ref_st_1 - the location of forward-oriented read start at the genome chromosome

The fragment sizes are sorted in ascending order, and for each fragment size the number of read pairs ($\#P$) having the given fragment size is calculated. Then min_frag_size and max_frag_size are found:

$$min\_frag\_size = \{frag\_size_i : \#P(frag\_size_i) \geq 10 \; and \; \forall \, j \; < i \;\; \#P(frag\_size_j) < 10 \\ and \; \exists k = i + 1 \ldots i + 10 \;\; \#P(frag\_size_k) \geq 10 \; and \; \#k \geq 3\}$$

$$max\_frag\_size = \{frag\_size_i : \#P(frag\_size_i) \geq 10 \; and \; \forall \, j \; > i \;\; \#P(frag\_size_j) < 10 \\ and \; \exists k = i - 11 \ldots i - 1 \;\; \#P(frag\_size_k) \geq 10 \; and \; \#k \geq 3\}$$

If the number of corresponding read pairs is less than 10 for any fragment size, then:

$$min\_frag\_size = \{\max(0, frag\_size_i - 50) : \forall k \neq i \; \#P(frag\_size_i) \geq \; \#P(frag\_size_k)\}$$
$$max\_frag\_size = \{frag\_size_i \; + 50 : \forall k \neq i \; \#P(frag\_size_i) \geq \; \#P(frag\_size_k)\}$$

2. Fragment size detection between properly mapped read pairs

Since the reads from properly mapped reads pairs may be soft-clipped in the start or at the end of the read depending on the read orientation, a fragment size inside properly mapped reads is calculated by the extended formula:

$frag\_size = ref\_end_2 + end\_clipped\_dist_2 - ref\_st_1 - start\_clipped\_dist_1 + 1$, where

ref_end_2 - the location of the reverse-oriented read end at the genome chromosome
$end\_clipped\_dist_2$ - the number of soft-clipped bases at the end of the reverse-oriented read
ref_st_1 - the location of the forward-oriented read start at the genome chromosome
$start\_clipped\_dist_1$- the number of soft-clipped bases in the beginning of the reverse-oriented read

3. The Velvet, ABySS and SPAdes parameter settings used to obtain assemblies

SPAdes was run with the "-t 2 -k 33 --cov-cutoff 2" parameter settings.
ABySS was run with "k=64" parameter setting.
Velveth was run with k-mer length equal to 31.
Velvetg was run with "-ins_length 180 -scaffolding yes -min_contig_lgth 250 -cov_cutoff 5" parameter settings.

4. The NucBreak, REAPR and FRCbam parameter settings used to detect assembly errors

In the Sections 3.1 and 3.2, we used the following parameter settings for the tools:

- NucBreak was run with "--min_frag_size 620 --max_frag_size 790" parameter settings
- In case of REAPR, perfectmap was run with 700 bp average insert size
- FRCbam was run with "--pe-max-insert 776" and the value for "--genome-size" parameter was detected automatically by using python script for each modification case.

In the Section 3.3, we used the following parameter settings for the tools:

- in case of REAPR, perfectmap was run with 300 bp average insert size
- FRCbam was run with "--pe-max-insert 776 --genome-size 112000000" parameter settings

In the Section 3.4, we used the following parameter settings for the tools:

- In case of REAPR, perfectmap was run with the following average insert sizes depending on the genome dataset used:
    - Salmonella dataset - 500 bp
    - Staphylococcus dataset - 400 bp
    - Escherichia dataset - 300 bp
    - Pseudomonas dataset - 180 bp
    - Bordetella dataset - 450 bp
    - Brucella dataset - 500 bp
    - Klebsiella dataset - 200 bp
    - Enterobacter dataset - 300 bp
- FRCbam was run with the following parameter settings depending on the genome dataset used:
    - Salmonella dataset - "--pe-max-insert 1060 --genome-size 4810000"
    - Staphylococcus dataset - "--pe-max-insert 1040 --genome-size 2860000"
    - Escherichia dataset - "--pe-max-insert 1110 --genome-size 5480000"
    - Pseudomonas dataset - "--pe-max-insert 844 --genome-size 6820000"
    - Bordetella dataset - "--pe-max-insert 890 --genome-size 4110000"
    - Brucella dataset - "--pe-max-insert 1120 --genome-size 3300000"
    - Klebsiella dataset - "--pe-max-insert 950 --genome-size 5720000"
    - Enterobacter dataset - "--pe-max-insert 819 --genome-size 5040000"

5. Result evaluation

The ground truth entries may be represented as dots (e.g. in case of deletions, simple relocations or translocations) or as intervals (e.g. in case of insertion, duplications, relocations with overlap). If a ground truth entry is an interval, it may be fully covered with reads mapped back to the query sequences (e.g. in case of inversions) or remain uncovered (e.g. in case of inserted regions that are not present in the reference genome). In the first case, a tested tool is expected to mark the regions corresponding to the start- and/or end-points of the ground truth entry as breakpoints, while in the second case the whole entry is expected to be predicted as a breakpoint.

We say that if a ground truth entry coincides with an obtained breakpoint or the ground truth entry start- and/or end-points coincide with obtained breakpoints, then we have a true positive (TP). If a ground truth entry does not coincide with any of obtained breakpoints, then we have a false negative (FN). To get TPs and FNs, we have run BEDTools with the pairtopair -both' option. With this option, BEDTool reports an overlap between two intervals A and B if both ends of A overlap B. If BEDTool reports an overlap for a whole ground truth entry or for its start- and/or end-points, then we get a TP, otherwise a FN. Having obtained the number of TPs and FNs, we calculate sensitivity by the formula:

$$Sensitivity = \frac{\#TP}{\#TP + \#FN}$$

Unlike ground truth entries, an obtained result can correspond only to one interval: either to a whole ground truth entry or to its start- or end-point. If an obtained breakpoint does not coincide with any of the ground truth entries and with any of the ground truth entry start- and end-points, then the given obtained breakpoint is a false positive (FP). To get FP, we have run BEDTools with the ‘‘pairtopair -notboth’ option. With this option, BEDTool reports an overlap between two intervals A and B, if one or neither of A's ends overlap B. If BEDTool reports an overlap for an obtained breakpoint with a whole ground truth entry or with its ends, then we get a FP. Having obtained the number of FPs, we calculate FDR by the formula:

$$FDR = \frac{FP}{FP + TP}$$

Supplementary figures

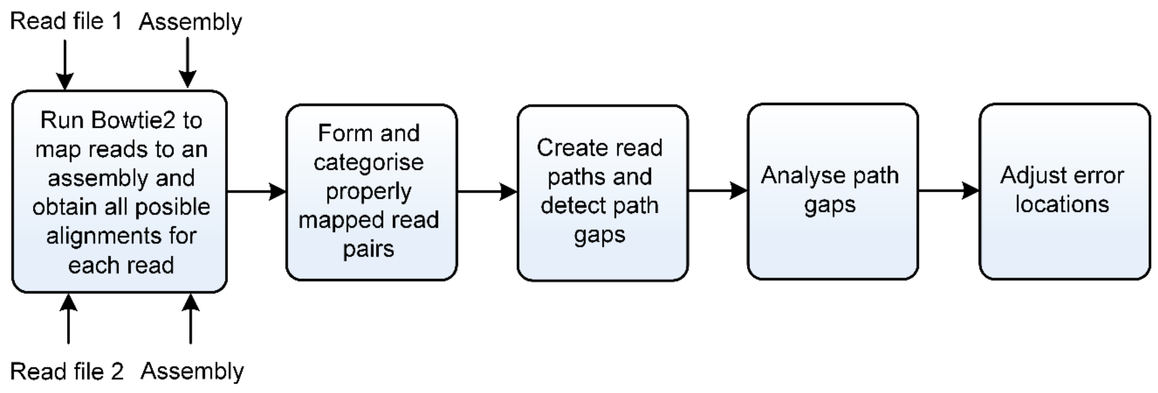

Read file 1
Assembly
Run Bowtie2 to map reads to an assembly and obtain all posible alignments for each read
Read file 2
Assembly
Form and categorise properly mapped read pairs
Create read paths and detect path gaps
Analyse path gaps
Adjust error locations

