Scaffold alignment characterization

Fernando Rodriguez

Information of two genome assemblies were used to generate the final assembly of the genome of Zerene cesonia (Z_cesonia_v-2). The first assembly (Z_cesonia_v-1) has a total length of ~229Mb and lacks the Z chromosome. The second assembly (Z_cesonia_v-d1) is a diploid assembly of length ~516.4Mb. The final assembly was constructed in three steps:

1) Split haplomes from Z_cesonia_v-d1 using Haplomerger
2) Align the haplomerger output to Z_cesonia_v-1
3) Identify and remove remaining duplicates using the alignment and a custom python script.

The output of Haplomerger generated a reference haplome of length ~381Mb, which is ~110Mb longer than the expected genome size. To identify the origin of this excess, the Haplomerger output was aligned with the Z_cesonia_v-1 assembly. This alignment revealed large duplicated sequences in various scaffolds (Fig 1) corresponding tho the ~110Mb excess.

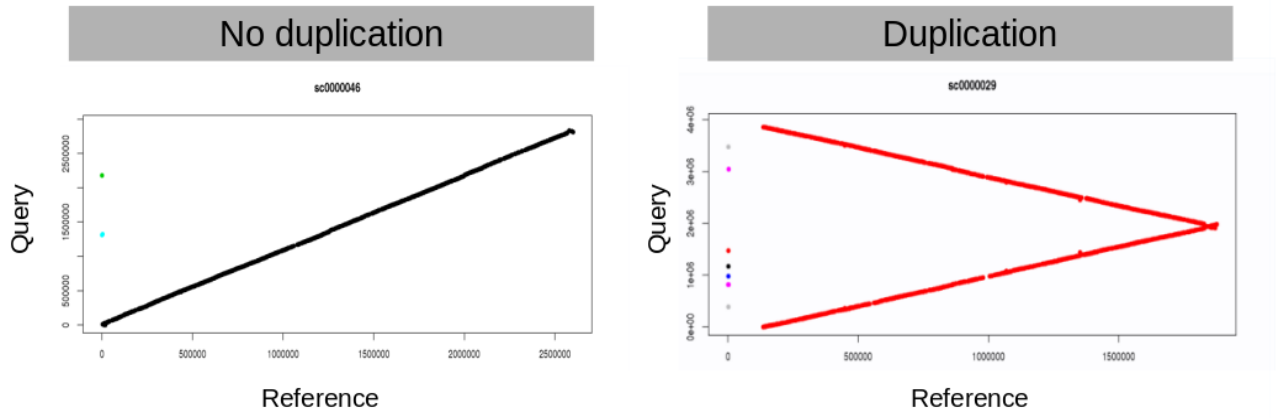

Fig1 Example of alignments showing scaffolds with and without duplicated sequence revealed by the MUMmer alignment.

A python script was designed to search and remove the duplicated fragments in the Haplomerger output using information from the MUMmer alignment. The pseudo-code below describe the main proceding of this script

```
Duplicated ← empty set
Breaking_points ← empty set

for scaffold in genome:
        category ← FindCategory(scaffold alignment)
        if category == Duplicated:
                add to Duplicated set

for scaffold in Duplicated:
        breaking point ← IdentifyBreakingPoints(scaffold alignment)
        add scaffold and breaking_points to Breaking_points

return Breaking_points
```

Box 1: Pseudo-code for removing duplicated regions

1. Alignment categorization - *FindCategory()*

The purpose first step of the algorithm is to categorized scaffolds based on the proprieties of their alignment. Three levels are are used to categorize scaffold alignments: i) mosaic, ii) linear and iii) duplicated (Fig. 1.1). Alignments with substantial fragments mapping to multiple reference scaffolds are considered **mosaic alignments** and filtered out for manual curation. **Linear alignments** are those in which the query aligned consistently to a single reference scaffold and did not show evidence of duplication. **Duplicated** alignments are those in which the query scaffold aligned consistently to a single reference but have evidence of duplication (<, >, or Z shape). Figure 1.1 provides examples of

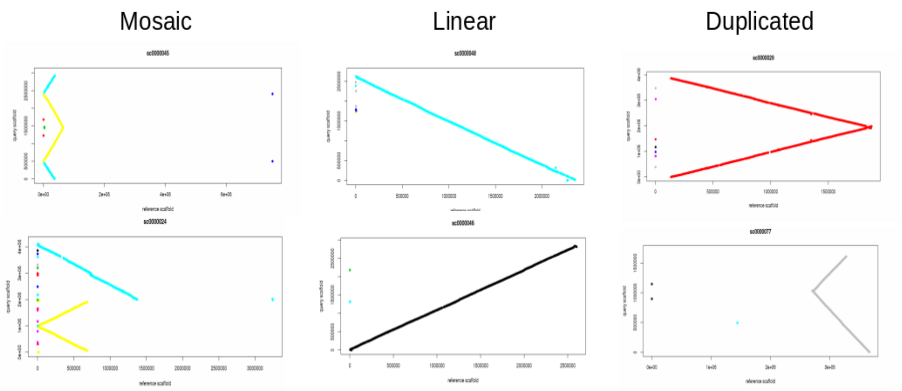

Figure 1.1 Examples of Mosaic, Linear and Duplicated scaffold alignments. Different colors correspond to different reference scaffolds

