Tutorials of SynGenoR v1.0

Introduction

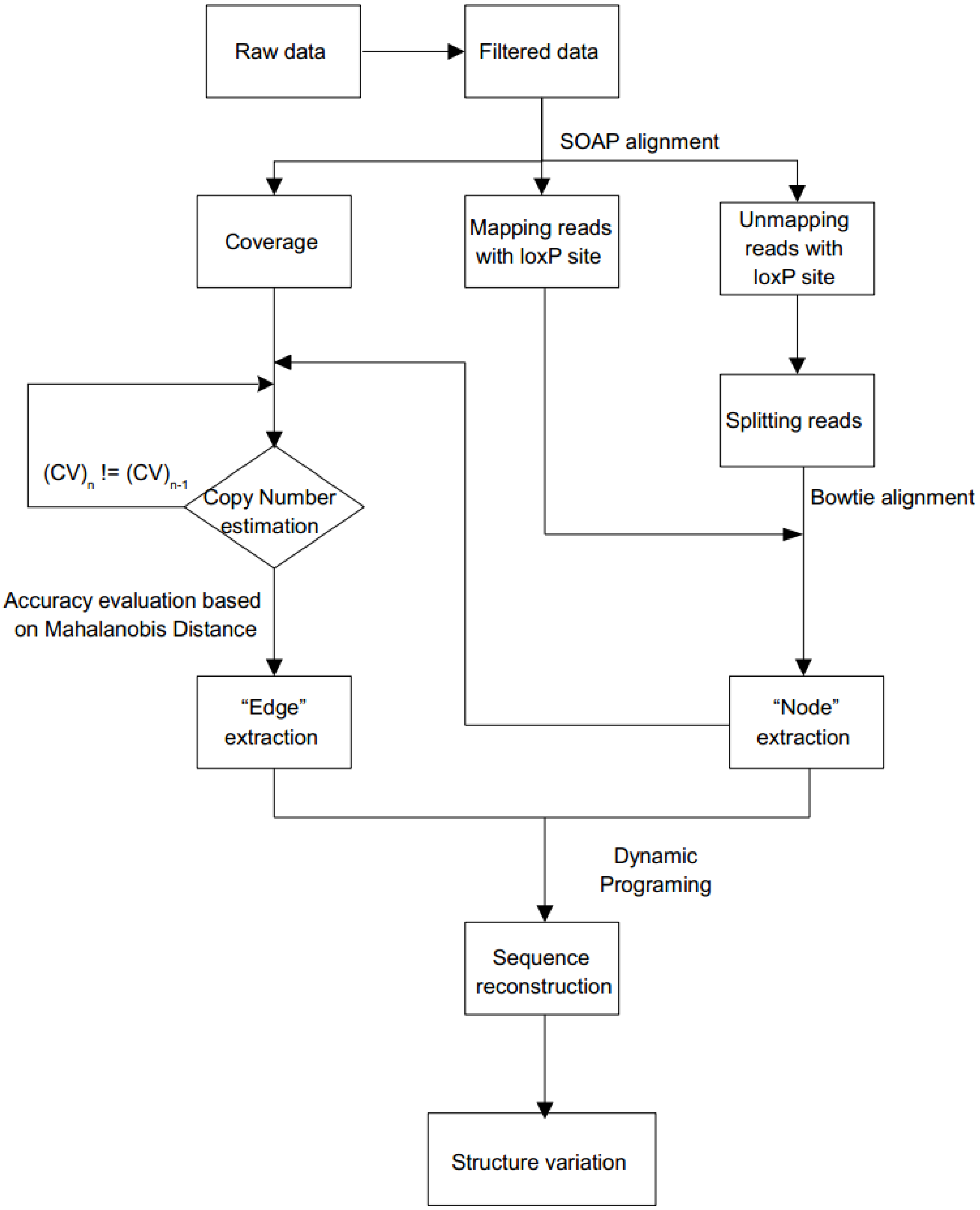
SynGenoR, which stands for Synthetic Genome Reconstruction software, is developed for the synthetic yeast genome project (Sc2.0 project). It is used to reconstruct the yeast genome and to analyse structural variations after induction of SCRaMbLE. The software reconstructs the synthetic chromosome based on loxP site, which mediated the chromosome rearrangement, and the positive correlation between sequencing depth and copy number. The software reconstructs the chromosome by using split reads mapping, iterative copy number estimates, Mahalanobis distance discriminant, and dynamic programming algorithm. By comparing the sequences before and after SCRaMbLE, user could precisely define the type, size, region of structural variations.

The software will be operated in this order:

Input	● Reference, SOAP coverage result, raw sequencing data (.fastq)， loxP site information, ● Sample name, working directory.
Processing	● Extracts loxP site data from the SOAP coverage result, get *.loxp.fq and *loxp.sp files. ● Picks out the unmapping data that related to loxP site from *.loxp.fq and *loxp.sp, get *.umap.fq file. ● Extracts split reads data from *.umap.fq file, get *.splitted.fq file. ● Aligns the *.splitted.fq to reference using bowtie, get *.splitted.map file that contain split reads mapping data. ● Extracts split reads with exceptional supporting reads data from *.splitted.map file to *.gvr file. ● Sort *.gvr file according to the coordinates of chromosome, get *.gvr.sort file. ● Obtain Edge (define as the sequence between two break point) and Node (define as the connecting relation of two Edge) data (*.edg and *.nod, respecitvely) from *.coverage.depthsingle data, *.gvr.sort data, *soap/*soap2 data, and ref.loxpreg data

	using iterative algorithm and Mahalanobis distance discriminant. ● Solves the reconstruction pattern using dynamic programming algorithm, based on *.edg data and *.nod data. ● Verifies the result by using SOAP to realign the raw data to the reconstructed genome. Creates a SVG image for visual verification.
Output	● Reconstruction result of target sequence, SVG image for visual verification

The flowchart of the software is shown below:



The list of programs or scripts included in the software package are shown below:

No.	Program ID	Mnemonic name
1	run_syngenor.pl	run_syngenor
2	filter_gz.pl	filter_gz
3	statreads.pl	statreads
4	pickunmap.pl	pickunmap
5	splitreads.pl	splitreads
6	bowtie2	bowtie
7	getvariareads.pl	getvariareads
8	sortool.pl	sortool
9	extnodedge.pl	exNaE
10	dynpath2recon.pl	dynpath
11	restructseq.pl	restructseq
12	drawSVGstdchc.pl	stdchc
13	drawSVGlxpchc.pl	lxpchc
14	getpereads.pl	getper
15	classifyspr.pl	classifyspr
16	lxpspstat.pl	lxpspstat
17	unmapcheck.pl	unmapcheck
18	fdloxpftr.pl	fdloxpftr

System requirement

32 or 64-bit CPU;

>2G memory;

No requirement on hard drive;

Linux OS (kernel version > 2.6);

GNU Complier Collection (version > 3.4);

Perl (version >5.8);

Installation

Prerequisite software:

SOAP2 (http://soap.genomics.org.cn/soapaligner.html);

BWA (http://bio-bwa.sourceforge.net/);

Bowtie2 (http://bowtie-bio.sourceforge.net).

soap.coverage （https://github.com/sunhappy2019/soap.coverage）

1. Unpack the software to the folder of your choice, by entering the command below in the command line of UNIX/Linux:

   tar -zxvf SynGenoR.v1.0.tar.gz

   then ENTER;

   then enter: cd SynGenoR.v1.0

   At this point, you could see all the perl scripts and executable programs that will be used the analysis.