Figure S1 NucBreak workflow

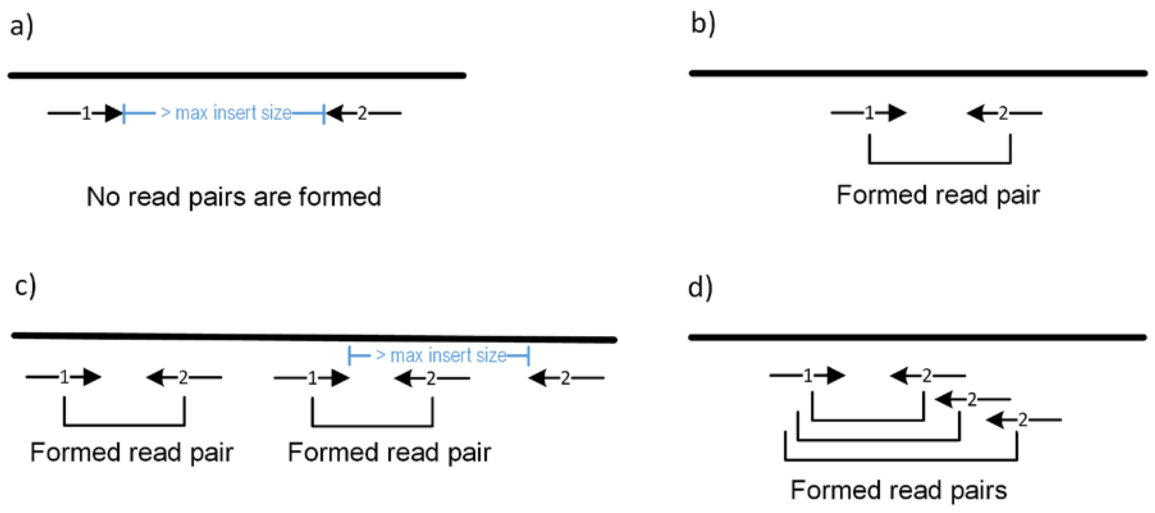


Figure S2 Properly mapped read pair formation. The black line represents an assembly. The arrows represent all possible read mapping locations. The cases a) and b) correspond to the situations when no read pairs are formed or just one read pair is formed, respectively. The cases c) and d) show examples when several read pairs are formed from two given reads. The case d) is an example of the situation when reads are mapped to a tandem repeat.

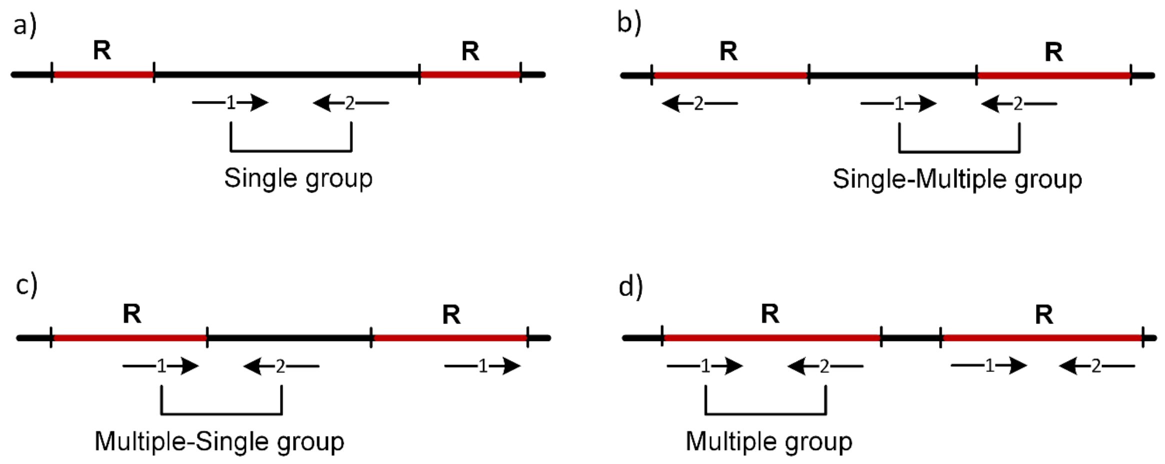


Figure S3 Properly mapped read pair categorization. The black line represents an assembly. The assembly regions marked by red colour correspond to repeated regions. The repeated regions are identical or near-identical copies of the same repeat. The arrows represent all possible read mapping locations.

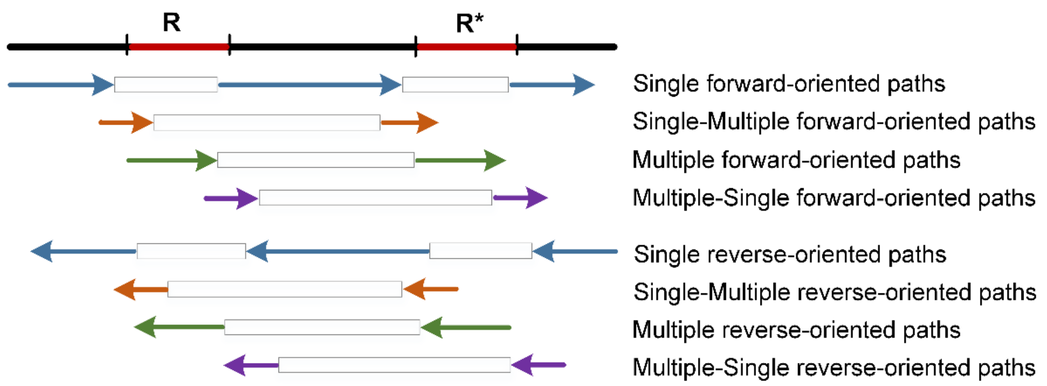
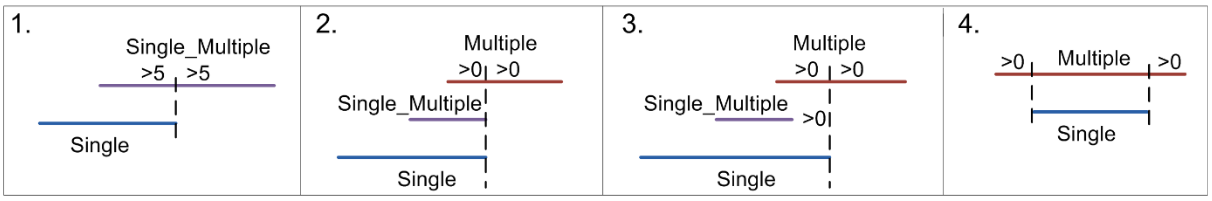
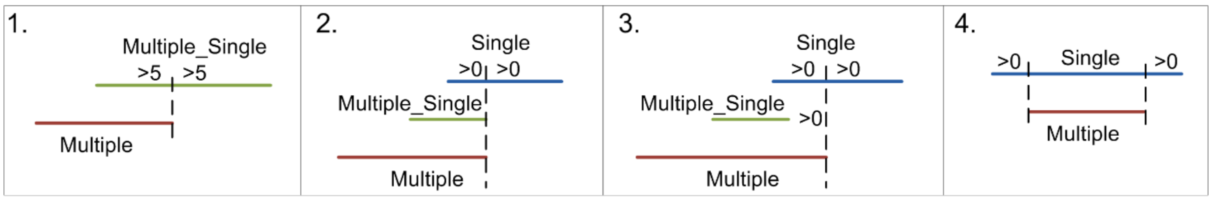

R
R*
Single forward-oriented paths
Single-Multiple forward-oriented paths
Multiple forward-oriented paths
Multiple-Single forward-oriented paths
Single reverse-oriented paths
Single-Multiple reverse-oriented paths
Multiple reverse-oriented paths
Multiple-Single reverse-oriented paths

Figure S4 Read paths and path gaps. The black line represents an assembly. The assembly regions marked by red colour correspond to repeated regions. The repeated regions are identical or near-identical copies of the same repeat or copies of different repeats. The arrows represent read paths. The arrows of the same colour correspond to the read paths of the same type. The rectangles between the read paths indicate path gaps. The example demonstrates the correct order of the read paths in the absence of assembly errors.