The pseudo-code below, describes the procedure of the *FindCathegory()* function used to categorize scaffolds

```
FindCathegory():

        Alignment ← mummer alignment hits for query scaffold
        Dominant reference ← reference with larger number of hits in Alignment
        M ← number of hits in dominant reference
        n ← total number of hits in Alignment
        D ← number of hits in the dominant direction in Dominant reference

        If M/n < 0.95:
                return Mosaic
        else:
                if D/M >= 0.98:
                        return Linear
                else:
                        return Duplicated
```

Box 2: Psudo-code for the FindCathegory function.

Two proprieties of the alignment were used to categorize alignments: **integrity (i)** and **continuity (c).** Integrity measures the proportion of hits that correspond to the major reference scaffolds that the query aligns to. Integrity score represent the proportion of hits to the major alignment

$$i = \frac{M}{n}$$

M = Number of hits to the dominant alignment
n = Total number of hits

Continuity refers to the consistence of alignment directionality. It ranges from 0.5 to 1 where 0.5 indicates that half of the hits map in the forward direction with respect to the reference and 1 indicates that all of the hits map in the same direction.

$$c = \frac{D}{M}$$

D = Number of hits in the dominant direction in the dominant reference

M = Number of hits in the dominant reference

The table below shows the criteria used to categorize scaffolds based on the proprieties of their alignments, and the number of scaffolds identified on each category using such thresholds .

	Mosaic	Linear	Duplicated
i	< 0.95	≥ 0.95	≥ 0.95
c	NA	≥0.98	<0.98
Number of scaffolds	12	255	216

Table 1.1 Criteria and results of scaffold characterization

2. Identification of braking point - *IdentifyBreakingPoint(scaffold)*

The second step in the algorithm is to identify the point in the scaffold where the duplication starts (breaking point). In all scaffolds, the shortest duplicated fragment was removed as shown in figure 2.1.

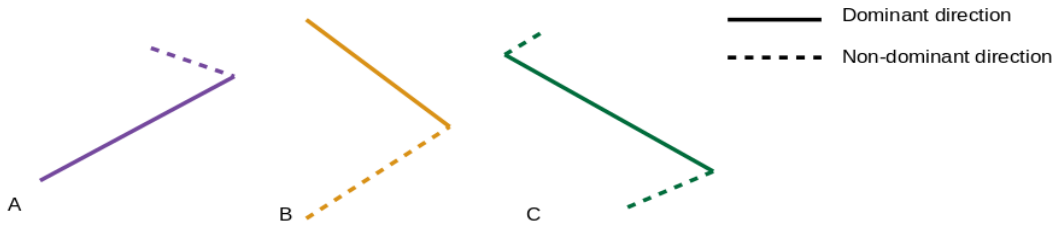

Figure 2.1 fragment of scaffold to be removed. In all cases both duplicated regions where meassured and the shortest one was removed. Examples of alignment types in duplicated scaffolds: **A)** A "hook" shaped alignment with about 30% of hits mapping in the reverse direction. **B)** A 'V' shaped alignment with equal sized duplicates. In these cases the fraction removed was chosen randomly. **C)** a zig-zag shaped alignment with about 40% of hits mapping in the forward direction.

One of the proprieties of the breaking point is a change on the direction of the alignment, however because of small biological rearrangements and inversions, many nodes in the alignment graph share this feature, therefore all the points in the alignment with a switch in direction were considered candidate break points and then, a score was assigned to each to identify the best candidate. The score was assigned based on the length and consistency of the "arms" formed by braking the alignment on each braking point. The best candidate was the one that formed the longer and more consistent arms as shown in figure 2.2

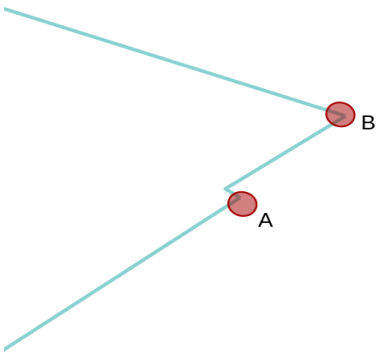

Figure 2.2 Identifying the true breaking point by scoring each candidate based on the consistency and length of the arms it forms. In this case The candidate the candidate point B receives the highest score as it forms the longest and consistent "arms" if the alignment is divided at B.