2. Configures the parameters of the program according to the config.cfg file. It be noted that the programs involved and all input files require a right absolute path before running.

3. Run the software by entering the command below in the command line:

   ```
   perl run_syngenor.pl <config_file> [-option]
   ```

   Read the **Operation instruction** part to look up the option of the software.

   After running the run_syngenor.pl, an executable script, run_syngenor.sh, will be created under the target directory. Some other tools, which will be used by an integrated script that created by the perl script, will be also generated.

Operation instruction

The operation of the software includes following steps:

1. Cleaning reads by quality control;
2. SOAP alignment;
3. Data pre-treatment (SOAP coverage, split reads mapping by Bowtie2);
4. Estimates Node types and Edge copy number using iterative algorithm and Mahalanobis distance discriminant;
5. Solves the reconstruction pattern using dynamic programming algorithm;
6. Visual verification for the reconstruction.

# Each step could be run independently if the input requirement of that step is met.

Operational information:

1 Purpose: reconstructs SCRaMbLEd synthetic chromosome.

2 Operation requirement: the input command should conform the Unix/Linux grammar and limitation.

3 Command line is the only way to initiate the software.

4 Estimated running time: <20min

5 Command:

```
 perl run_syngenor.pl <config_file> [option]

# <config_file> configuration file, format example please see: $Bin/../config.cfg

 Option:

 -show_sfware       show the software use at the process to screen.

 -step [str]        run_syngenor step as follow, default=123456.

                          1. run read quality control

                          2.run mapping by SOAP

                          3.run preprocessing of mapping result files, SOAP Coverage and split reads
                            mapping by Bowtie

                          4. run Extracting Node & Edge information

                          5. run Reconstructing sequence

                          6. run validation

 -run                the step for running shell, default equal -step set.

 -shell [str]        output main shell name, default=run_syngenor.sh.

 -help               output help information to screen.
```

Remark:

1. You can use -run no, then just write the shell but not run it.

2. At config_file options end with "\n", see the form at $Bin/config.cfg.

3. You may not be able to get a reconstructed sequence due to CN evaluation error. You can CHECK the *.edg and change the copy number, then run step 45 again.

Example:

1. run all the progress.

```
perl run_syngenor.pl config.cfg &
```

2. just write the **shell and** then run run_syngenor.sh

```
perl run_syngenor.pl config.cfg –run no

nohup run_syngenor.sh &
```

NOTE: the input sequencing data should have an original Illumina read id like (the end of /1 or /2 is requisite):

```
@FCC0CH9ACXX:7:1101:15678:3662#ACCTTGCA/1

  Flow_cell_ID  :  postion      # 8bp_index /1
```

Input-output file

Input file: $Bin/../config.cfg

Output file:

```
|--work_dir
   |--sample_ID                                 #sample directory
      |-- 01.cleandata                          #step 01. Quality control
         |-- sample_ID_1.clean.dedup.fq         #clean read 1
         |-- sample_ID_2.clean.dedup.fq         #clean read 2
      |-- 02.soap                               #step 02. SOAP alignment
         |-- sample_ID.soap                     #pair-end (PE) result
         |-- sample_ID.soap2                    #single-end (SE) result
      |-- 03.prep                               #step 03. Data pre-treatment
         |--01.coverage                         #s1 do Soap coverage single base sequencing depth statistics
```

```
          |-- sample_ID.coverage.depthsingle
        |--02.readstat                 #s2 do reads statistics and extracts loxP reads related reads
          |-- sample_ID.loxp.fq        #complete loxP sequence-contained reads in the clean reads data
          |-- sample_ID.loxp.sp        # SOAP mapping result related to loxP reads
          |-- sample_ID.stat           #reads stats
        |-- 03.bowtie                  # s3 split reads Bowtie2 alignment
          |-- sample_ID.umap.fq        # loxP reads that were not mapped by SOAP
          |-- sample_ID.splitted.fq    #splitted split reads result
          |-- sample_ID.splitted.map   #Bowtie mapping result
        |-- 04.sortgvr                 #s4 gets and sorts pair-end bowtie mapping result
            |-- sample_ID.sort.gvr
    |--04.extNaE                       # step 04.obtain Node classes and Edge copy number
        |-- sample_ID.edg              #Edge information
        |-- sample_ID.nod              #Node information
        |--sample_ID.log               #log file
    |--05.dynpath                  # step 05.Generates reconstruction path by dynamic programming algorithm
        |--sample_ID.encod             #reconstruction path information
        |--sample_ID.info              #record for all reconstruction pattern that fit the setting
        |--sample_ID.log               #record for all reconstruction pattern information
    |--06.validation                   # step 06.verification of reconstruction sequence
        |--01.valsoap                  #s1 validation SOAP alignment
            |--sample_ID.reseq          #reconstructed chromosome DNA sequence
            |--sample_ID.valsp            #reconstructed genome SOAP alignment (PE) result
            |--sample_ID.valsp2          #reconstructed genome SOAP alignment (SE) result
        |--02.stdchc                      #s2 reads mapping check
            |-- sample_ID.stdchc.svg    # reads mapping SVG
            |-- sample_ID.depth            #sequecing depth information
            |-- sample_ID.lxps              # loxP site reads stats
            |-- sample_ID.sort.spr          #sorted SOAP alignment (SE) result
            |-- sample_ID.sort.val2.spr    #sorted validation SOAP alignment (SE) result
            |-- sample_ID.sort.val.spr      #sorted validation SOAP alignment (PE) result
            |-- sample_ID.varid              #loxp reads classification information
        |--03.umapBWA                     #s3 unmapping loxP reads BWA alignment results
            |-- sample_ID.val.umap.sort.fai
        |--04. lxpchc                      #s4 loxp reads mapping check
            |-- sample_ID.lxpchc.svg      # loxp reads mapping SVG
            |-- sample_ID.loxp.sort.spr    #sorted SOAP alignment (SE) result of loxP reads
            |-- sample_ID.lxps               #loxP site reads stats
            |-- sample_ID. val loxp.sort.spr # validation SOAP alignment (SE) result of loxP reads
            |-- sample_ID.val.lxps          # statistics of loxP site reads in reconstructed genome
            |-- sample_ID.lxpftr.lst          # annotation of loxP in reconstructed genome
            |-- sample_ID.val.umap.fq      # unmapping loxP reads in validation SOAP alignment
            |-- sample_ID.splitted.umc      # dubious loxP reads checked by Bowtie2
        |-- sample_ID.umc                # unmapping loxP reads checked by bwa
```

