This tutorial describes the use of Scaffremodler tools to correct assemblies.

Go to scaffremodler folder (Scripts can be run from any folder but the command lines in this tutorial assume you are in this folder and that you have python version 2.6 or 2.7).

Available data:

data/scaffolds/ contained a multifasta file (scaffolds_to_correct.fasta) containing 8 sequences to correct

data/reads/ contained 2 fastq files (reads_mate1_rf.fq and reads_mate2_rf.fq) containing paired read (5 kb insert size) that will used to correct scaffold. The fastq files correspond to mate pair which expected orientation is reverse/forward (rf).

We assume that we suspect miss-assemblies in scaffold_2 and scaffold_3 between 978000 and 1090000 region and 981000 and 1020000 region respectively. These suspicions can be based on genetic map or physical map.

By default, options in all tools are tuned by default for this type of data. Changing dataset type (read orientation and insert size) will necessitate changing options when running scripts.

A - Refinement of scaffold miss-assemblies boundaries

1) Running scaffremodler_wrapper.py (step 1 to 6) to look for regions not overlapped by paired reads that will help refine the region of miss-assembly identified by genetic/physical map:

```
python bin/scaffremodler_wrapper.py --ref data/scaffolds/scaffolds_to_correct.fasta --q1 data/reads/reads_mate1_rf.fq --q2 data/reads/reads_mate2_rf.fq --step 123456 --tool bowtie2_single --orient rf --prefix check_assembly
```

2) Preparing files for circos drawing of paired reads in the suspected miss-assembled zone:

```
python bin/conf4circos.py --cov check_assembly.cov --chr check_assembly.chrom --window 1000 --orient rf --liste_read check_assembly.list --dis_prop check_assembly.prop --ref data/scaffolds/scaffolds_to_correct.fasta --prefix circos_check_assembly
```

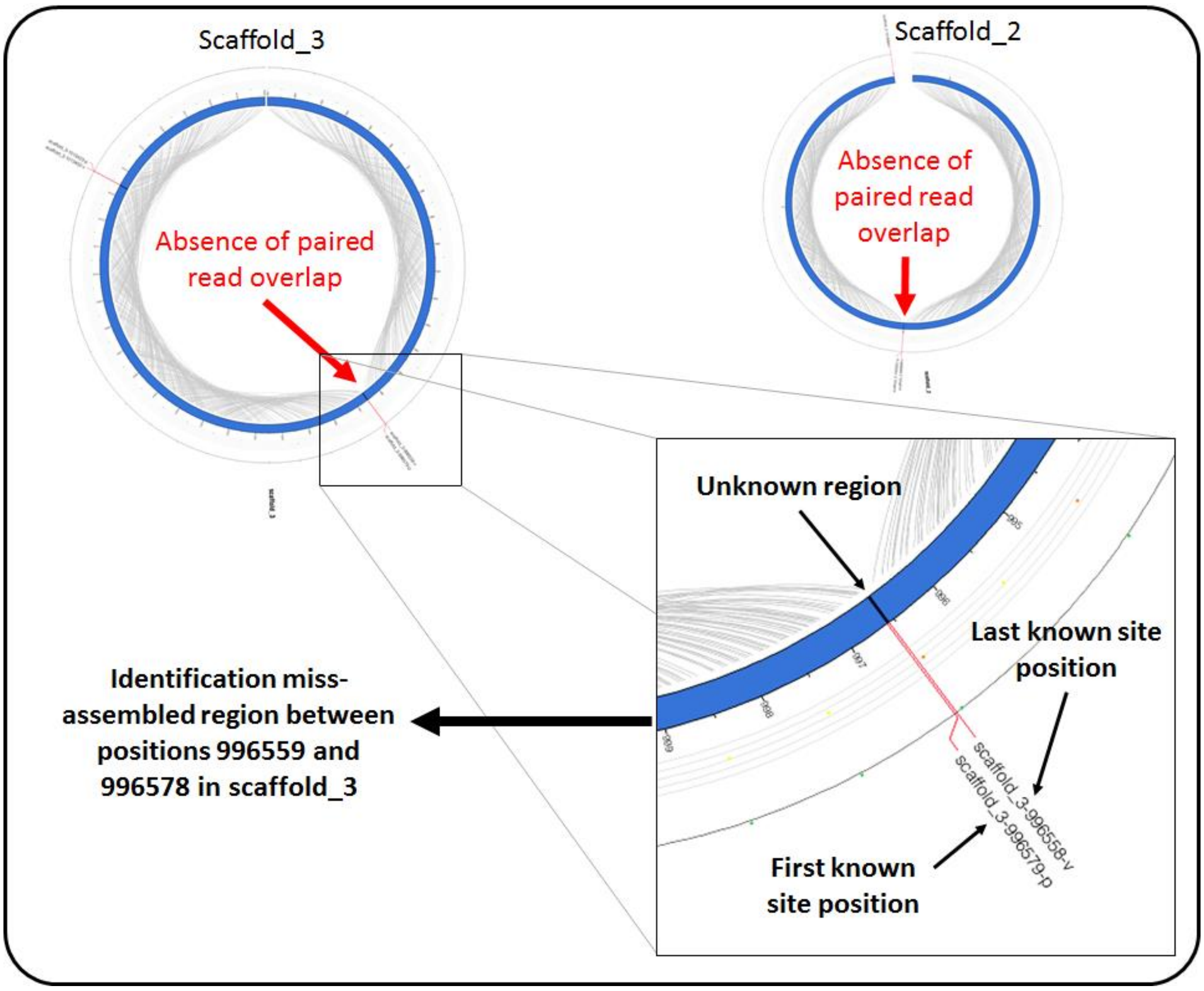
3) Visualization of paired reads in the suspected miss-assembled regions:

```
python bin/draw_circos.py --config circos_check_assembly.conf --out scaffold_2_miss.png --draw scaffold_2:978000:1090000
```

```
python bin/draw_circos.py --config circos_check_assembly.conf --out scaffold_3_miss.png --draw scaffold_3:981000:1020000
```

When looking at circos picture, in both observed regions, reads do not overlap a DNA fragment containing unknown sites (Figure 1). These are the region of miss-assembly that needed to be split.