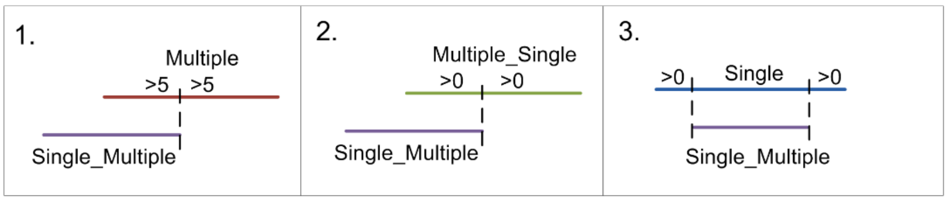
Single path



Multiple path



Single_Multiple path



Multiple_Single path

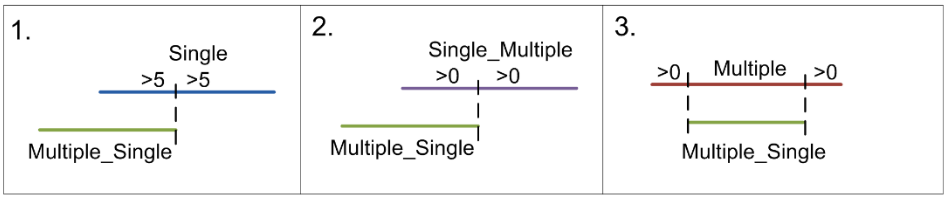


Figure S5 Possible type order and locations of read paths in the absence of breakpoints.

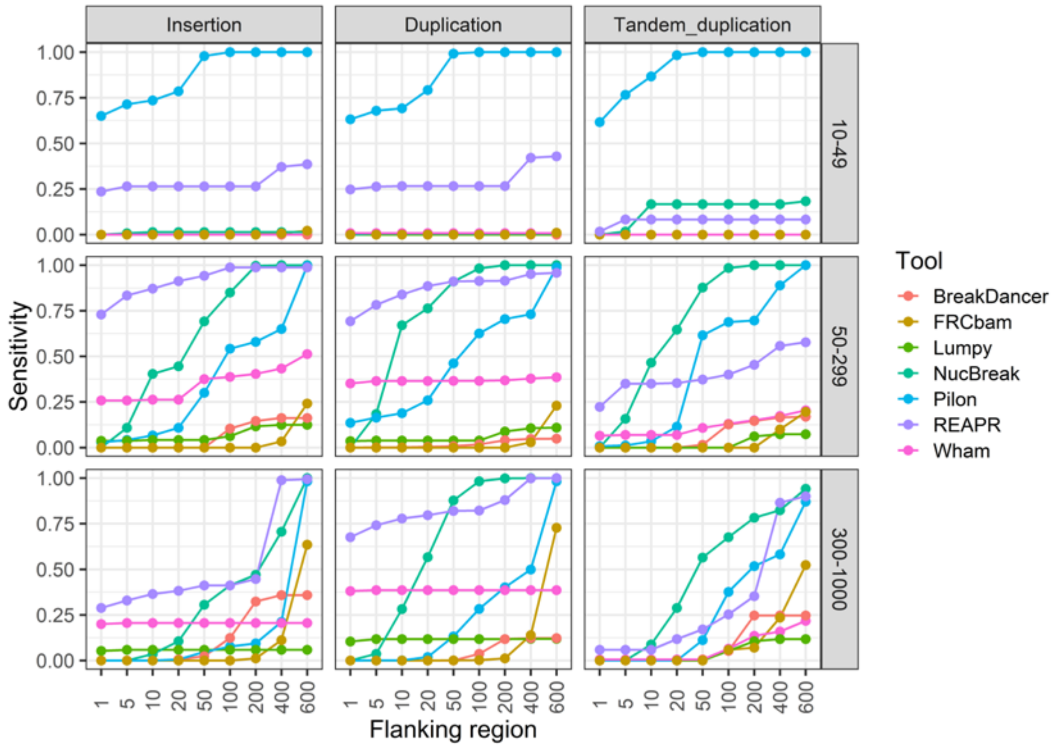


Figure S6 Sensitivity results for the insertion, duplication and tandem duplication groups, obtained using the simulated datasets.

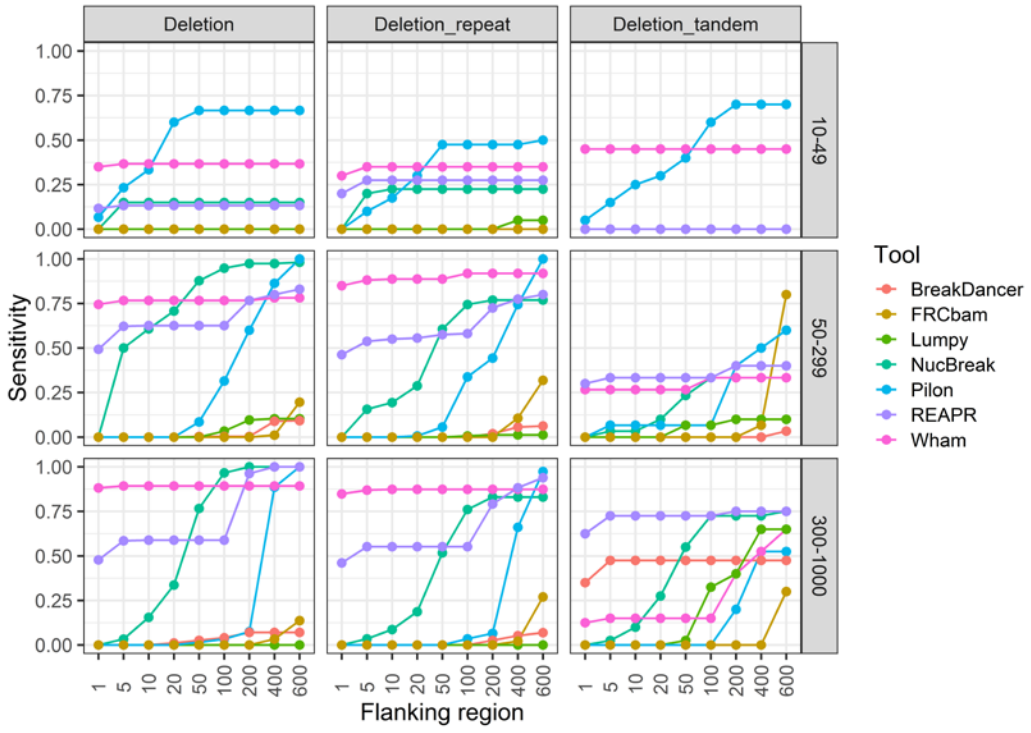


Figure S7 Sensitivity results for the deletion, deletion_repeat and deletion_tandem groups, obtained using the simulated datasets. The deletion_repeat group contains deletions of interspersed repeats or their parts. The deletion_tandem group contains deletions of tandem repeats or their parts.

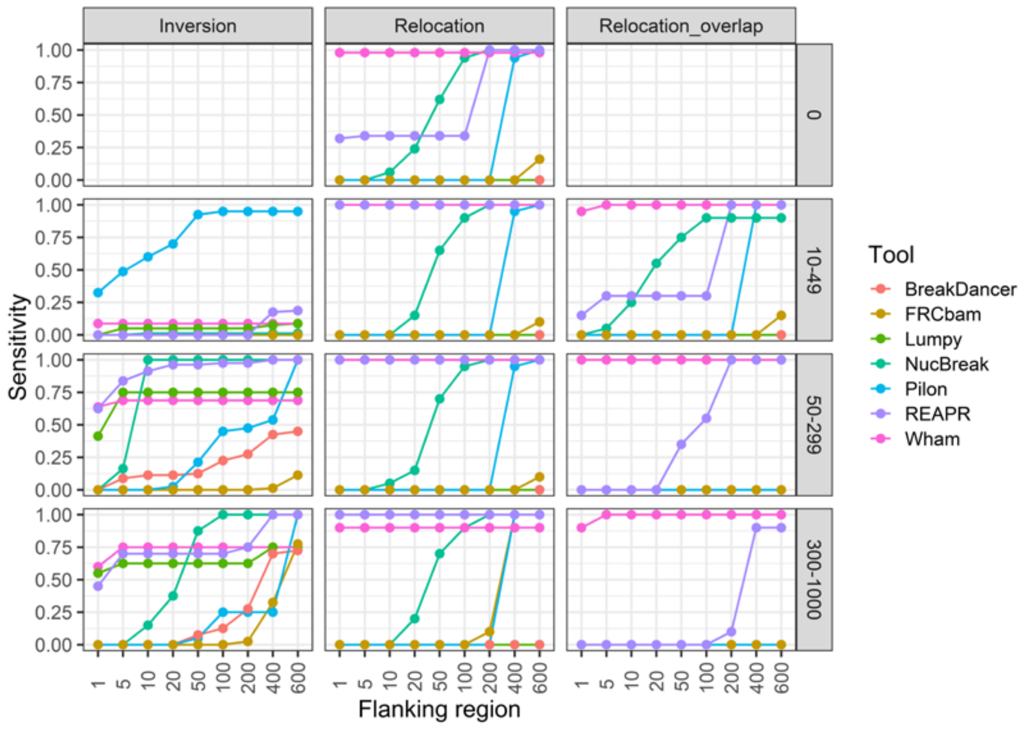


Figure S8 Sensitivity results for the inversion, relocation and relocation_overlap groups, obtained using the simulated datasets. The relocation group consists of relocations with either inserted regions between misjoined regions (size varied between 10 and 1000) or without them (size is equal to 0). The relocation_overlap group consists of relocations with overlapped misjoined regions.