Operation process and output

Here we use an example (sample JS606) to illustrate the operation process and output:

Step 01. SOAP alignment

(1) soap –a JS606_1.dup.clean –b JS606_2.dup.clean -m 375 -x 625 -v 4 -p 6 -o JS606.soap -2 JS606.soap2

#SOAP usage and output format please visit: SOAP2 (http://soap.genomics.org.cn/soapaligner.html)

Step 02. Pre-treatment

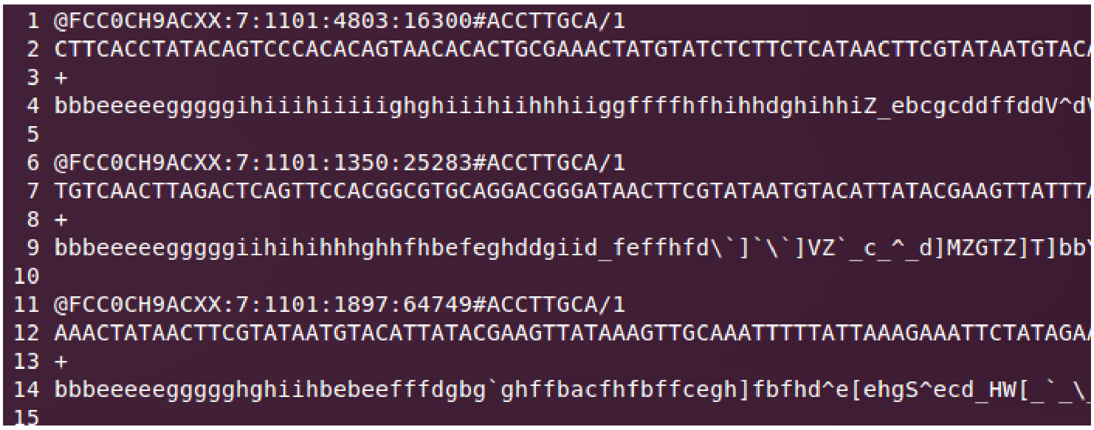
(1) Soap coverage

./soap.coverage2.27 -cvg -p 4 –refsingle BY4741chr9RD_SynIXR.fa –i JS606.soap -2 JS606.soap2 –depthsingle JS606.coverage.depthsingle –o soappaired.out

(2) Do reads statistics and extracts loxP reads related reads

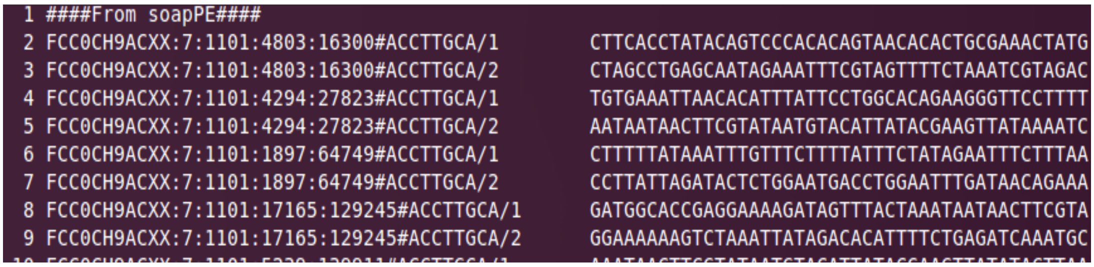
perl statreads.pl JS606_1.dup.clean JS606_2.dup.clean JS606.soap JS606.soap2 [option]

>sample_ID.loxp.fq: complete loxP sequence reads(*.fq) in the clean reads data.



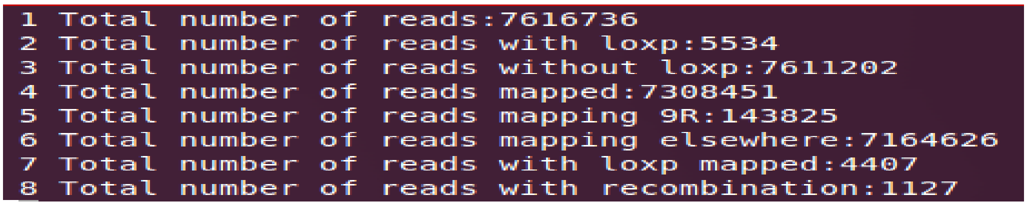
```
@FCC0CH9ACXX:7:1101:4803:16300#ACCTTGCA/1
CTTCACCTATACAGTCCCACACAGTAACACACTGCGAAACTATGTATCTCTTCTCATAACTTCGTATAATGTAC
+
bbbeeeeeggggggihiiihiiiiighghiiihiihhhiiggffffhfhihhdghihhiZ_ebcgcddffddV^d

@FCC0CH9ACXX:7:1101:1350:25283#ACCTTGCA/1
TGTCAACTTAGACTCAGTTCCACGGCGTGCAGGACGGGATAACTTCGTATAATGTACATTATACGAAGTTATTT
+
bbbeeeeeggggggiihihihhhghhfhbefeghddgiid_feffhfd\`]`\`]VZ`_c_^_d]MZGTZ]T]bb

@FCC0CH9ACXX:7:1101:1897:64749#ACCTTGCA/1
AAACTATAACTTCGTATAATGTACATTATACGAAGTTATAAAGTTGCAAATTTTTATTAAAGAAATTCTATAGA
+
bbbeeeeeggggghghiihbebeefffdgbg`ghffbacfhfbffcegh]fbfhd^e[ehgS^ecd_HW[_`_\

```

>sample_ID.loxp.sp: SOAP file (includes Pe and SE) related to loxp



```
####From soapPE####
FCC0CH9ACXX:7:1101:4803:16300#ACCTTGCA/1       CTTCACCTATACAGTCCCACACAGTAACACACTGCGAAACTATG
FCC0CH9ACXX:7:1101:4803:16300#ACCTTGCA/2       CTAGCCTGAGCAATAGAAATTTCGTAGTTTTCTAAATCGTAGAC
FCC0CH9ACXX:7:1101:4294:27823#ACCTTGCA/1       TGTGAAATTAACACATTTATTCCTGGCACAGAAGGGTTCCTTTT
FCC0CH9ACXX:7:1101:4294:27823#ACCTTGCA/2       AATAATAACTTCGTATAATGTACATTATACGAAGTTATAAAATC
FCC0CH9ACXX:7:1101:1897:64749#ACCTTGCA/1       CTTTTTATAAATTTGTTTCTTTTATTTCTATAGAATTTCTTTAA
FCC0CH9ACXX:7:1101:1897:64749#ACCTTGCA/2       CCTTATTAGATACTCTGGAATGACCTGGAATTTGATAACAGAAA
FCC0CH9ACXX:7:1101:17165:129245#ACCTTGCA/1     GATGGCACCGAGGAAAAGATAGTTTACTAAATAATAACTTCGTA
FCC0CH9ACXX:7:1101:17165:129245#ACCTTGCA/2     GGAAAAAAGTCTAAATTATAGACACATTTTCTGAGATCAAATGC
```