Reading circos text layers (Figure 1), these regions of unknown sites can be well identified and splitting can be applied on these unknown sites.



Scaffold_3
Scaffold_2
Absence of paired read overlap
Absence of paired read overlap
Unknown region
Last known site position
First known site position
scaffold_3-996558-v
scaffold_3-996579-p
Identification miss-assembled region between positions 996559 and 996578 in scaffold_3

Figure 1 : Circos picture obtained in suspected miss-assembled regions of scaffold_2 and scaffold_3. All paired reads aligning in the zone are drawn with color coded link (grey : concordant read, other color: discordant reads; see Annex 1 for color coding details). The absence of paired reads overlapping regions indicate miss-assembly that can be either missing sequence or junction of two regions that should not. This information in addition to the suspicion of miss-assembly in these zone based on genetic/physical map support the second hypothesis.

To do so a file containing splitting zone should be created. This file should contain the following information:

```
scaffold_2        1000000
scaffold_3        996559          996578
```

Create this file; it will be named scaffold_to_split.txt in the following command lines.

4) Splitting scaffold on specified regions in scaffold_to_split.txt
   - Converting zone to split in "X" in the sequence

```
python bin/convert2X.py --table scaffold_to_split.txt --fasta data/scaffolds/scaffolds_to_correct.fasta --out X_converted.fasta
```

   - Splitting scaffold sequence on "X"

```
python bin/SplitOnX.py --fasta X_converted.fasta --out scaffold_split.fasta
```

B - Identification of scaffold fusions and junctions to perform on the newly split scaffolds

1) Running scaffremodler_wrapper.py to identify significantly linked non-contiguous regions:

```
python bin/scaffremodler_wrapper.py --ref scaffold_split.fasta --q1 data/reads/reads_mate1_rf.fq --q2 data/reads/reads_mate2_rf.fq --step 1234567 --tool bowtie2_single --orient rf --prefix check_assembly2
```

2) Preparing file for identification of scaffold fusions and junctions:

```
python bin/conf4circos.py --cov check_assembly2.cov --chr check_assembly2.chrom --window 1000 --frf check_assembly2_fr.score --ff check_assembly2_ff.score --rr check_assembly2_rr.score --ins check_assembly2_ins.score --delet check_assembly2_del.score --chr_rr check_assembly2_chr_rr.score --chr_rf check_assembly2_chr_rf.score --chr_ff check_assembly2_chr_ff.score --chr_fr check_assembly2_chr_fr.score --orient rf --liste_read check_assembly2.list --dis_prop check_assembly2.prop --ref scaffold_split.fasta --prefix circos_check_assembly2
```

3) Visualization of significantly linked non-contiguous regions (facultative):

```
python bin/draw_circos.py --config circos_check_assembly2.conf --read_fr n --read_rf n --read_ff n --read_rr n --read_ins n --read_delet n --read_chr_rr n --read_chr_rf n --read_chr_fr n --read_chr_ff n --text n --out discordant_zones.png
```