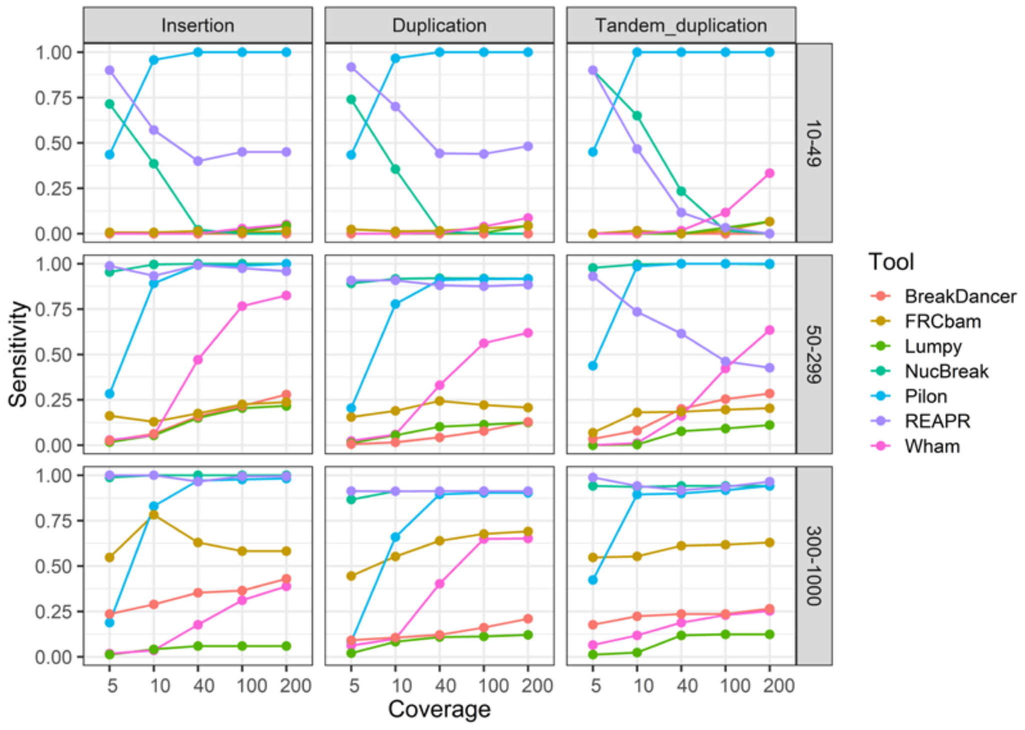


Figure S9 Sensitivity results for the insertion, duplication and tandem duplication groups, obtained using the simulated datasets.

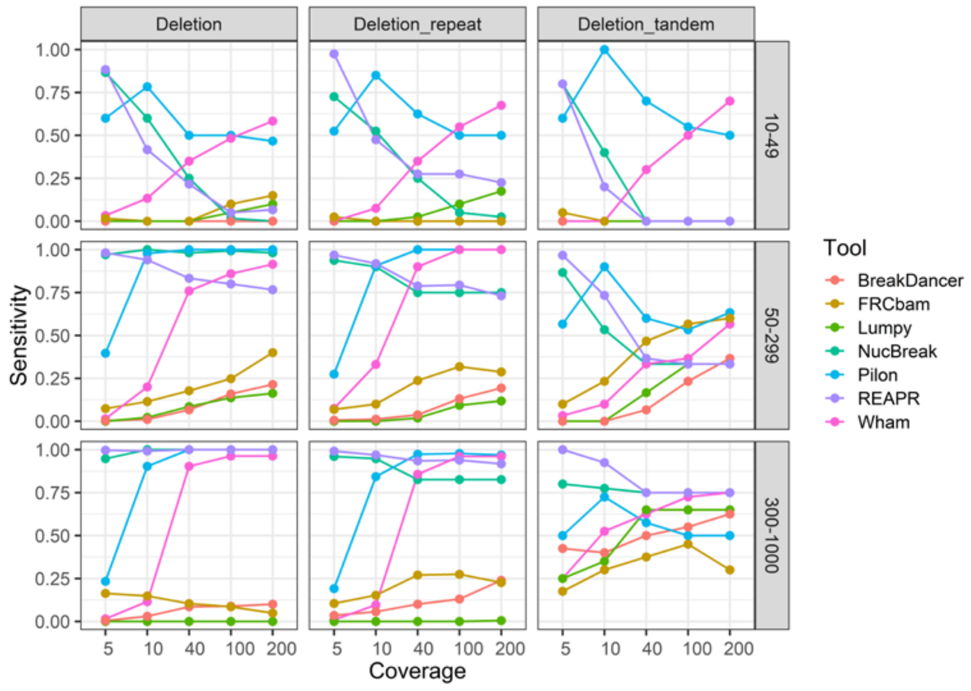


Figure S10 Sensitivity results for the deletion, deletion_repeat and deletion_tandem groups, obtained using the simulated datasets. The deletion_repeat group contains deletions of interspersed repeats or their parts. The deletion_tandem group contains deletions of tandem repeats or their parts.

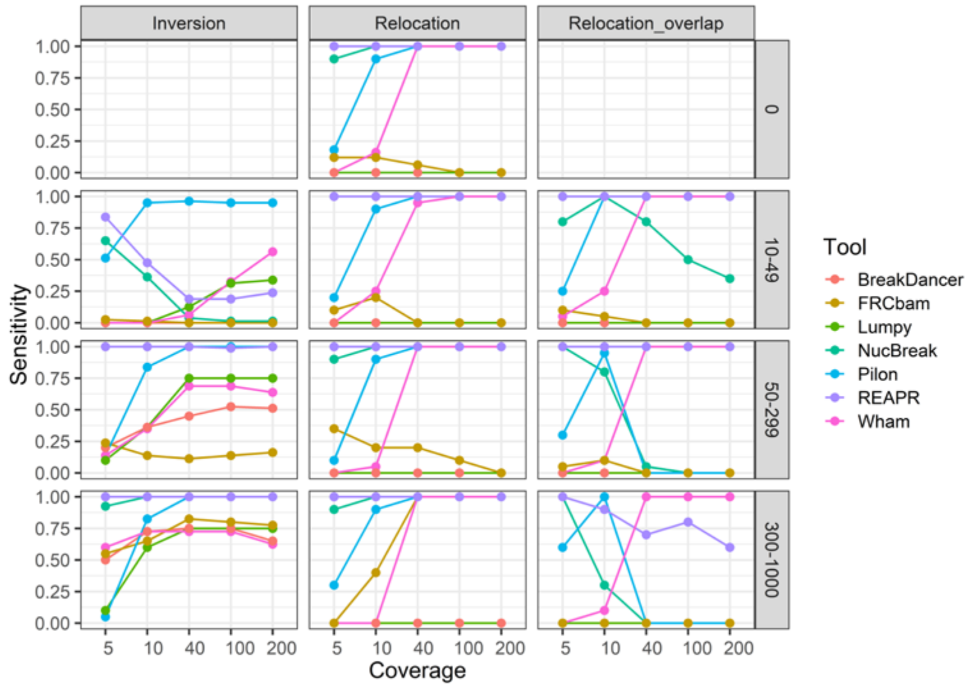


Figure S11 Sensitivity results for the inversion, relocation and relocation_overlap groups, obtained using the simulated datasets. The relocation group consists of relocations with either inserted regions between misjoined regions (size varied between 10 and 1000) or without them (size is equal to 0). The relocation_overlap group consists of relocations with overlapped misjoined regions.