>sample_id.stat: loxP sites statistic data

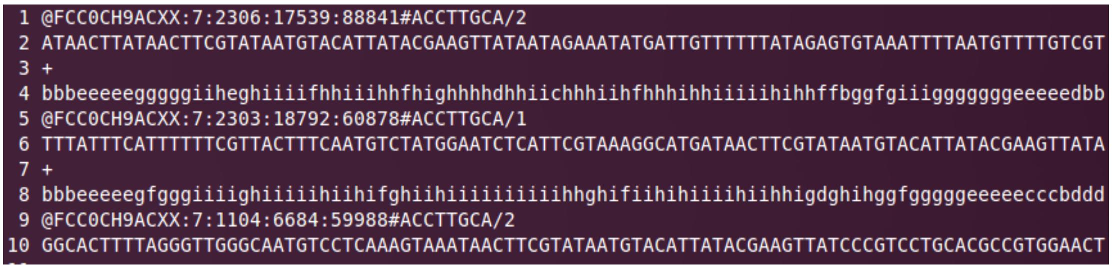


```
Total number of reads:7616736
Total number of reads with loxp:5534
Total number of reads without loxp:7611202
Total number of reads mapped:7308451
Total number of reads mapping 9R:143825
Total number of reads mapping elsewhere:7164626
Total number of reads with loxp mapped:4407
Total number of reads with recombination:1127
```

(3) Aligns split reads using Bowtie2

perl pickunmap.pl JS606.loxp.fq JS606.loxp.sp -spid JS606

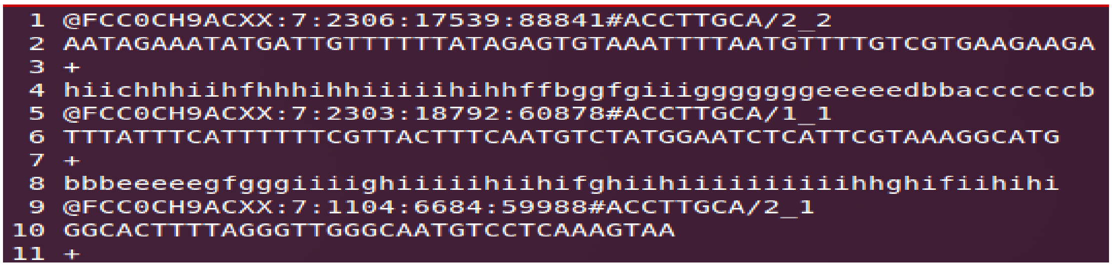
>JS606.umap.fq: unmapping fastq file related to loxP site



```
@FCC0CH9ACXX:7:2306:17539:88841#ACCTTGCA/2
ATAACTTATAACTTCGTATAATGTACATTATACGAAGTTATAATAGAAATATGATTGTTTTTTATAGAGTGTAAATTTTAATGTTTTGTCGT
+
bbbeeeeeggggggiiheghiiiifhhiiihhfhighhhhdhhiichhhiihfhhhihhiiiiihihhffbggfgiiiggggggeeeeedbb
@FCC0CH9ACXX:7:2303:18792:60878#ACCTTGCA/1
TTTATTTCATTTTTTCGTTACTTTCAATGTCTATGGAATCTCATTCGTAAAGGCATGATAACTTCGTATAATGTACATTATACGAAGTTATA
+
bbbeeeeegfgggiiiighiiiiihiihifghiihiiiiiiiiiihhghifiihihiiiihiihhigdghihggfgggggeeeeecccbddd
@FCC0CH9ACXX:7:1104:6684:59988#ACCTTGCA/2
GGCACTTTTAGGGTTGGGCAATGTCCTCAAAGTAAATAACTTCGTATAATGTACATTATACGAAGTTATCCCGTCCTGCACGCCGTGGAACT
```

perl splitreads.pl JS606.umap.fq -spid JS606

> JS606.splitted.fq: split reads fastq file extracted from unmap.fq

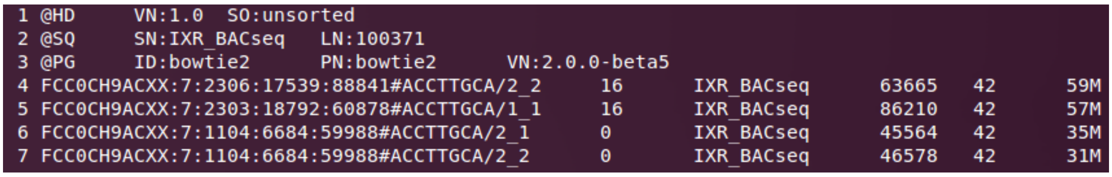


```
@FCC0CH9ACXX:7:2306:17539:88841#ACCTTGCA/2_2
AATAGAAATATGATTGTTTTTTATAGAGTGTAAATTTTAATGTTTTGTCGTGAAGAAGA
+
hiichhhiihfhhhihhiiiiihihhffbggfgiiigggggggeeeeedbbacccccb
@FCC0CH9ACXX:7:2303:18792:60878#ACCTTGCA/1_1
TTTATTTCATTTTTTCGTTACTTTCAATGTCTATGGAATCTCATTCGTAAAGGCATG
+
bbbeeeeegfgggiiiighiiiiihiihifghiihiiiiiiiiiihhghifiihihi
@FCC0CH9ACXX:7:1104:6684:59988#ACCTTGCA/2_1
GGCACTTTTAGGGTTGGGCAATGTCCTCAAAGTAA
+
```

bowtie2 –x index/refseq –q JS606.splitted.fq –S JS606.splitted.map

>JS606.splitted.map: result of aligning split reads fq file to the reference

Output form please refer to Bowtie2: http://bowtie-bio.sourceforge.net

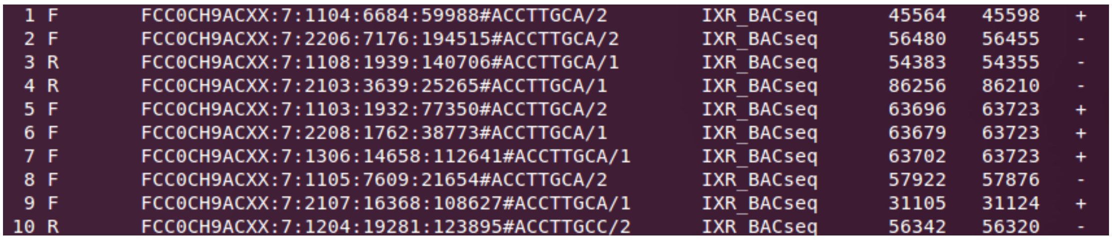


```
@HD     VN:1.0  SO:unsorted
@SQ     SN:IXR_BACseq   LN:100371
@PG     ID:bowtie2      PN:bowtie2      VN:2.0.0-beta5
FCC0CH9ACXX:7:2306:17539:88841#ACCTTGCA/2_2     16      IXR_BACseq      63665   42      59M
FCC0CH9ACXX:7:2303:18792:60878#ACCTTGCA/1_1     16      IXR_BACseq      86210   42      57M
FCC0CH9ACXX:7:1104:6684:59988#ACCTTGCA/2_1      0       IXR_BACseq      45564   42      35M
FCC0CH9ACXX:7:1104:6684:59988#ACCTTGCA/2_2      0       IXR_BACseq      46578   42      31M
```