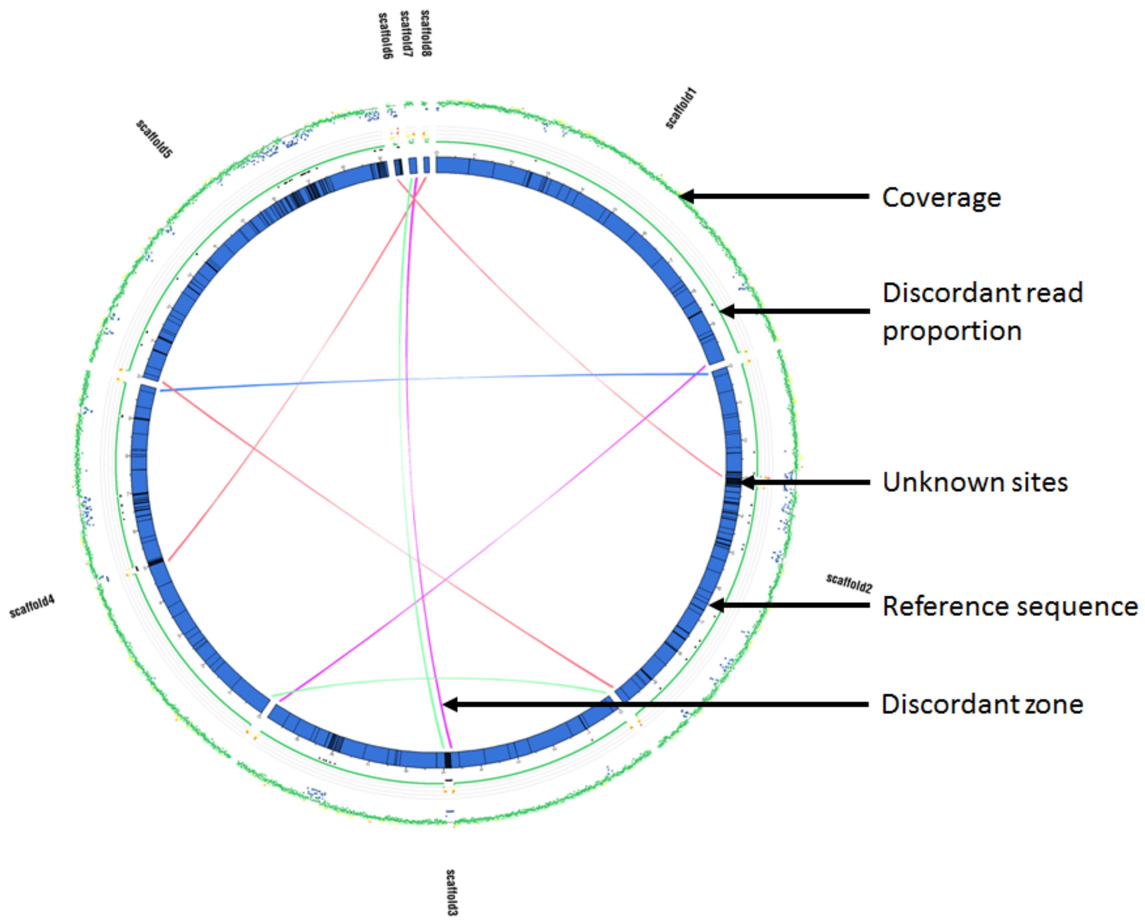


Figure 2 : Circos file representing linked non-contiguous regions identified by scaffremodler_wrapper.py

This circos picture shows non-contiguous regions linked by discordant read pairs. The links are color coded depending on the type of discordant reads (orientation and insert size). Interpretation of these links (automatically performed in next point) allows identifying possible scaffold fusions and junctions to correct the assembly.

4) Identification of possible junction and fusion to perform

```
python bin/look4fusion.py --config circos_check_assembly2.conf --bound 10000 --out possible_fusion.txt --out_tar possible_fusion.tar
```

This script identify possible fusion and junction to perform and return a table file (here possible_fusion.txt) containing possible editing to perform.

Table 1: Table file indicating possible junctions and fusions to perform.

scaffold3	1	996558	FWD	scaffold1	999999	999999	FWD	REV	contig
scaffold4	1	992957	FWD	scaffold2	1	1	FWD	FWD	contig
scaffold4	1	992957	FWD	scaffold3	1	1	FWD	REV	contig
scaffold5	1	914761	FWD	scaffold2	999377	999377	FWD	FWD	contig
scaffold6	1	20001	FWD	scaffold2	299998	319379	FWD	FWD	prob_fusion
scaffold7	1	19999	FWD	scaffold3	480000	496560	FWD	REV	fusion
scaffold7	1	19999	FWD	scaffold3	480000	496560	FWD	REV	prob_fusion
scaffold8	1	13848	FWD	scaffold4	499999	511143	FWD	FWD	prob_fusion

This file should be interpreted in addition to the circos pictures generated and archived in the other file generated by look4fusion.py (here possible_fusion.tar).

The following point will explain how to interpret the table file together with circos files generated.

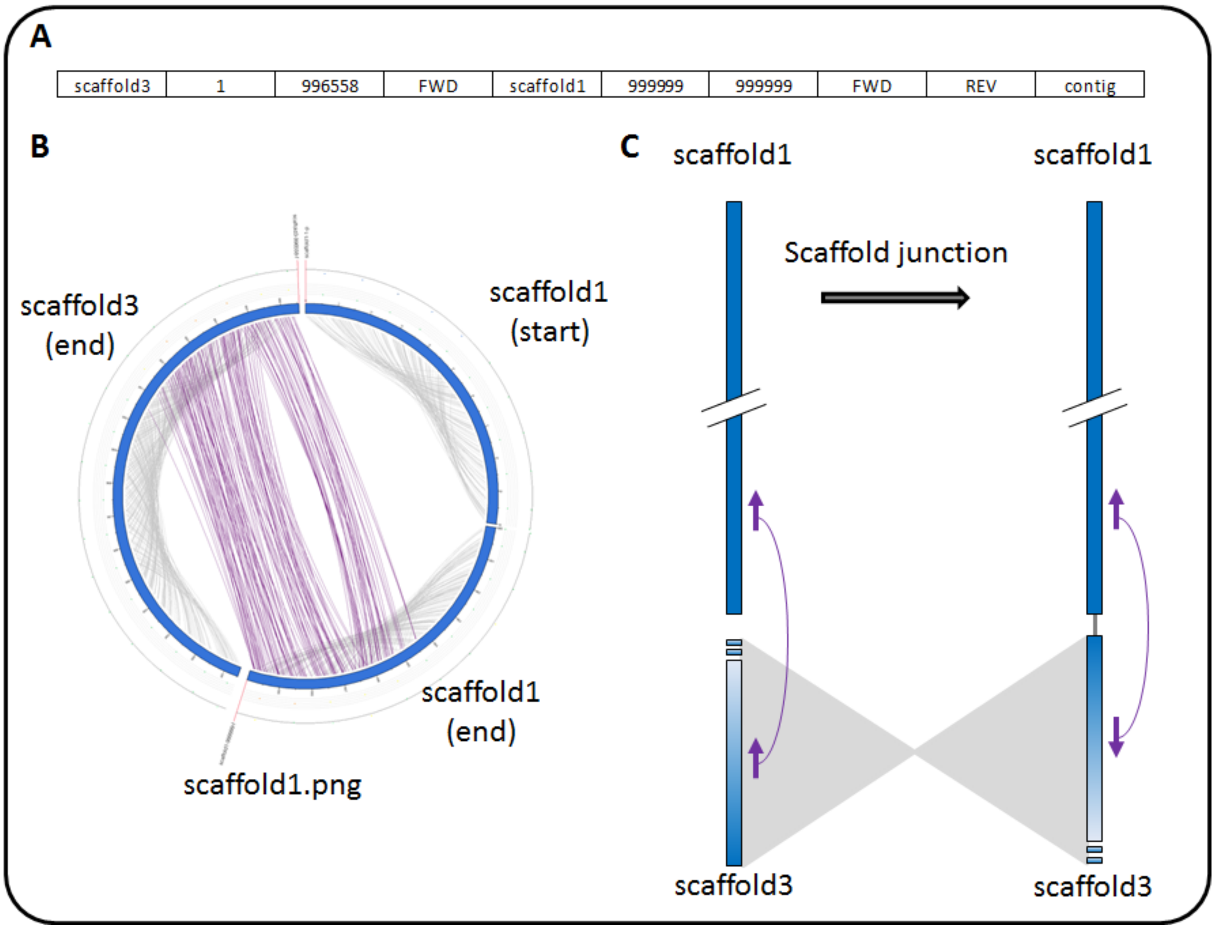
The table file is a ten column file containing several informations:

- Columns 1 to 4 contain information on the first scaffold to join or scaffold to insert into another one. From columns 1 to 4 respectively, scaffold name, scaffold start, scaffold end and original scaffold orientation in the multifasta (forward = FWD; reverse = REV).
- Columns 5 to 8 contain information on the second scaffold to join or where to insert the first scaffold. From columns 5 to 8 respectively, scaffold name, scaffold start, scaffold end and original scaffold orientation in the multifasta (forward = FWD; reverse = REV).
- Column 9 tells the way (orientation) first scaffold should be joined or inserted with/into second scaffold (forward = FWD; reverse = REV).
- Column 10 tells the type of operation to perform. There are four possible values:
  - "contig" : correspond to scaffold junction. The relative position of the junction (first scaffold before second scaffold or first scaffold after second scaffold) is indicated by columns 6 and 7 (that are identical in this case). If the value is "1" first scaffold is before second scaffold, else it is after.
  - "fusion" : correspond to insertion of first scaffold into second scaffold into the zone specified by columns 6 and 7
  - "prob_fusion" : correspond to probable scaffold junction the signature is not complete. Indeed in fusion case, two discordant regions are associated corresponding to both boundaries of the fusion. In these cases, only a boundary has been found but this boundary is associated to a region with unknown sites where first scaffold can be inserted. This can happen when the unknow region where the scaffold should be inserted is larger than the fragment to insert.
  - "removed : no N found" : Correspond to possible scaffold fusion signature but in these cases there are no unknown sites present in the region where the scaffold should be inserted. This case is not found in our example as we decided that they are not valid.

The .tar file is an archive containing circos pictures (one circos for each scaffold) depicting for each scaffold extremities (10 kb around boundaries --bound 10000 option in look4fusion.py) reads linking significantly these extremities to others scaffolds.

The table file should be checked line by line together with the corresponding circos picture; the fisrt column of the line tells you the circos picture to look. For example: if the firts column of the first line in the table file tells you to check this line together with the scaffold1.png file in possible_fusion.tar, for the second line it is scaffold2.png.

Example 1: scaffold junction.



scaffold3	1	996558	FWD	scaffold1	999999	999999	FWD	REV	contig

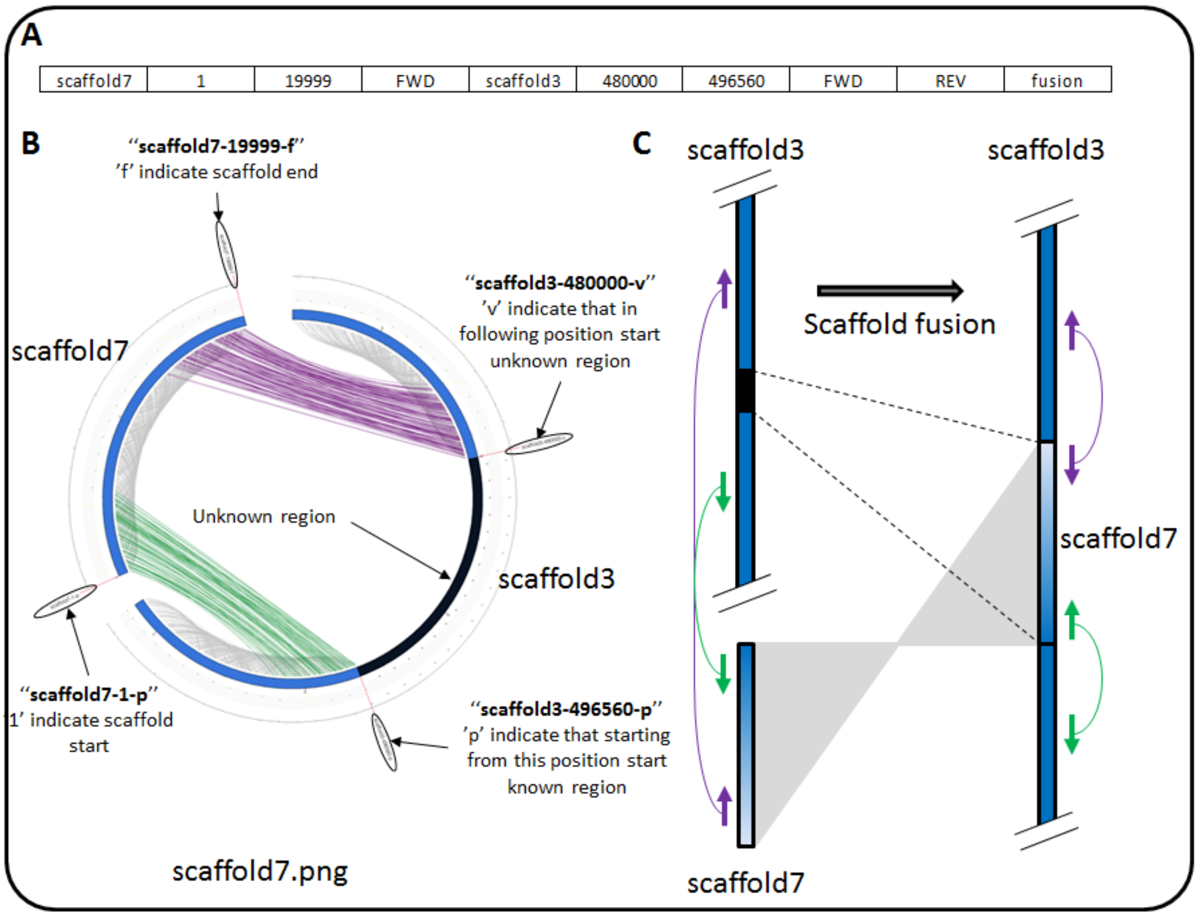
Figure 3 : Example of scaffold junction.