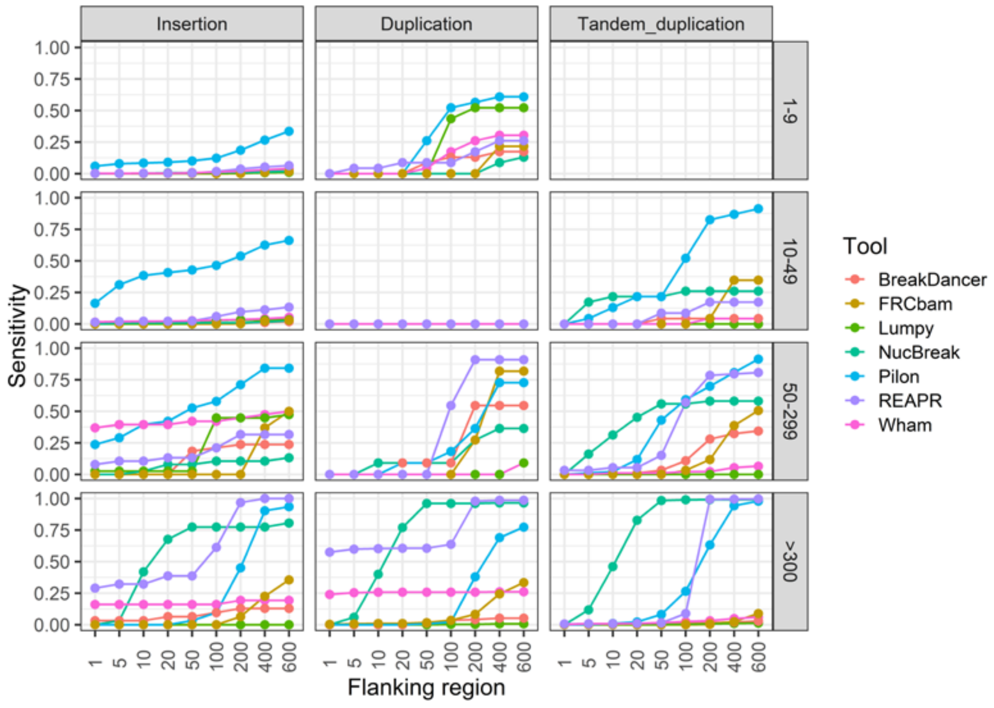


Figure S12 Sensitivity results for the insertion, duplication and tandem duplication groups, obtained using the datasets from the Assemblathon 1 project.

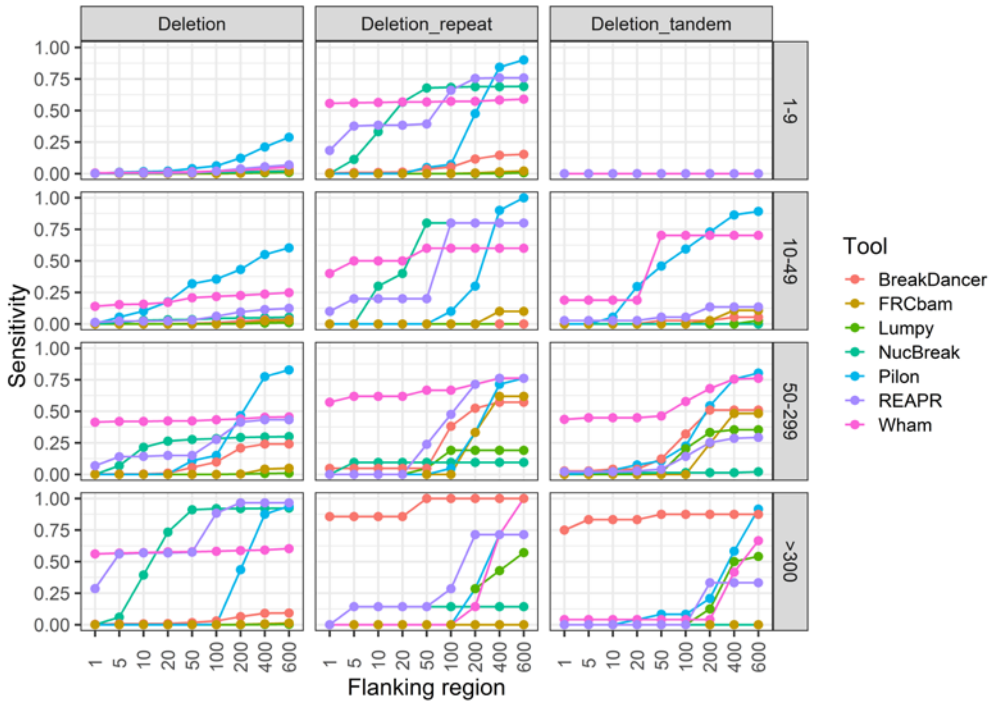


Figure S13 Sensitivity results for the deletion, deletion_repeat and deletion_tandem groups, obtained using the datasets from the Assemblathon 1 project. The deletion_repeat group contains deletions of interspersed repeats or their parts. The deletion_tandem group contains deletions of tandem repeats or their parts.

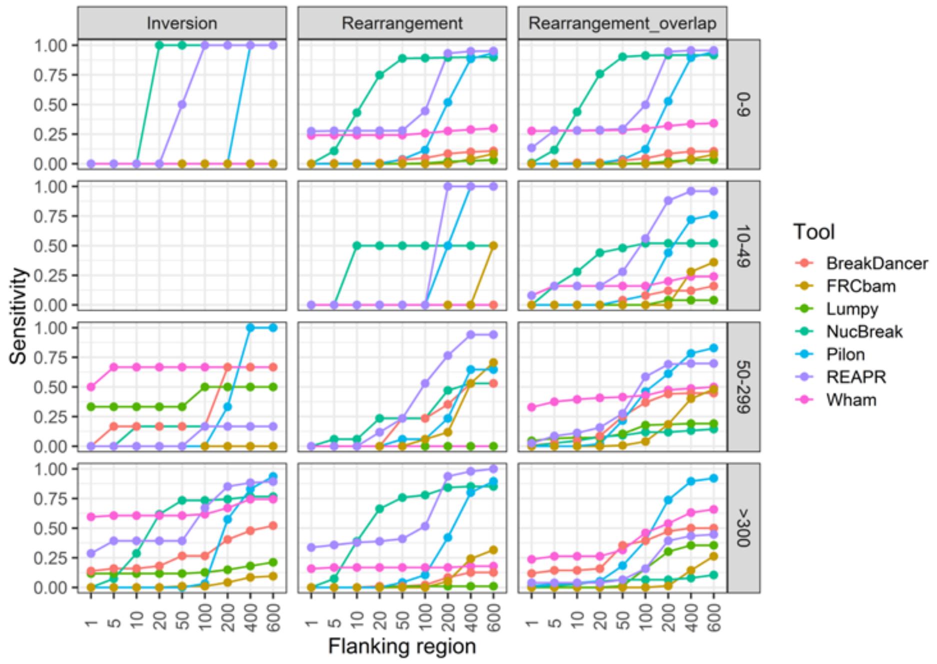


Figure S14 Sensitivity results for the inversion, rearrangement and rearrangement_overlap groups, obtained using the datasets from the Assemblathon 1 project.The rearrangement group consists of relocations and translocations with either inserted regions between misjoined regions (size varied between 1 and 1000) or without them (size is equal to 0). The rearrangement_overlap group consists of relocations and translocations with overlapped misjoined regions.

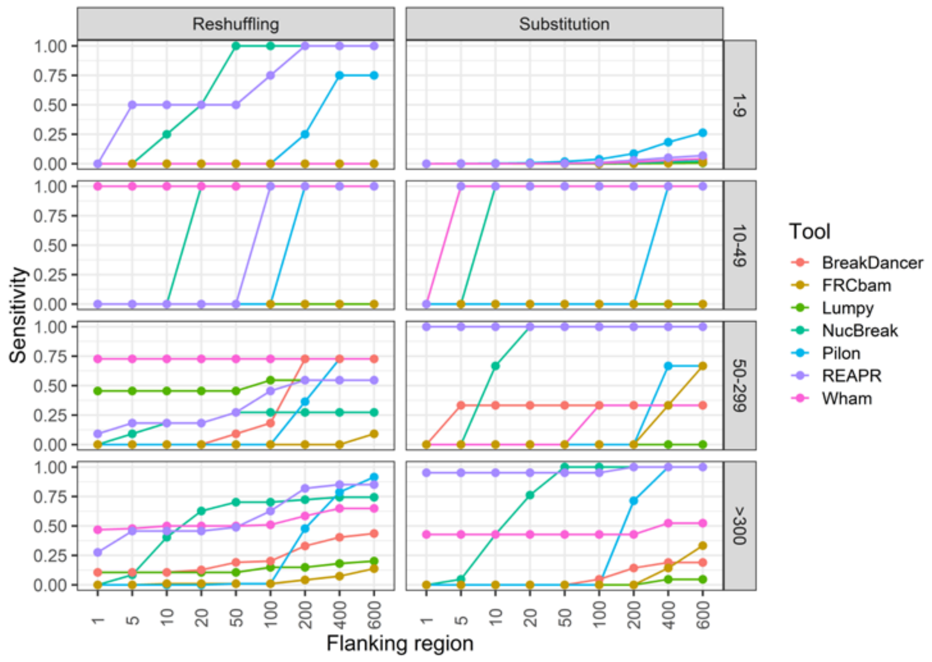


Figure S15 Sensitivity results for the reshuffling and substitution groups, obtained using the datasets from the Assemblathon 1 project.