(4) Obtains and sorts pair-end bowtie mapping result

perl getvariareads.pl JS606.splitted.map –sp JS606

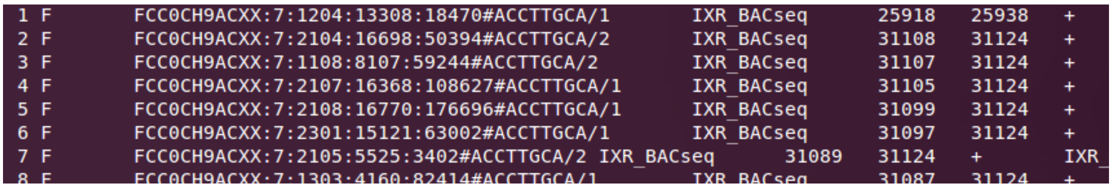
> JS606.gvr: statistics about split reads' position and direction. F indicates the split read is at the same direction as the mapping read, while R indicate the opposite.



```
F       FCC0CH9ACXX:7:1104:6684:59988#ACCTTGCA/2        IXR_BACseq      45564   45598   +
F       FCC0CH9ACXX:7:2206:7176:194515#ACCTTGCA/2       IXR_BACseq      56480   56455   -
R       FCC0CH9ACXX:7:1108:1939:140706#ACCTTGCA/1       IXR_BACseq      54383   54355   -
R       FCC0CH9ACXX:7:2103:3639:25265#ACCTTGCA/1        IXR_BACseq      86256   86210   -
F       FCC0CH9ACXX:7:1103:1932:77350#ACCTTGCA/2        IXR_BACseq      63696   63723   +
F       FCC0CH9ACXX:7:2208:1762:38773#ACCTTGCA/1        IXR_BACseq      63679   63723   +
F       FCC0CH9ACXX:7:1306:14658:112641#ACCTTGCA/1      IXR_BACseq      63702   63723   +
F       FCC0CH9ACXX:7:1105:7609:21654#ACCTTGCA/2        IXR_BACseq      57922   57876   -
F       FCC0CH9ACXX:7:2107:16368:108627#ACCTTGCA/1      IXR_BACseq      31105   31124   +
R       FCC0CH9ACXX:7:1204:19281:123895#ACCTTGCC/2      IXR_BACseq      56342   56320   -
```

perl sortool.pl JS606.gvr.temp -spid JS606 -type gvr

>JS606.sort.gvr sorted result

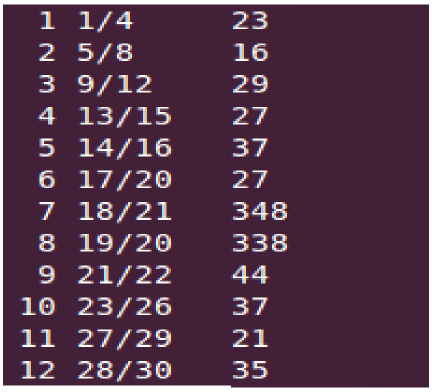


```
F       FCC0CH9ACXX:7:1204:13308:18470#ACCTTGCA/1       IXR_BACseq      25918   25938   +
F       FCC0CH9ACXX:7:2104:16698:50394#ACCTTGCA/2       IXR_BACseq      31108   31124   +
F       FCC0CH9ACXX:7:1108:8107:59244#ACCTTGCA/2        IXR_BACseq      31107   31124   +
F       FCC0CH9ACXX:7:2107:16368:108627#ACCTTGCA/1      IXR_BACseq      31105   31124   +
F       FCC0CH9ACXX:7:2108:16770:176696#ACCTTGCA/1      IXR_BACseq      31099   31124   +
F       FCC0CH9ACXX:7:2301:15121:63002#ACCTTGCA/1       IXR_BACseq      31097   31124   +
F       FCC0CH9ACXX:7:2105:5525:3402#ACCTTGCA/2 IXR_BACseq      31089   31124   +       IXR_
F       FCC0CH9ACXX:7:1303:4160:82414#ACCTTGCA/1        IXR_BACseq      31087   31124   +
```

Step 03. Get Node classes and Edge copy number

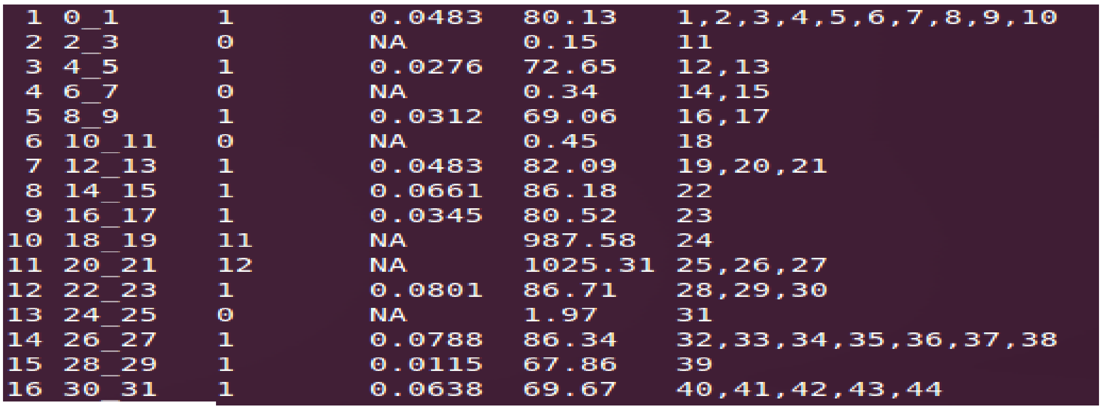
perl extnodedge.pl -prefix JS606 -chrid IXR_BACseq –coverage JS606.coverage.depthsingle –refcover soap.coverage.depthsingle –abnorm JS606.sort.gvr –soap JS606.soap -soap2 JS606.soap2 –loxpregion IXR_BACseq.loxpreg -minread 2 -mdep 10 -mcycle 10 -cutlen 500 > JS606.log

> JS606.nod: node information. The first column indicates the possible connecting relation between two edges. The second column indicates the number of split reads that support the relation.