To find candidate brake points, the alignment was represented as an array of '+1' and '-1' representing alignment hits. A positive value indicates that the alignment hit occurs in the forward direction with respect to the reference the negative value correspond to alignments in the reverse drection (Figure 2.2). Candidate breaking points are identified as pairs of nodes with opposite directions (+1,-1 or -1,+1)

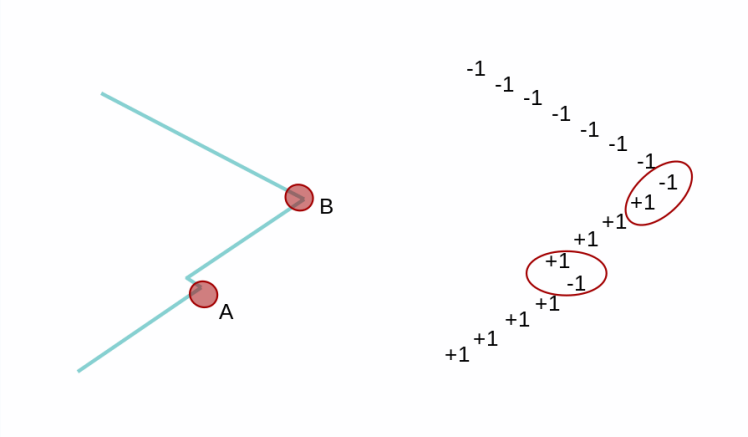

Figure 3.2 Representation of the alignment as an array of +1 and -1 and identification of candidate breaking points. Note that candidate breakpoints are identified as pair of nodes with different signs (+1, -1).

The pseudo code below shows the procedure used to identify the breaking point of each alignment with duplicated fragments.

```
IdentifyBreakingPoint():

    Candidates ← empty list

    for i in alignment hits index:
        if  direction(i) is different than direction(i+1):
            add i to Candidates

    for Point in Candidates:
        score left ← sum of all values to the left of point in alignment
        score right ← sum of all values to the right of point in alignment
        score ← sum of absolute values of left and right scores

    return Point with higher score in Candidates
```

Box 3: Pseudo-code for the IdentifyBreakingPoint function.

The score of each candidate break point was calculated in two steps: **1)** each arm formed by the candidate breaking point was scored by adding the values to its left and to its right separately. Note that the presence of opposite signs on the same arm, reduces the absolute value of the score of each arm. **2)** Adding the absolute values of both arms.

$$a = \sum (l_i)$$
$$b = \sum (r_i)$$
$$S = |a| + |b|$$

l_i = individual nodes to the left of the candidate
r_i = individual nodes to the right of the candidate

The true node will score higher than any other candidate since its arms are the longest and and consistent in direction (Fig 3.3)

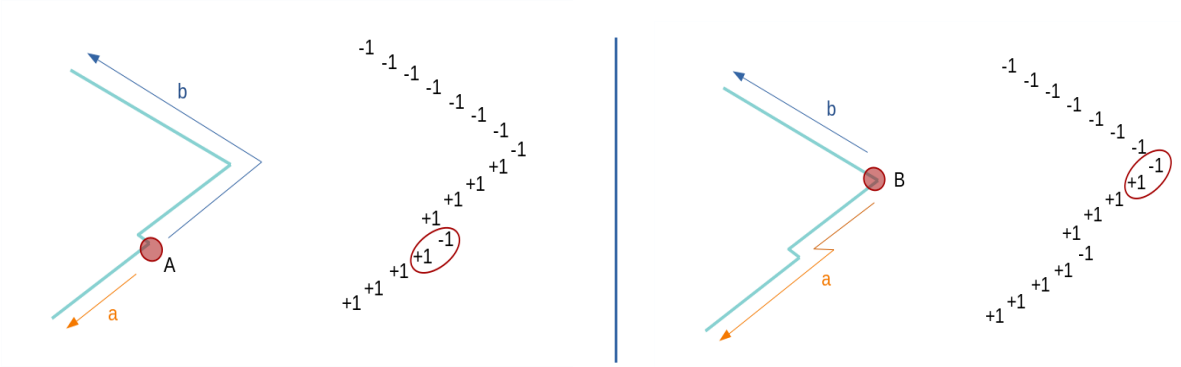

Figure 3.3 Scoring candidate nodes. The score of the candidate break point A is S = | 4 | + |-5| = 9. Despite of the length of the right arm, the score of this arm decreases because of the inconsistency of the directions of its nodes. In contrast, the score for the candidate B is S = |+8| + |-8| = 16.