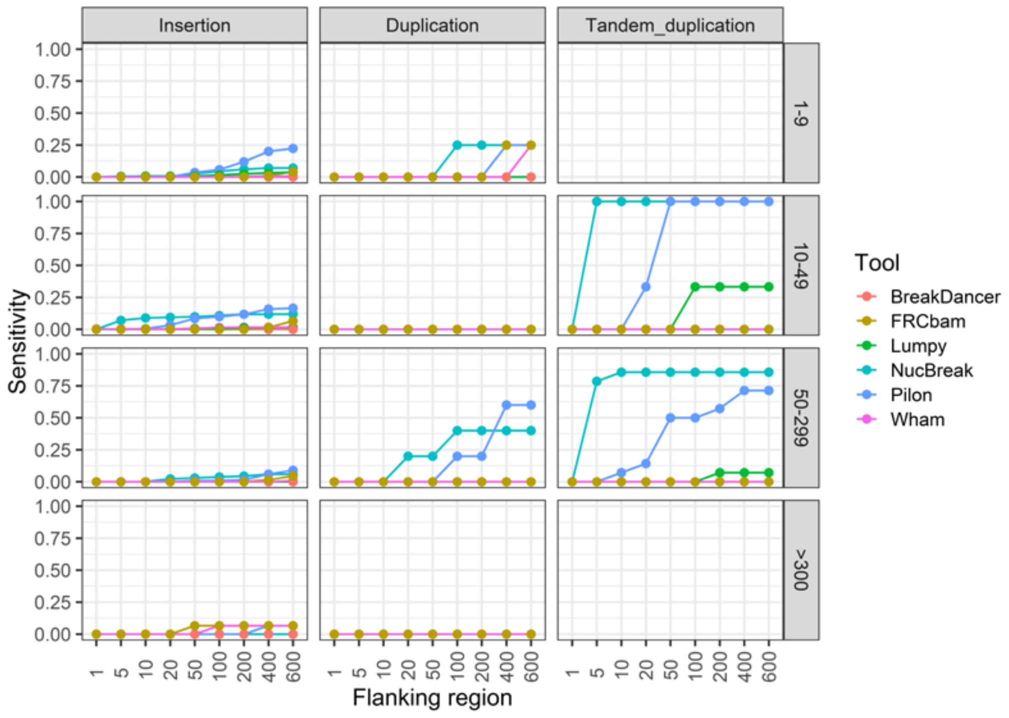


Figure S16 Sensitivity results for the insertion, duplication and tandem duplication groups obtained using the bacterial genome datasets.

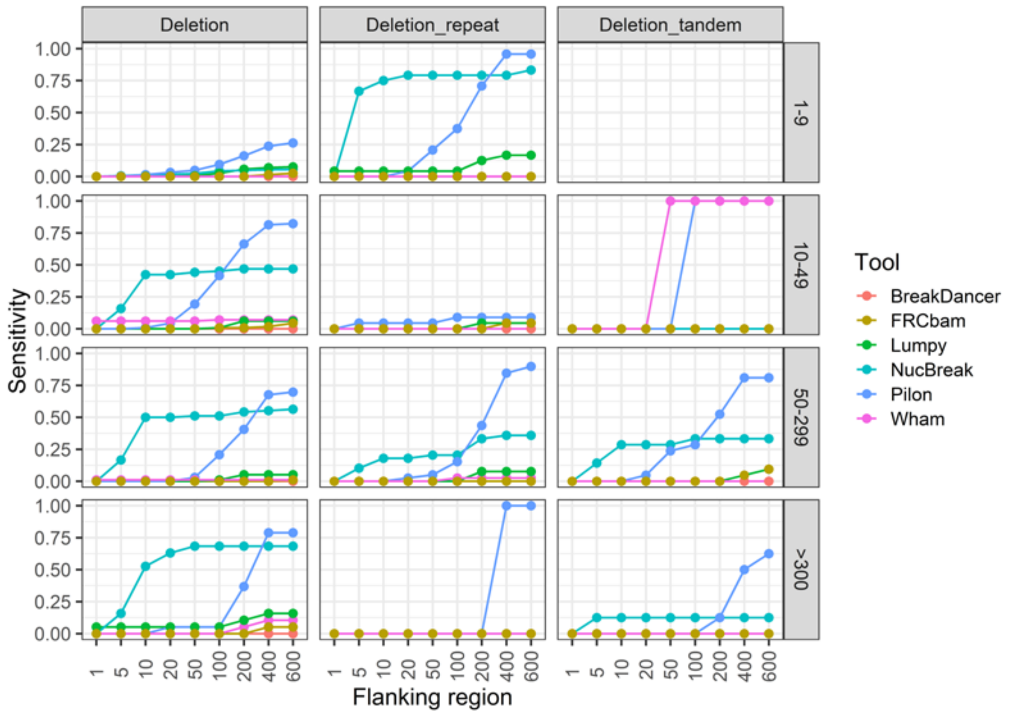


Figure S17 Sensitivity results for the deletion, deletion_repeat and deletion_tandem groups, obtained using the bacterial genome datasets. The deletion_repeat group contains deletions of interspersed repeats or their parts. The deletion_tandem group contains deletions of tandem repeats or their parts.

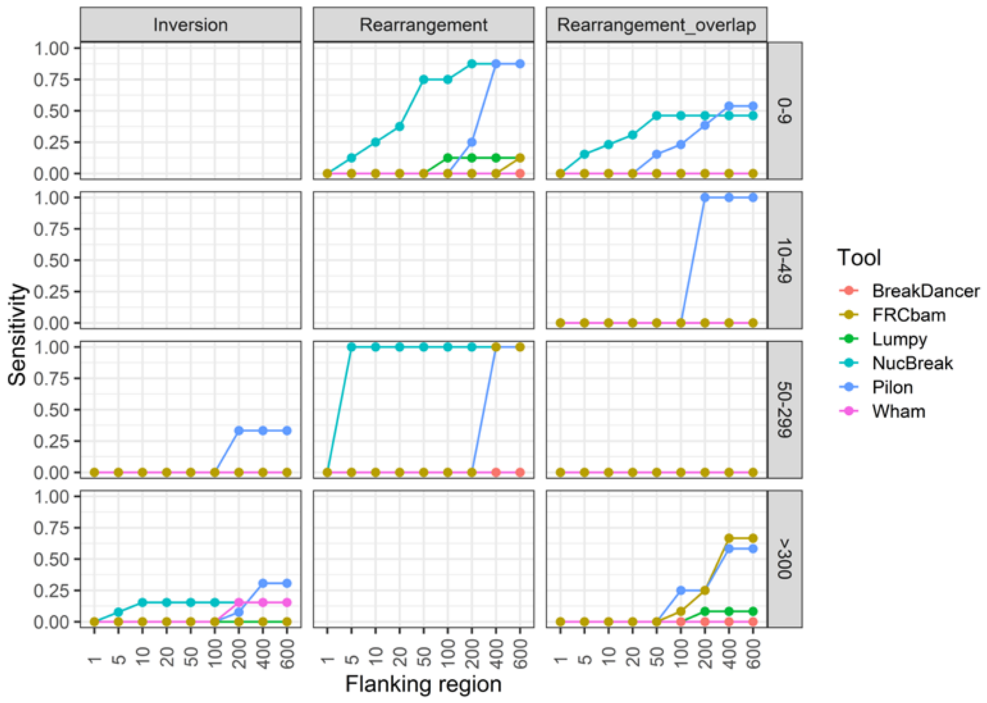


Figure S18 Sensitivity results for the inversion, rearrangement and rearrangement_overlap groups, obtained using the bacterial genome datasets. The rearrangement group consists of relocations and translocations with either inserted regions between misjoined regions (size varied between 1 and 1000) or without them (size is equal to 0). The rearrangement_overlap group consists of relocations and translocations with overlapped misjoined regions.