This example (first line of the table file – Figure 3A) tells that scaffold3 (column 1) should be joined (column 10) in reverse complement orientation (column 9) to scaffold1 (column 5) at its end (column 6 and 7). Interpretation of circos picture (Figure 3B) validates this event has we can observe discordant links (purple links) linking the end of scaffold1 to the end of scaffold3. The junction, in reverse complement orientation, of scaffold3 at the end of scaffold1 solves the discordant reads that were previously in reverse-reverse orientation on two different sequences. When joining these two scaffolds the discordant reads are in good orientation (reverse-forward) good insert size and on the same sequence (Figure 3C). When the joining is performed 100 N are inserted between the two joined fragments. This N number will be re-estimated in a next step.

Example 2: scaffold fusion (type 1).

This example (7th line of the table file – Figure 4A) tells that scaffold7 (column 1) should be inserted (column 10) in reverse complement orientation (column 9) into scaffold3 (column 5) between 480000 (column 6) position and 496560 (column 7). Interpretation of circos picture (Figure 4B) validates this event has we can observe discordant green links linking start of scaffold7 into scaffold3 and discordant purple links linking end of scaffold7 into scaffold3. In addition these discordant links suggested that scaffold7 should be integrated in an unknown region of scaffold3 indicated by black color on the band representing the scaffold3. Boundaries of the unknown region are indicated on the text layer (Figure 4B). The insertion, in reverse complement orientation, of scaffold7 into this unknown region of scaffold3 solves the discordant reads that were previously in reverse-reverse (purple) and forward-forward (green) orientation (Figure 4C). In addition, the unknown sites of scaffold3 are filled with known sites of scaffold7. Scaffold7 will be inserted at the center of the region specified in column 6 and 7 of the table file and two cases can be found:

- if scaffold7 size is ≥ specified insertion region, all N will be removed and replaced by 20 ‘n’ at each boundaries of the insertion site.
- else, scaffold7 is inserted in the center of the unknown region and N number at each boundaries is re-estimated to reach the exact size of previously unfilled region. These N are equally distributed at each boundaries of the insertion site.



scaffold7	1	19999	FWD	scaffold3	480000	496560	FWD	REV	fusion

A
B
“scaffold7-19999-f”
’f’ indicate scaffold end
scaffold7
“scaffold3-480000-v”
’v’ indicate that in following position start unknown region
Unknown region
scaffold3
“scaffold7-1-p”
’1’ indicate scaffold start
“scaffold3-496560-p”
’p’ indicate that starting from this position start known region
scaffold7.png
C
scaffold3
scaffold3
Scaffold fusion
scaffold7
scaffold7