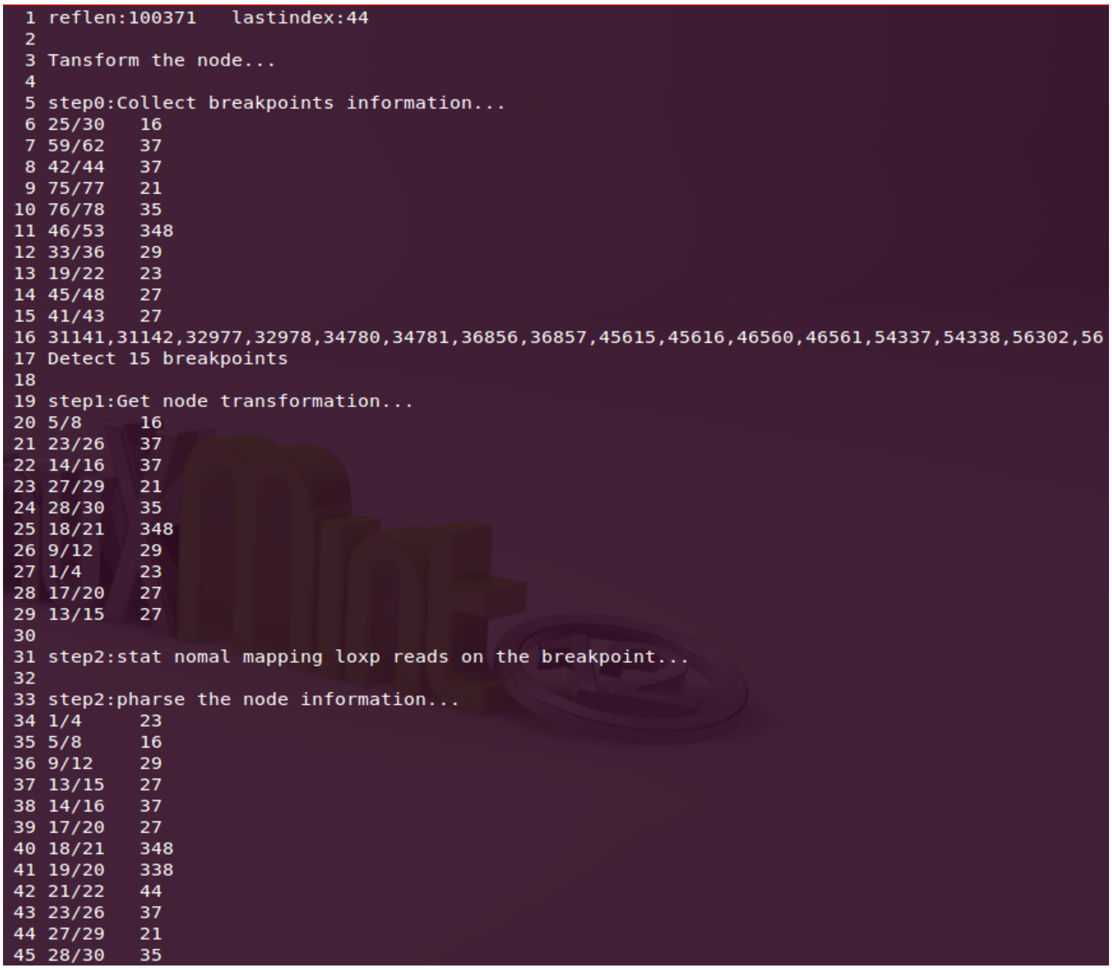
```
 1 1/4     23
 2 5/8     16
 3 9/12    29
 4 13/15   27
 5 14/16   37
 6 17/20   27
 7 18/21   348
 8 19/20   338
 9 21/22   44
10 23/26   37
11 27/29   21
12 28/30   35
```

>JS66.edg: edge information. The first column is the No. assigned to the edge. The second column is the edge copy number estimated by iterative algorithm and Mahalanobis distance discriminant. The third column is the error rate of the estimation. The fourth column is the average sequencing depth of the edge region. The fifth column is the reference region mapped by the edge.



```
 1 0_1     1       0.0483  80.13   1,2,3,4,5,6,7,8,9,10
 2 2_3     0       NA      0.15    11
 3 4_5     1       0.0276  72.65   12,13
 4 6_7     0       NA      0.34    14,15
 5 8_9     1       0.0312  69.06   16,17
 6 10_11   0       NA      0.45    18
 7 12_13   1       0.0483  82.09   19,20,21
 8 14_15   1       0.0661  86.18   22
 9 16_17   1       0.0345  80.52   23
10 18_19   11      NA      987.58  24
11 20_21   12      NA      1025.31 25,26,27
12 22_23   1       0.0801  86.71   28,29,30
13 24_25   0       NA      1.97    31
14 26_27   1       0.0788  86.34   32,33,34,35,36,37,38
15 28_29   1       0.0115  67.86   39
16 30_31   1       0.0638  69.67   40,41,42,43,44
```

>JS606.log: log file records the solving and checking process.

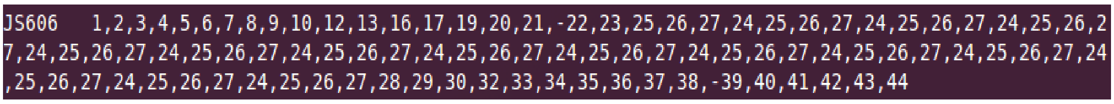


```
reflen:100371   lastindex:44

Tansform the node...

step0:Collect breakpoints information...
25/30   16
59/62   37
42/44   37
75/77   21
76/78   35
46/53   348
33/36   29
19/22   23
45/48   27
41/43   27
31141,31142,32977,32978,34780,34781,36856,36857,45615,45616,46560,46561,54337,54338,56302,56
Detect 15 breakpoints

step1:Get node transformation...
5/8     16
23/26   37
14/16   37
27/29   21
28/30   35
18/21   348
9/12    29
1/4     23
17/20   27
13/15   27

step2:stat nomal mapping loxp reads on the breakpoint...

step2:pharse the node information...
1/4     23
5/8     16
9/12    29
13/15   27
14/16   37
17/20   27
18/21   348
19/20   338
21/22   44
23/26   37
27/29   21
28/30   35
```

