RAPID CURATION OF GENOMIC ASSEMBLIES

The focus of rapid curation is to identify and correct large scale scaffolding errors of genomic assemblies. This is done by modifying an assembly via a Tile Path File (TPF) using evidence from Hi-C data.

Requirements

- Fasta file of genomic assembly
- Rapid curation scripts (https://gitlab.com/wtsi-grit/rapid-curation)
- Access to HiGlass (HiGlass)
- PretextMap (https://github.com/wtsi-hpag/PretextMap)
- PretextViewer (https://github.com/wtsi-hpag/PretextView)

Overview of workflow:

Convert your fasta file into a Tile Path File (TPF)

A Tile Path File (TPF) is a simple file that simply lists the order of clones along a chromosome. These files are often used in genome assemblies in an effort to convey mapping information to the assembly program. Gap types and sizes can be specified, along with desired orientation of the component sequences.
The TPF format supports rapid manipulation of the underlying components thereby allowing the user to efficiently reshape a genome assembly.
For further information, please refer to Tiling Path File (TPF) Specification v1.4: 2/15/2011

rapid_split.pl will create this file format for you.

Eg:
rapid_split.pl -fa <yourfastafilename>

The created TPF file is sorted into 4 tab separated columns as in the following example:

1	2	3	4
?	scaffold_1:1-10627236	scaffold_1	PLUS
GAP	TYPE-2	500	
?	scaffold_1:10627737-26896573	scaffold_1	PLUS
?	scaffold_2:1-2001924	scaffold_2	PLUS
GAP	TYPE-2	500	
?	scaffold_2:2002425-4716520	scaffold_2	PLUS
GAP	TYPE-2	500	
?	scaffold_2:4717021-10487029	scaffold_2	PLUS
GAP	TYPE-2	500	
?	scaffold_2:10487530-12959303	scaffold_2	PLUS
GAP	TYPE-2	500	
?	scaffold_2:12959804-16728812	scaffold_2	PLUS
GAP	TYPE-2	500	
?	scaffold_2:16729313-19399053	scaffold_2	PLUS
GAP	TYPE-2	500	
?	scaffold_2:19399554-23803696	scaffold_2	PLUS
GAP	TYPE-2	500	
?	scaffold_2:23804197-26337532	scaffold_2	PLUS
?	scaffold_3:1-20769075	scaffold_3	PLUS
GAP	TYPE-2	500	
?	scaffold_3:20769576-24535592	scaffold_3	PLUS
GAP	TYPE-2	500	
?	scaffold_3:24536093-25417327	scaffold_3	PLUS
?	scaffold_4:1-5801213	scaffold_4	PLUS
GAP	TYPE-2	500	
?	scaffold_4:5801714-14085539	scaffold_4	PLUS
GAP	TYPE-2	500	
?	scaffold_4:14086040-16309388	scaffold_4	PLUS
GAP	TYPE-2	500	
?	scaffold_4:16309889-19868079	scaffold_4	PLUS

Content within the rows can take on one of two formats; Sequence line and non-sequence lines.

For **Sequence lines** Column 1 will be ?
When TPF formats were originally created, Column 1 was used for the Genbank sequence identifier. For our rapid curation purpose, this is not applicable, but an entry into column 1 is still required, hence we use the ?
Column 2 gives the scaffold Name followed by coordinates of the scaffold piece
Column 3 gives the scaffold ID
Column 4 gives the orientation of the scaffold

The scaffold name and ID will come from the name which is assigned during the assembly process.

E.g:

```
?   scaffold_1:1-10627236   scaffold_1  PLUS
```

rapid_split.pl will create the TPF file of your sequences using their scaffold identifier followed by its coordinates. Where the script sees a run of N's within the fasta sequence, it will add a **non-sequence line** to indicate a gap is present, with a default size of 500bp.
Column 1 states it is a gap
Column 2 states the gap type, in this process, we use only TYPE-2 gaps
Column 3 states the gap size
Column 4 is unused in this situation, but can be used to specify how the size of the gap was determined.

E.g:

```
GAP TYPE-2  500
```

Use PretextView to manipulate your assembly, and generate an AGP file

See PretextView - Tutorial for detailed instructions on use of pretext.
Arrange scaffolds into updated order.

For an explanation of AGP files please refer to AGP Specification v2.1

Where gaps exist in the assembly, you do not need to change anything in your TPF file in order for the script to incorporate breaks and joins correctly.
If there is no gap in the TPF where you wish to make an update, you would need to manually insert a gap into your TPF, editing the coordinates of the scaffold accordingly. There is no requirement to rearrange the TPF as the script does this.

For example, if you wished to insert a scaffold into scaffold_1 at position 10982492 where there are no N's within the assembly, your TPF would originally look like this.

```
?       scaffold_1:1-14982492       scaffold_1      PLUS
```

You would have to edit your TPF file to make it look like this:

```
?       scaffold_1:1-10982492       scaffold_1      PLUS
GAP     TYPE-2          200
?       scaffold_1:10982493-14982492        scaffold_1      PLUS
```

You can then paint the scaffolds into chromosomal units, ordered and oriented, and generate an AGP as described in the pretextview instructions.