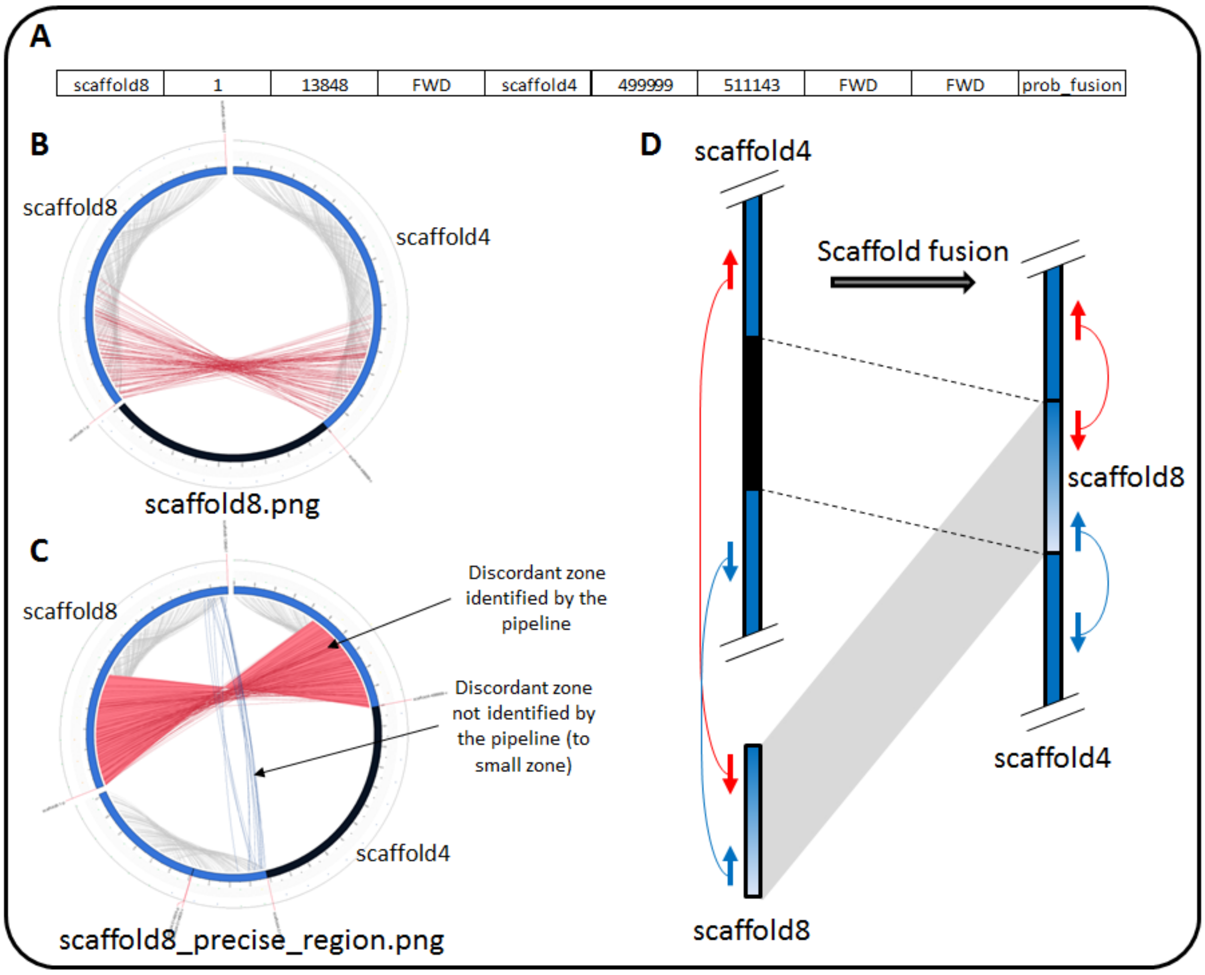
Figure 4 : Example of scaffold fusion1.

Example 3: scaffold fusion (type 2).

This example (8th line of the table file – Figure 5A) tells that scaffold8 (column 1) may be inserted (column 10) in forward orientation (column 9) into scaffold4 (column 5) between 499999 (column 6) position and 511143 (column 7). Interpretation of circos picture (Figure 5B) validates this event has we can only observe discordant red links linking start of scaffold8 into scaffold4 near an unknown region of scaffold4 indicated by black color on the band representing the scaffold4.

In normal case of fusion the other extremity of scaffold8 should be significantly linked to the other boundary of the unknown region of scaffold4. This is not the case here. Two explanations can be brought to explain the absence of such second discordant zone:

- the unknown zone of scaffold4 can be larger than solely scaffold8 and in this case, all or a part of read at the other extremity of scaffold8 can go to another scaffold or to unassembled part. If only a part of read goes to the other extremity of unknown region of scaffold4 they may not be sufficient to identify a significant link between this region and scaffold8.
- the observed discordant red links linking start of scaffold8 into scaffold4 are not revealing a miss-assembly but rather reflect a mappability problem due to duplicated regions (duplicate genes, transposable element, …) and should thus not be used to correct assembly.



scaffold8	1	13848	FWD	scaffold4	499999	511143	FWD	FWD	prob_fusion

A
B
scaffold8
scaffold4
scaffold8.png
C
scaffold8
Discordant zone identified by the pipeline
Discordant zone not identified by the pipeline (to small zone)
scaffold4
scaffold8_precise_region.png
D
scaffold4
Scaffold fusion
scaffold8
scaffold4
scaffold8

Figure 5 : Example of scaffold fusion2.

At this point, it is up to the user to decide what to do. To help user to decide, reads linking the other boundary of the unknown region of scaffold4 and scaffold8 can be searched and inspected by drawing the circos picture comprising all scaffold8 and boundaries of unknown region of scaffold4 (that are reported in column 6 and 7 in the bale file). To do so, run the following command:

```
python bin/draw_circos.py --config circos_check_assembly2.conf --out scaffold8_precise_region.png
--draw scaffold8:1:13848-scaffold4:489999:521143
```

In this command, by default, all reads and discordant zones will be drawn. The --draw argument “scaffold8=1=13848=scaffold4=489999=521143” instruct de program to draw scaffold8 from position 1 to 13848 (the complete scaffold8) and to draw scaffold4 from position 489999 to 521143 that correspond to the unknown region ± 10 kb. The obtained figure is presented in Figure 5C. Discordant reads linking the other extremity of scaffold8 and boundary of unknown region of scaffold4. This picture shows that the pipeline identifies the discordant zone of one extremity (red band) but there is not enough reads to validate the other boundary (blue links). In this context we can validate the fusion proposed by the program and once fusion is performed, the insertion, in forward orientation, of scaffold8 into this unknown region of scaffold4 solves the discordant reads that were previously in reverse-forward (red) and forward-reverse (blue) orientation (Figure 5D).