Step 04. Generates the reconstruction path by dynamic programming algorithm

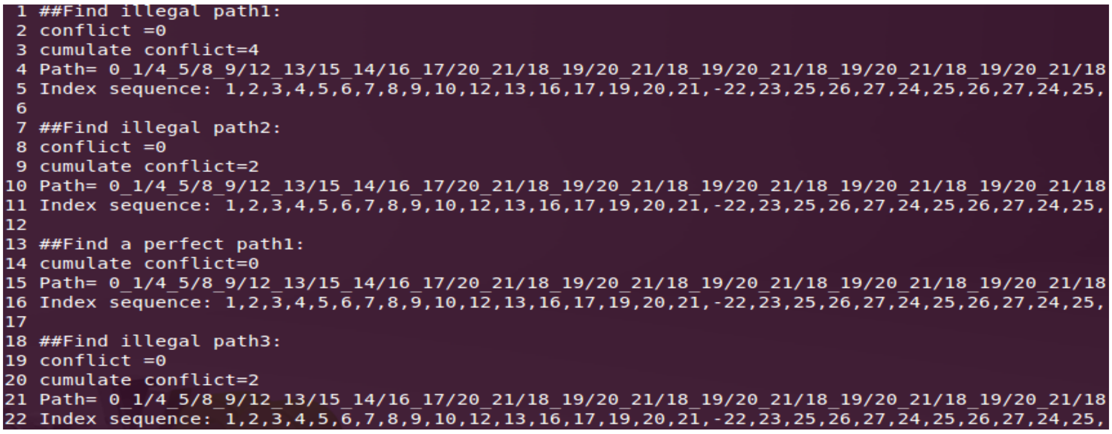
perl dynpath2recon.pl JS606.edg JS606.nod -spid JS606 -mcf 2

> JS606.encod: The most possible fragment order for the rearranged chromosome. A digit represents a chromosome fragment and its original position in the reference. The reference genome is separated into 44 fragments.



```
JS606   1,2,3,4,5,6,7,8,9,10,12,13,16,17,19,20,21,-22,23,25,26,27,24,25,26,27,24,25,26,27,24,25,26,2
7,24,25,26,27,24,25,26,27,24,25,26,27,24,25,26,27,24,25,26,27,24,25,26,27,24,25,26,27,24,25,26,27,24
,25,26,27,24,25,26,27,24,25,26,27,28,29,30,32,33,34,35,36,37,38,-39,40,41,42,43,44
```

> JS606.info: record of all possible reconstruction pattern.



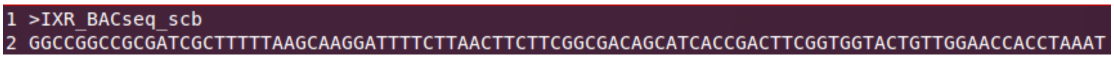
```
##Find illegal path1:
conflict =0
cumulate conflict=4
Path= 0_1/4_5/8_9/12_13/15_14/16_17/20_21/18_19/20_21/18_19/20_21/18_19/20_21/18_19/20_21/18
Index sequence: 1,2,3,4,5,6,7,8,9,10,12,13,16,17,19,20,21,-22,23,25,26,27,24,25,26,27,24,25,

##Find illegal path2:
conflict =0
cumulate conflict=2
Path= 0_1/4_5/8_9/12_13/15_14/16_17/20_21/18_19/20_21/18_19/20_21/18_19/20_21/18_19/20_21/18
Index sequence: 1,2,3,4,5,6,7,8,9,10,12,13,16,17,19,20,21,-22,23,25,26,27,24,25,26,27,24,25,

##Find a perfect path1:
cumulate conflict=0
Path= 0_1/4_5/8_9/12_13/15_14/16_17/20_21/18_19/20_21/18_19/20_21/18_19/20_21/18_19/20_21/18
Index sequence: 1,2,3,4,5,6,7,8,9,10,12,13,16,17,19,20,21,-22,23,25,26,27,24,25,26,27,24,25,

##Find illegal path3:
conflict =0
cumulate conflict=2
Path= 0_1/4_5/8_9/12_13/15_14/16_17/20_21/18_19/20_21/18_19/20_21/18_19/20_21/18_19/20_21/18
Index sequence: 1,2,3,4,5,6,7,8,9,10,12,13,16,17,19,20,21,-22,23,25,26,27,24,25,26,27,24,25,
```

Step05 Verification of the reconstructed sequence

(1) validation SOAP alignment

perl restructseq.pl IXR_BACseq.loxpreg JS606.encod -spid JS606 -reseqid IXR_BACseq_scb

>JS606.reseq： reconstructed JS606 fasta file.



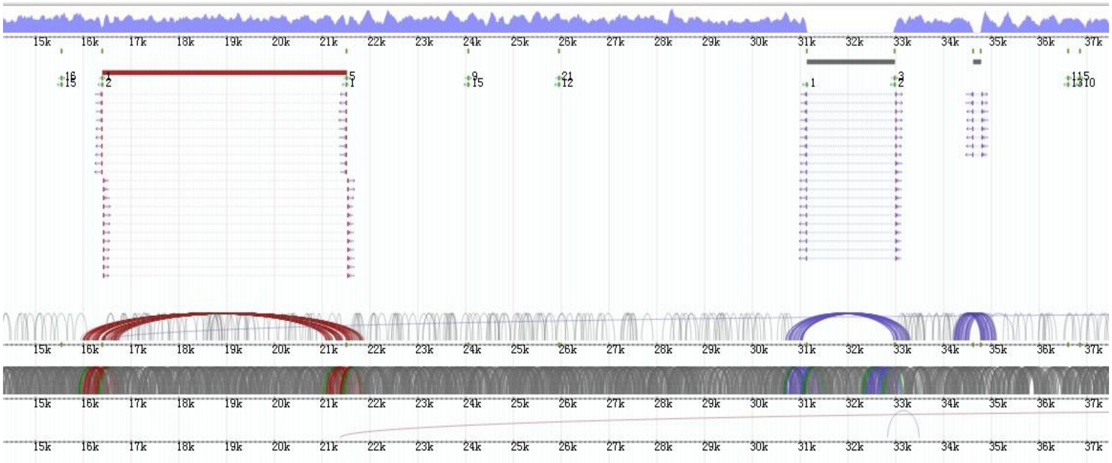
```
>IXR_BACseq_scb
GGCCGGCCGCGATCGCTTTTTAAGCAAGGATTTTCTTAACTTCTTCGGCGACAGCATCACCGACTTCGGTGGTACTGTTGGAACCACCTAAAT
```

./soap –a JS606_1.dup.clean –b JS606_2.dup.clean -m 375 -x 625 -v 4 -p 6 -o JS606.soap -2 JS606.soap2 –D reseq_scb.fa.index

(2) reads mapping check

perl drawSVGstdchc.pl -prefix JS606 -insize 500 -refid IXR_BACseq -reflen 100731 –reseq JS606.reseq –loxpftr loxpftr.lst –cvg JS606.depth –lxps JS606.lxps –gvr JS606.sort.gvr –spr JS606.sort.spr –varid JS606.varid -val2 JS606.sort.val2.spr –val JS606.sort.val.spr –lsv JS606.lsv

> JS606.stdchc.svg: reads mapping SVG. This file could be open by google chrome, other browser or any other svg software. For file specification, please refer to svg_specification.pdf.