Convert your AGP file to an updated TPF

In order to make an updated fasta for submission, we first need to have an updated TPF. To do this, Trim the AGP file which you produced from Pretext so that only the scaffolds remain which you have identified to be chromosomes.

Use rapid_pretext2tpf.py <original_tpf> <Your_trimmed.agp>

Eg:
rapid_pretext2tpf.py iyLasLatv2.tpf iyLasLatv2_trimmed.agp

This will produce a file named rapid_pretext.tpf which will have renamed the scaffolds which you painted in pretext, in size order, using the prefix RAPID_ and also contains the remaining scaffolds of the assembly with their original names.

Anywhere a join has been made between scaffolds which you have specified using the scaffold painting in PretextView will be assigned a default gap size of 200bp. We use this so that we can easily see which changes we have made during the curation process.

When the script runs successfully you should get a printed message within the terminal window. This contains some information and statistics about your assembly and will look like this:

```
length ilSpiLutu1.edited.tpf            584,908,687 bp  (input)
length rapid_prtxt.tpf                  584,908,687 bp  (output)

Texel length:                   17,850bp                (texels in map: 32,767)
Break count:                    3
Join count:                     9
Interventions per Gb:           12              (very low)
```

Create a .csv file

We then need to create a simple .csv file to tell the rapid joining script which scaffolds in your updated TPF are associated with chromosomes.
In this instance, the .csv file is essentially a list of your renamed scaffolds which you are confident are chromosomes.
In the simplest of cases where you have for example 5 chromosomes, this would look like this:

RAPID_1
RAPID_2
RAPID_3
RAPID_4
RAPID_5

If you have scaffolds in your TPF which you have identified as belonging to a chromosome, but are unsure of the exact placement on that chromosome, then these can be added to the .csv file **on the same line as the chromosome they are associated with. They will be named as belonging to that chromosome, but unlocalised.**

For the same example as above where we have 5 chromosomes, but also some unlocalised scaffolds, the .csv file would look like this:

RAPID_1
RAPID_2,scaffold_35,scaffold_62
RAPID_3
RAPID_4
RAPID_5,scaffold_143

You also need to make sure that there are no empty spaces, line breaks or additional characters within the .csv or the next script will fail.

If you have managed to identify one or both sex chromosomes, then the names of these need to be manually altered within the TPF and .csv file.

For example where RAPID_1 was identified as Z:

```
?       scaffold_1:1-14982492          RAPID_1       PLUS
GAP     TYPE-2          500
?       scaffold_1:14982993-32961442       RAPID_1       PLUS
GAP     TYPE-2          500
?       scaffold_1:32961943-39057171       RAPID_1       PLUS
GAP     TYPE-2          500
?       scaffold_1:39057672-41590670       RAPID_1       PLUS
```

Would become:

```
?       scaffold_1:1-14982492          Z       PLUS
GAP     TYPE-2          500
?       scaffold_1:14982993-32961442   Z       PLUS
GAP     TYPE-2          500
?       scaffold_1:32961943-39057171   Z       PLUS
GAP     TYPE-2          500
?       scaffold_1:39057672-41590670   Z       PLUS
```

The .csv file would then look like this:

```
Z
RAPID_2
RAPID_3
RAPID_4
RAPID_5
```

Dealing with haplotypes

If you have identified any haplotypic duplications, these will need to be removed from the rapid_pretext.tpf and placed on a separate haplotigs.tpf.

Create files for submission

In order to create the files needed for submission, the following files are required:

- Original fasta file
- Your updated tpf (rapid_pretext.tpf)
- Your .csv file

The script takes the original assembly fasta file, a one-line per chromosome csv file and a TPF and creates the updated assembly from the TPF.

This is done by using rapid_join.pl

rapid_join.pl -fa <original.fasta> -tpf <your tpf file> -csv <your.csv> -out <outfile.fasta>

You do not need to use the -out option. If you don't then files will be written to <original.fasta**.new**>
Eg:

rapid_join.pl -fa -fa iyLasLatv2.original.fa -tpf rapid_prtxt.tpf -csv iyLasLatv.csv -out iyLasLatv2

You will now have an updated, curated version of your assembly ready for submission, and a related .csv file indicating the chromosome names.

If you have created a haplotigs.tpf, you will also needs to run rapid_join.pl on this TPF in order to create the haplotigs fasta file for submission:

rapid_join.pl -fa <input.fa> -tpf <haplotype.tpf> -out <outfile> -hap

Assembly statistics

If you wish to know how many manual interventions you have made to your assembly, there are scripts to do this for you:

To count rapid breaks:

python rapid_count_breaks.py original.tpf rapid_prtxt.tpf haps.tpf

To count rapid joins:

grep 'GAP'$'\t''TYPE-2'$'\t''200' rapid_prtxt.tpf | wc -l