This validation step should be performed for all lines in the table file. For this set the finale table file is the following:

scaffold3	1	996558	FWD	scaffold1	999999	999999	FWD	REV	contig	OK
scaffold4	1	992957	FWD	scaffold2	1	1	FWD	FWD	contig	OK
scaffold4	1	992957	FWD	scaffold3	1	1	FWD	REV	contig	OK
scaffold5	1	914761	FWD	scaffold2	999377	999377	FWD	FWD	contig	OK
scaffold6	1	20001	FWD	scaffold2	299998	319379	FWD	FWD	prob_fusion	OK
scaffold7	1	19999	FWD	scaffold3	480000	496560	FWD	REV	fusion	OK
~~scaffold7~~	~~1~~	~~19999~~	~~FWD~~	~~scaffold3~~	~~480000~~	~~496560~~	~~FWD~~	~~REV~~	~~prob_fusion~~	Already treated
scaffold8	1	13848	FWD	scaffold4	499999	511143	FWD	FWD	prob_fusion	OK

C – Performing scaffold fusions and junctions

As scaffold junctions and fusions are treated separately, lines corresponding to junctions (contig) and fusions (fusion and prob_fusion) **should be separated**. **In this example contig lines will be in a file called to_contig.txt and the other lines (fusion and prob_fusion) will be put in a file called to_merge.txt.**

For scaffold fusions and junctions, as fusion step uses positions to insert scaffold this step should be performed before junction step.

a – Scaffold fusions:

Put the to_merge.txt in the current directory and run the following command:

```
python bin/fusion_scaff.py --table to_merge.txt --fasta scaffold_split.fasta --out fused_scaffold.fasta --out_verif fused_scaffold.verif
```

There are two output files:

- fused_scaffold.fasta contains the new scaffold version, including scaffolds that have not been edited.
- fused_scaffold.verif is a table file containing informations on edited scaffolds. The first column give scaffold name and the following columns recapitulate this scaffold composition.

The fusion performed can be verified (optional) by running:

1) scaffremodler_wrapper.py program on the new multifasta:

```
python bin/scaffremodler_wrapper.py --ref fused_scaffold.fasta --q1 data/reads/reads_mate1_rf.fq --q2 data/reads/reads_mate2_rf.fq --step 123456 --tool bowtie2_single --orient rf --prefix verif_fused_scaffold
```

2) running conf4circos.py:

```
python bin/conf4circos.py --cov verif_fused_scaffold.cov --chr verif_fused_scaffold.chrom --window 1000 --orient rf --liste_read verif_fused_scaffold.list --dis_prop verif_fused_scaffold.prop --ref fused_scaffold.fasta --prefix circos_verif_fused_scaffold
```

3) running verif_fusion.py script:

```
python bin/verif_fusion.py --config circos_verif_fused_scaffold.conf --list fused_scaffold.verif --bound 30000 --out_tar verif_fused_scaffold.tar
```

The archive file obtained (tar file of --out_tar argument) contained a circos figure representing paired reads overlapping the boundary of a fusion. As a fusion event comprise two boundaries, for each event two circos figures are drawn.

For example when checking the insertion of scaffold7 into scaffold3 the two circos file are represented in Figure 6. Unless the fusion zone is at an extremity of a scaffold, the fusion boundary at the bottom of the picture. If you are unsure that you are looking at the fusion boundary the position of the fusion zone is indicated in the text layer that indicate respectively end of the first scaffold and beginning of the next scaffold (Figure 6B). In figure 6 we observe read showing a correct orientation overlapping both fusion zone extremity (Figure 6A and C). It is important to note that in this example the reads are concordant (grey) but for this verification they could either have been discordant of insertion (orange) or deletion (red) as the size of the fusion zone (quantity of unknown sites between the scaffold3 and scaffold7) has not been recalculated. In the other side, if read overlapping the fusion zones have been discordant of orientation (forward-forward: green, reverse-reverse: purple or forward-reverse: blue), this will have meant that an error has been introduced and the fusion performed is incorrect.

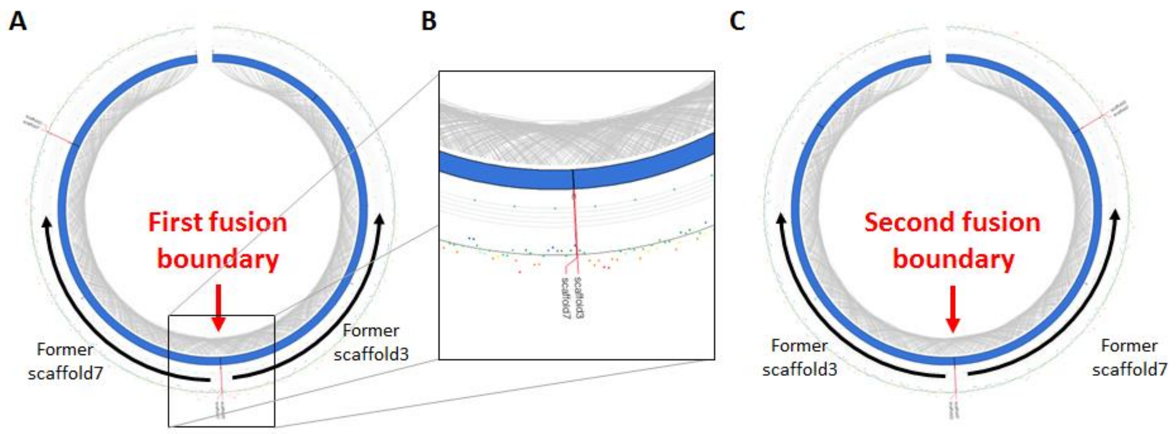


Figure 6 : Circos representation showing paired reads at the boundaries of the insertion of scaffold7 into scaffold3.

b – Scaffold junctions:

For scaffold junctions, scaffolds are automatically ordered one after the other using the to_contig.txt file containing pairwise junctions to perform. This step can be done running the following command:

```
python bin/group4contig.py --table to_contig.txt --out to_contig_grouped.txt
```

The output file is a table file having the following lines:

>scaffold1	
scaffold1	FWD
scaffold3	REV
scaffold4	FWD
scaffold2	FWD
scaffold5	FWD

This file suggest that scaffold1, 2, 3, 4 and 5 could be joined together in this top to bottom order with the suggested orientation (FWD : forward, REV : reverse-complement) to obtain a single sequence that will be called scaffold1.