(3) BWA alignment of unmapped loxP reads

```
./bwa aln -o 3 -e 64 -i 2 -L -l 31 -k 4 -t 2 -M 1 -O 8 -E 2 -m 2000000 reseq _scb.fa JS606.val.umap.fq > JS606.val.umap.sai
```

```
./bwa samse reseq _scb.fa JS606.val.umap.sai JS606.val.umap.fq > JS606.val.umap.fai
```

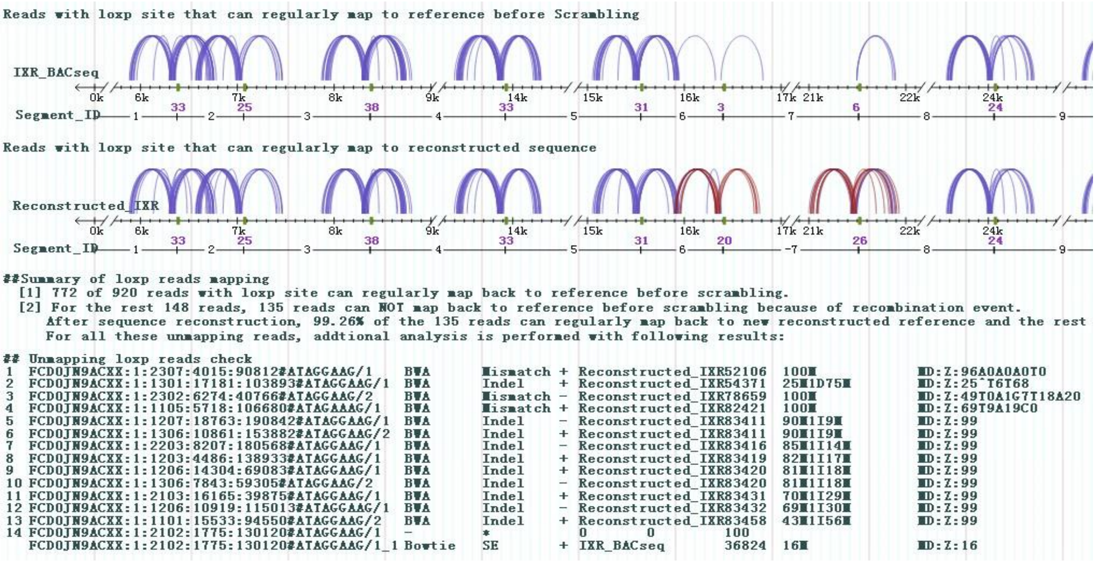
```
sort -k 3,3 -k 4g,4 JS606.val.umap.fai    JS606.val.umap.sort.fai
```

For BWA data format and usage, please refer to BWA : http://bio-bwa.sourceforge.net/

(4) loxP reads mapping check

```
perl drawSVGlxpchc.pl -prefix JS606 -insize 500 -DNAtype Circular -refid IXR_BACseq –refcode refseq.encod –refseq IXR_BACseq.fa -reflen 100731    -loxpftr loxpftr.lst –lxpfq JS606.loxp.fq –lxpspr JS606.loxp.sort.spr –lxpspid JS606.loxp.spid –lxpspstat JS606.lxps –rencode JS606.encod –reseq JS606.reseq –relxpftr JS606.lxpftr.lst –relxpspr JS606.val.loxp.sort.spr –relxpspid JS606.val.loxp.spid –relxpspstat JS606.val.lxps –reumc JS606.umc –splitumc JS606.splitted.umc
```

> JS606.lxpchc.svg: loxP reads mapping SVG. This file could be open by google chrome, other browser or other svg softwares. For file specification, please refer to svg_specification.docx



```
Reads with loxp site that can regularly map to reference before Scrambling

Reads with loxp site that can regularly map to reconstructed sequence

##Summary of loxp reads mapping
  [1] 772 of 920 reads with loxp site can regularly map back to reference before scrambling.
  [2] For the rest 148 reads, 135 reads can NOT map back to reference before scrambling because of recombination event.
      After sequence reconstruction, 99.26% of the 135 reads can regularly map back to new reconstructed reference and the rest
      For all these unmapping reads, addtional analysis is performed with following results:

## Unmapping loxp reads check
1  FCD0JN9ACXX:1:2307:4015:90812#ATAGGAAG/1    BWA     Mismatch + Reconstructed_IXR52106  100M          MD:Z:96A0A0A0T0
2  FCD0JN9ACXX:1:1301:17181:103893#ATAGGAAG/1  BWA     Indel    + Reconstructed_IXR54371  25M1D75M      MD:Z:25^T6T68
3  FCD0JN9ACXX:1:2302:6274:40766#ATAGGAAG/2    BWA     Mismatch - Reconstructed_IXR78659  100M          MD:Z:49T0A1G7T18A20
4  FCD0JN9ACXX:1:1105:5718:106680#ATAGAAAG/1   BWA     Mismatch + Reconstructed_IXR82421  100M          MD:Z:69T9A19C0
5  FCD0JN9ACXX:1:1207:18763:190842#ATAGGAAG/1  BWA     Indel    - Reconstructed_IXR83411  90M1I9M       MD:Z:99
6  FCD0JN9ACXX:1:1306:10861:153882#ATAGGAAG/2  BWA     Indel    + Reconstructed_IXR83411  90M1I9M       MD:Z:99
7  FCD0JN9ACXX:1:2203:8207:180568#ATAGGAAG/1   BWA     Indel    - Reconstructed_IXR83416  85M1I14M      MD:Z:99
8  FCD0JN9ACXX:1:1203:4486:138933#ATAGGAAG/1   BWA     Indel    + Reconstructed_IXR83419  82M1I17M      MD:Z:99
9  FCD0JN9ACXX:1:1206:14304:69083#ATAGGAAG/1   BWA     Indel    + Reconstructed_IXR83420  81M1I18M      MD:Z:99
10 FCD0JN9ACXX:1:1306:7843:59305#ATAGGAAG/2    BWA     Indel    - Reconstructed_IXR83420  81M1I18M      MD:Z:99
11 FCD0JN9ACXX:1:2103:16165:39875#ATAGGAAG/1   BWA     Indel    + Reconstructed_IXR83431  70M1I29M      MD:Z:99
12 FCD0JN9ACXX:1:1206:10919:115013#ATAGGAAG/1  BWA     Indel    - Reconstructed_IXR83432  69M1I30M      MD:Z:99
13 FCD0JN9ACXX:1:1101:15533:94550#ATAGGAAG/2   BWA     Indel    + Reconstructed_IXR83458  43M1I56M      MD:Z:99
14 FCD0JN9ACXX:1:2102:1775:130120#ATAGGAAG/1    -       *          0         0      100
   FCD0JN9ACXX:1:2102:1775:130120#ATAGGAAG/1_1 Bowtie  SE       + IXR_BACseq         36824  16M           MD:Z:16
```