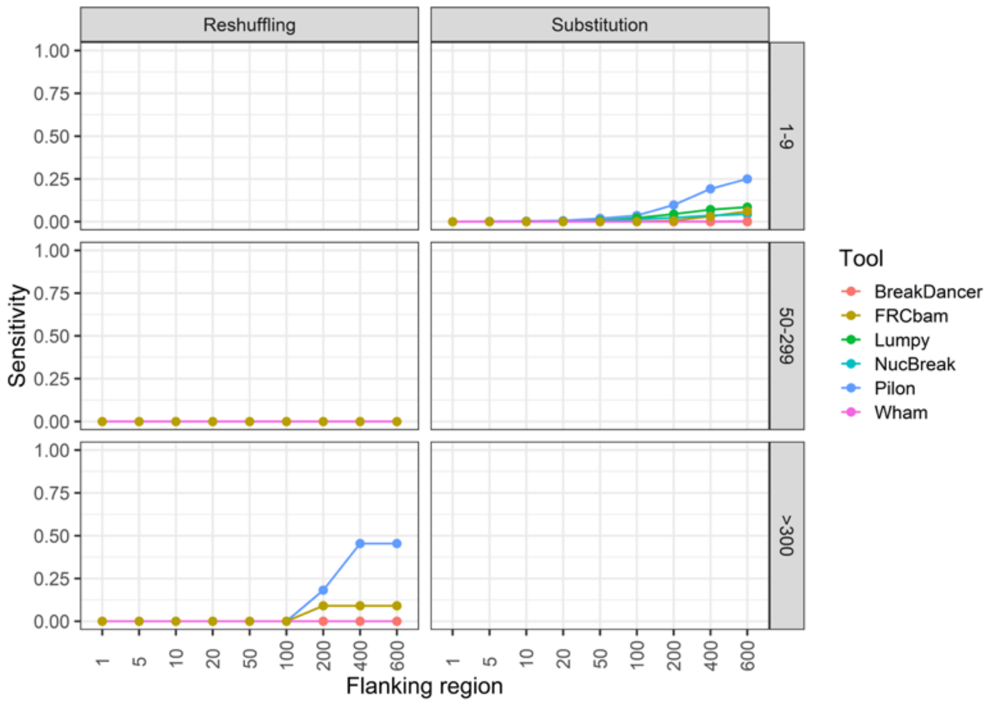


Figure S19 Sensitivity results for the reshuffling and substitution groups, obtained using the bacterial genome datasets.

Supplementary tables

Table S1 Genome modifications implemented during the simulation process. G and A denote a reference genome and assembly, respectively. All other letters denote reference genome and assembly sequence regions. Diff means difference. C’ is the reverse complement of C.

Insertions	
1. G: HB A: HCB Diff: insertion	46. G: DLLLLxC A: DCLLLLxC Diff: duplication
2. G: HBxC A: HCBxC Diff: duplication	47. G: DLLLLxTKxTKxTK A: DTLLLLxTKxTKxTK Diff: duplication
3. G: HBxTKxTKxTK A: HTBxTKxTKxTK Diff: duplication	48. G: DLLLLxTKxTKxTK A: DKLLLLxTKxTKxTK Diff: duplication
4. G: HBxCxCxC A: HCBxCxCxC Diff: duplication	49. G: DLLLLxCxCxC A: DCLLLLxCxCxC Diff: duplication
5. G: HBxTKTKTKTK A: HTBxTKTKTKTK Diff: duplication	50. G: DKTKTKTKT A: DTKTKTKTKT Diff: duplication
6. G: HBxCCCC A: HCBxCCCC Diff: duplication	51. G: DKTKTKTKT A: DKKTKTKTKT Diff: tandem_duplication
7. G: HBxTKTKTKTK A: HKTBxTKTKTKTK Diff: duplication	52. G: DCCCC A: DCCCCC Diff: tandem_duplication
8. G: HBxCCCC A: HCCCCBxCCCC Diff: duplication	53. G: DKTKTKTKT A: DTKKTKTKTKT Diff: duplication
9. G: RxRxR A: RxCRxR Diff: insertion	54. G: DPPPPxTKTKTKTK A: DTPPPPxTKTKTKTK Diff: duplication
10. G: RxRxRxC A: RxCRxRxC Diff: duplication	55. G: DPPPPxTKTKTKTK A: DKPPPPxTKTKTKTK Diff: duplication
11. G: TKxTKxTK A: TKxTTKxTK Diff: tandem_duplication	56. G: DPPPPxCCCC A: DCPPPPxCCCC Diff: duplication
12. G: TKxTKxTK A: TKxKTKxTK Diff: duplication	57. G: DPPPPxTKTKTKTK A: DKTPPPPxTKTKTKTK Diff: duplication