In this example all scaffolds can be grouped together, however for large genomes, not all scaffolds will be grouped together. In this case each line beginning with ">" define a new group of scaffold and the output file is as followed:

>scaffold1	
scaffold1	FWD
scaffold3	REV
scaffold4	FWD
scaffold2	FWD
scaffold5	FWD
>scaffoldX	
scaffoldX	REV
scaffoldY	REV
scaffoldZ	FWD
>scaffold…	

However, sometime the program cannot decide which scaffold putting one after another (one scaffold extremity linked to more than one scaffold). This is the case if working on small scaffolds (shorter than read insert size). For these cases, the program take arbitrarily one scaffold and the other scaffold in reported at the end of the scaffold group in the output file as followed:

>scaffold1530				
scaffold450	FWD			
scaffold1530	FWD			
not_grouped	scaffold450	FWD	scaffold681	FWD

In this case, the end of scaffold450 is grouped to scaffold1530 and scaffold681. The program proposed to group scaffold450 and scaffold1530. Scaffold681 has not been grouped but the 4th line says that it could have been joined (in FWD orientation) after scaffold450 (in FWD orientation). It's up to the user to decide (by inspecting paired reads in circos figures) if these three scaffold should be grouped and in which order (scaffold 450 → 1530 → 681 or 450 → 681 → 1530) or if it's preferable to group scaffold681 rather than 1530. In any case, this file should be edited to remove the "not_grouped" line in order to reach the file format required for scaffold junction presented earlier.

Once this file is edited (not necessary in our example), scaffolds are joined by running the command:

```
python bin/contig_scaff.py --table to_contig_grouped.txt --fasta fused_scaffold.fasta --out super_contig.fasta --out_verif contig2verif.txt
```

The multifasta file containing all scaffold including joined ones are recorded in super_contig.fasta file. As scaffold fusions, scaffold junction performed can be verified (**optional**) running the commands:

1) scaffremodler_wrapper.py program on the new multifasta:

```
python bin/scaffremodler_wrapper.py --ref super_contig.fasta --q1 data/reads/reads_mate1_rf.fq --q2 data/reads/reads_mate2_rf.fq --step 123456 --tool bowtie2_single --orient rf --prefix verif_joined_scaffold
```

2) running conf4circos.py:

python bin/conf4circos.py --cov verif_joined_scaffold.cov --chr verif_joined_scaffold.chrom --window 1000 --orient rf --liste_read verif_joined_scaffold.list --dis_prop verif_joined_scaffold.prop --ref super_contig.fasta --prefix circos_verif_joined_scaffold

3) running verif_fusion.py script:

python bin/verif_fusion.py --config circos_verif_joined_scaffold.conf --list contig2verif.txt --bound 30000 --out_tar contig2verif.txt.tar

The outputs are identical to the fusion steps and can be interpreted the same way.

D – Re-estimating unknown regions

During junctions and fusions steps, unknown regions are not re-estimated. This steps aims at re-estimating the size of these regions once editing are performed. This re-estimation can be lauched as followed:

1) scaffremodler_wrapper.py program on the new multifasta:

python bin/scaffremodler_wrapper.py --ref super_contig.fasta --q1 data/reads/reads_mate1_rf.fq --q2 data/reads/reads_mate2_rf.fq --step 1234 --tool bowtie2_single --orient rf --prefix N-restimate

2) Re-estimation of unknown regions size:

python bin/reEstimateN.py --conf N-restimate.conf --min_read 10 --out re-estimated_with_E_1.fasta > re-estimated1.tab

This pipeline re-estimates the size of unknown regions using paired reads. As very close unknown regions can be found, reads overlapping two regions or more cannot be used to re-estimate insert size. To solves this problem, re-estimated unknown regions are converted into ‘E’ to allow the program to be run several time without re-estimating the same zone several time (only zone containing ‘n’ or ‘N’ are re-estimated). In this case, the first run estimates one of the unknown close region. Re-mapping reads onto this new reference with scaffremodler_wrapper.py and running a second time reEstimateN.py will allow to re-estimate the second close unknown region: in this case, as the first close zone has already been estimated, the first zone is no-more identified as an unknown region (replaced with E), and now paired read overlap only one unknown region (the second zone) which size could be re-estimated.

Second run

python bin/scaffremodler_wrapper.py --ref re-estimated_with_E_1.fasta --q1 data/reads/reads_mate1_rf.fq --q2 data/reads/reads_mate2_rf.fq --step 1234 --tool bowtie2_single --orient rf --prefix N-restimate2

python bin/reEstimateN.py --conf N-restimate2.conf --min_read 10 --out re-estimated_with_E_2.fasta > re-estimated2.tab

Finally ‘E’ inserted in the reference sequence after unknown region re-estimation can be converted into ‘N’ to correspond to the IUPAC code by running the following command line:

```
sed 's/E/N/g' re-estimated_with_E_2.fasta > corrected_reference_sequence.fasta
```

This reference sequence can be compared to the original reference sequence used to generate the the working dataset of this tutorial. With the exception of unknown regions that could vary of few bases, the two sequences should be identical.

Now, have fun with our own data!

Annex 1: Circos color coding.

All drawn links are color coded based on discordant type:

- Concordant read : grey
- Inter chromosomal discordance:

Reverse-forward : red
Forward-reverse : blue
Forward-forward : green
Reverse-reverse : purple

- Intra chromosomal discordance:

Insertion : red
Deletion : orange
Reverse-reverse : purple
Forward-forward : green

If expected orientation is reverse-forward :
    Forward-reverse : blue
If expected orientation is forward-reverse:
    Reverse-forward : blue