13. G: CxCxC
A: CxCCxC
Diff: tandem_duplication

14. G: RxRxRxTKTKTKTK
A: RxTRxRxTKTKTKTK
Diff: duplication

15. G: RxRxRxTKTKTKTK
A: RxKRxRxTKTKTKTK
Diff: duplication

16. G: RxRxRxTKTKTKTK
A: RxTKRxRxTKTKTKTK
Diff: duplication

17. G: RxRxRxTKTKTKTK
A: RxKTRxRxTKTKTKTK
Diff: duplication

18. G: RxRxRxCCCC
A: RxCCCCRxRxCCCC
Diff: duplication

19. G: RxRxRxTKxTKxTK
A: RxTRxRxTKxTKxTK
Diff: duplication

20. G: RxRxRxTKxTKxTK
A: RxKRxRxTKxTKxTK
Diff: duplication

21. G: RxRxRxCxCxC
A: RxCRxRxCxCxC
Diff: duplication

22. G: RxRxR
A: RxRCxR
Diff: insertion

23. G: RxRxRxC
A: RxRCxRxC
Diff: duplication

24. G: TKxTKxTK
A: TKxTKTxTK
Diff: duplication

25. G: TKxTKxTK
A: TKxTKKxTK
Diff: tandem_duplication

26. G: RxRxRxTKTKTKTK
A: RxRTxRxTKTKTKTK
Diff: duplication

27. G: RxRxRxTKTKTKTK
A: RxRKxRxTKTKTKTK
Diff: duplication

58. G: DPPPPxCCCC
A: DCCCCPPPPxCCCC
Diff: duplication

59. G: LLLLD
A: LLLLCD
Diff: insertion

60. G: LLLLDxC
A: LLLLCDxC
Diff: duplication

61. G: LLLLDxTKxTKxTK
A: LLLLTDxTKxTKxTK
Diff: duplication

62. G: LLLLDxTKxTKxTK
A: LLLLKDxTKxTKxTK
Diff: duplication

63. G: LLLLDxCxCxC
A: LLLLCDxCxCxC
Diff: duplication

64. G: TKTKTKTKD
A: TKTKTKTKTD
Diff: duplication

65. G: TKTKTKTKD
A: TKTKTKTKKD
Diff: tandem_duplication

66. G: TKTKTKTKD
A: TKTKTKTKKTD
Diff: duplication

67. G: PPPPDxTKTKTKTK
A: PPPPTDxTKTKTKTK
Diff: duplication

68. G: PPPPDxTKTKTKTK
A: PPPPKDxTKTKTKTK
Diff: duplication

69. G: PPPPDxCCCC
A: PPPPCDxCCCC
Diff: duplication

70. G: PPPPDxTKTKTKTK
A: PPPPKTDxTKTKTKTK
Diff: duplication

71. G: PPPPDxCCCC
A: PPPPCCCCDxCCCC
Diff: duplication

72. G: PPPP
A: PPCPP
Diff: insertion

28. G: RxRxRxTKTKTKTK
    A: RxRTKxRxTKTKTKTK
    Diff: duplication

29. G: RxRxRxTKTKTKTK
    A: RxRDTxRxTKTKTKTK
    Diff: duplication

30. G: RxRxRxCCCC
    A: RxRCCCCxRxCCCC
    Diff: duplication

31. G: RxRxRxTKxTKxTK
    A: RxRTxRxTKxTKxTK
    Diff: duplication

32. G: RxRxRxTKxTKxTK
    A: RxRKxRxTKxTKxTK
    Diff: duplication

33. G: RxRxRxCxCxC
    A: RxRCxRxCxCxC
    Diff: duplication

34. G: RDxRDxRD
    A: RDxRCDxRD
    Diff: insertion

35. G: RDxRDxRDxC
    A: RDxRCDxRDxC
    Diff: duplication

36. G: TKxTKxTK
    A: TKxTKKxTK
    Diff: tandem_duplication

37. G: RDxRDxRDxTKTKTKTK
    A: RDxRTDxRDxTKTKTKTK
    Diff: duplication

38. G: RDxRDxRDxTKTKTKTK
    A: RDxRKDxRDxTKTKTKTK
    Diff: duplication

39. G: RDxRDxRDxCCCC
    A: RDxRCDxRDxCCCC
    Diff: duplication

40. G: RDxRDxRDxTKTKTKTK
    A: RDxRKTDxRDxTKTKTKTK
    Diff: duplication

41. G: RDxRDxRDxCCCC
    A: RDxRCCCCDxRDxCCCC
    Diff: duplication

42. G: RDxRDxRDxTKxTKxTK
    A: RDxRTDxRDxTKxTKxTK
    Diff: duplication

73. G: PPPPxC
    A: PPCPPxC
    Diff: duplication

74. G: PPPPxTKxTKxTK
    A: PPTPPxTKxTKxTK
    Diff: duplication

75. G: PPPPxTKxTKxTK
    A: PPKPPxTKxTKxTK
    Diff: duplication

76. G: PPPPxCxCxC
    A: PPCPPxCxCxC
    Diff: duplication

77. G: TKTKTKTK
    A: TKTKKTKTK
    Diff: tandem_duplication

78. G: PPPPxTKTKTK
    A: PPTPPxTKTKTK
    Diff: duplication

79. G: PPPPxTKTKTK
    A: PPKPPxTKTKTK
    Diff: duplication

80. G: PPPPxCCC
    A: PPCPPxCCC
    Diff: duplication

81. G: PPPPxTKTKTK
    A: PPKTPPxTKTKTK
    Diff: duplication

82. G: PPPPxCCC
    A: PPCCCPPxCCC
    Diff: duplication

83. G: DKL
    A: DKTKL
    Diff: insertion

84. G: DKLxT
    A: DKTKLxT
    Diff: insertion

85. G: LxKLxLxT
    A: LxKTKLxLxT
    Diff: insertion

86. G: KLxKLxKLxT
    A: KLxKTKLxKLxT
    Diff: insertion

87. G: LKxLKxLKxT
    A: LKxLKTKxLKxT
    Diff: insertion

43. G: RDxRDxRDxTKxTKxTK A: RDxRKDxRDxTKxTKxTK Diff: duplication	88. G: DCR A: DCCR Diff: tandem_duplication
44. G: RDxRDxRDxCxCxC A: RDxRCDxRDxCxCxC Diff: duplication	89. G: DCR A: DCCCCR Diff: tandem_duplication
45. G: DLLLL A: DCLLLL Diff: insertion	90. G: LxCLxL A: LxCCLxL Diff: tandem_duplication
Deletions	
1. G: RCD A: RD Diff: deletion	14. G: DTKKKK A: DKKKK Diff: deletion
2. G: RxCRxR A: RxRxR Diff: deletion	15. G: DTKKKK A: DKKK Diff: deletion
3. G: KRxTKRxKR A: KRxRxKR Diff: deletion_repeat	16. G: DTKKKK A: DK Diff: deletion_tandem
4. G: RxCRxR A: RxxR Diff: deletion_repeat	17. G: DTKKKKF A: DF Diff: deletion
5. G: KxTKFxK A: KxxK Diff: deletion_repeat	18. G: DTLLLLK A: D Diff: deletion
6. G: CRxCRxCR A: CRxRxCR Diff: deletion_repeat	19. G: RTKLKLKLKL A: RLKLKLKL Diff: deletion
7. G: CxCxC A: CxxC Diff: deletion_repeat	20. G: DTKTKTKTK A: DKTKTKTK Diff: deletion
8. G: KxKTxK A: KxxK Diff: deletion_repeat	21. G: DTKTKTKTK A: DTKTTKTK Diff: deletion
9. G: KTxKTxKT A: KTxKxKT Diff: deletion_repeat	22. G: TKTKTKTKD A: TKTKTKTD Diff: deletion
10. G: RTxRTKxRT A: RTxRxRT Diff: deletion_repeat	23. G: DCCCC A: DCCC Diff: deletion_tandem
11. G: RxRCxR A: RxRxR Diff: deletion	24. G: RCCCCD A: RD Diff: deletion
12. G: KTKxK A: KxK Diff: deletion_repeat	25. G: DTTTTK A: DTTT Diff: deletion

13. G: KTK A: K Diff: deletion_repeat	26. G: DTTTTK A: D Diff: deletion
Relocations	**Inversions**
1. G: SzV A: SCV Diff: relocation 2. G: SzV A: SV Diff: relocation 3. G: SCzCV A: SCV Diff: relocation_overlap	1. G: DCR A: DC’R Diff: inversion 2. G: DCRxRxR A: DC’RxRxR Diff: inversion 3. G: DRCxRxR A: DRC’xRxR Diff: inversion 4. G: RCDxRCDxRCD A: RCDxRC’DxRCD Diff: inversion

In the simulated modifications, the following lengths of regions were used:

1. Distance between each manipulation case =2500 bp
2. len(H)=800 bp
3. len(B)=800 bp
4. len(x)=800 bp
5. len(C)={17,30,100,250,800} bp
6. len(TK)={[50,70],[100,150],[250,250],[600,600]} bp, where first number in a pair is len(T) and second number in a pair is len(K)
7. len(R)=600 bp
8. len(D)=600 bp
9. len(L)=200 bp
10. len(P)=400 bp
11. len(F)=len(T) in len(TK)
12. len(S)=1500 bp
13. len(V)=1500 bp
14. len(Z)=15000 bp

Table S2 List of bacterial genomes.

Genome	Genome length, Mb	Accession number	Reads length, bp (first, second)	Coverage	Read library accession number
Bordetella pertussis str. J081	4,11	GCA_002859625.1	250 250	32x	SRR5829829
Brucella melitensis str. 1	3,30	GCA_900236405.1	243 ± 28.8 243 ± 28.7	40x	ERR2192800
Enterobacter cloacae str. AR_0136	5,04	GCA_002204775.1	233 ± 34.9 233 ± 34.8	23x	SRR4025988
Escherichia coli str. 2014C-3599	5,48	GCA_003018935.1	236 ± 39.0 236 ± 38.8	60x	SRR1609862
Klebsiella pneumonia str. SGH10	5,72	GCA_002813595.1	146 ± 15.8 146 ± 15.7	32x	SRR5082357
Pseudomonas aeruginosa str. AR_0095	6.82	GCA_002997005.1	229 ± 38.2 229 ± 36.9	60x	SRR3242025
Salmonella enterica str. CFSAN047866	4,81	GCA_003073535.1	244 ± 27.3 244 ± 27.3	37x	SRR3272258
Staphylococcus aureus str. CFSAN007896	2,86	GCA_003031425.1	236 ± 41.8 236 ± 41.7	28x	SRR5912676

Table S3 Number of ground truth errors in each group.

Error type	Error size	Simulated datasets	Assemblathon 1 dataset	Bacterial genome datasets
insertion	0-9	0	6658	402
	10-49	140	892	414
	50-299	240	38	133
	>300	170	31	15
duplication	0-9	0	23	4
	10-49	380	1	12
	50-299	1510	11	5
	>300	840	287	1

tandem_duplication	0-9	0	0	0
	10-49	60	23	3
	50-299	260	93	14
	>300	170	683	0
deletion	0-9	0	6933	437
	10-49	60	1091	113
	50-299	270	307	96
	>300	270	527	19
deletion_repeat	0-9	0	424	24
	10-49	40	10	22
	50-299	160	21	39
	>300	230	7	2
deletion_tandem	0-9	0	1	0
	10-49	20	37	1
	50-299	30	147	21
	>300	40	24	8
inversion	0-9	0	2	0
	10-49	80	0	0
	50-299	80	6	3
	>300	40	94	13
relocation/ rearrangement	0-9	50	749	8
	10-49	20	2	0
	50-299	20	17	1
	>300	10	95	0
relocation_overlap/ rearrangement_overlap	0-9	0	744	13
	10-49	20	25	1
	50-299	20	152	1
	>300	10	76	12
reshuffling	0-9	0	4	0
	10-49	0	1	0
	50-299	0	11	1

	>300	0	94	11
substitution	0-9	0	225721	8000
	10-49	0	1	0
	50-299	0	3	0
	>300	0	21	0