SUPPLEMENTARY MATERIAL

1. Variant Representation Issues In Family Trios

Different variant representations can cause two types of problems in Mendelian Violation assessment:

1.1. Wrong Violation Decision Example

In the following example, we illustrated how the method of line-by-line checking can report wrongly identify violations. The mistake is made in the second variant.

Chr	Pos	Ref	Alt		M	F	C
1	165592	GA	G	GT	0/0	0/1	0/0
1	166160	TTATATA	T,TTTTATA	GT	0/0	1/2	1/2
1	166162	A	T	GT	0/1	0/0	0/0
1	166189	C	G	GT	0/1	0/1	0/1

If the prefix(red) and suffix(blue) of second allele is trimmed from variant as shown below, we see that second alternate allele of first variant and the alternate allele of the second variant are identical.

1	166160	**TT**A**TATA**	T,**TT**T**TATA**	GT	0/0	1/2	1/2
1	166162	A	T	GT	0/1	0/0	0/0

The two variants above can be written as a single line as follows, which can be correctly identified as Mendelian consistent by line-by-line checking:

1	166160	TTATATA	T,TTTTATA	GT	0/2	1/2	1/2

1.2. Missing Violation Decision Example

In the following example, we illustrate a case where line-by-line check misses a violation.

Chr	Pos	Ref	Alt		M	F	C
1	8835006	CTTTCT	C	GT	0/0	0/0	0/1
1	8835010	C	T	GT	0/0	0/1	0/0

If we did strict line-by-line checking we would mistakenly conclude that the second line is Mendelian consistent. In reality, these variants, after considering them together, can be written as a single line as follows, exhibiting Mendelian violation.

```
1    8835006  CTTTCT   C, CTTTTT    GT   0/0  0/2  0/1
```

2. Violation Validation Pipeline

2.1. Preparing the Inputs for Validation

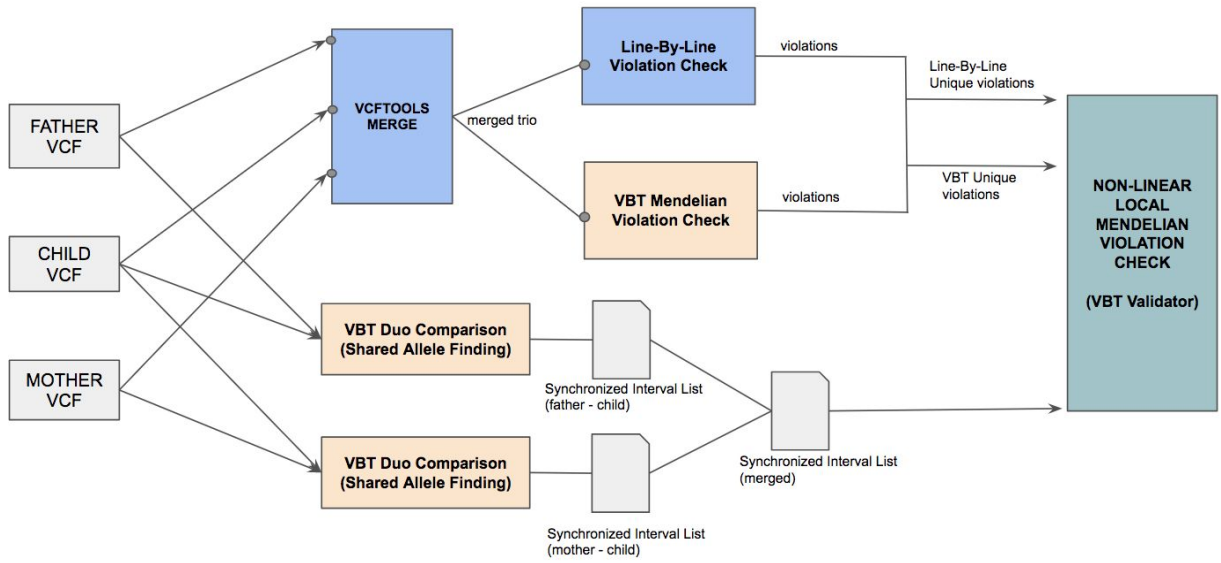

Figure 2: VBT violation validation pipeline. Unique violations of VBT and Naive method are assessed for correctness along the provided small interval list

In order to assess VBT results, we implement a validation pipeline (Figure 2) including a nonlinear local Mendelian violation tool. Naive line-by-line checking tools accept only trio vcfs, so we first merge the three vcfs using Vcftools merge. After obtaining the trio vcf, we run VBT and our line-by-line checking method and identify the records where their results differ.

To determine each small region that violations belong to, we used ***sync points***, which we obtained from duo variant comparison. A ***sync point*** is a location where baseline and called side are at the same position and it does not overlap with any variant while applying variants to the reference sequence. Two consecutive *sync points* delimit a region, and variants inside each region are independent from variants in other regions. In order to obtain these points, we run duo comparison tool in ***allele matching mode*** (ga4gh benchmarking method 2) for mother-child and father-child vcfs and then, we take the common sync points from these two point set.

2.2 Nonlinear Local Mendelian Violation Check

We used two-step method to evaluate violation calls of VBT and our line-by-line checking method (LBL) according to the following pseudo code:

1. **FOR EACH** violation region: *<run VBT and LBL separately>*
    a. **IF** any consistent variant overlaps with a violation
        i. **Set** region as *Falsely Assessed*
    b. **ELSE**
        i. Clip the substring between two sync points from the reference
        ii. Remove violation variants from region
        iii. Apply all possible variant combinations (in order or in reverse order) to the clipped substring. (Total of $2^M + 2^F + 2^C$ variants are generated where M, F and C represents the consistent variant count of mother, father and child in that region)
        iv. **IF** there is a combination that is Mendelian consistent (i.e. mother string is equal to child *string-1* and father string is equal to child *string-2*) **THEN**
            SET region as *Correctly Assessed*
            ELSE
            SET region as *Falsely Assessed*

2. **FOR EACH** violation region:
    a. **IF** LBL is *Falsely Assessed* **AND** VBT is *Falsely Assessed*
        i. **Increment** VBT false decision count
        ii. **Increment** LBL false decision count
    b. **IF** LBL is *Falsely Assessed* **AND** VBT is *Correctly Assessed*
        i. **Increment** VBT true decision count
        ii. **Increment** LBL false decision count
    c. **IF** LBL is *Correctly Assessed* **AND** VBT is *Falsely Assessed*
        i. **Increment** VBT false decision count
        ii. **Increment** LBL true decision count
    d. **IF** LBL is *Correctly Assessed* **AND** VBT is *Correctly Assessed*
    AND LBL violation count is **GREATER THAN** VBT violation count
        i. **Increment** VBT true decision count
        ii. **Increment** LBL false decision count
    e. **IF** LBL is *Correctly Assessed* **AND** VBT is *Correctly Assessed*
    AND LBL violation count is **EQUAL TO** VBT violation count
        i. **Increment** VBT true decision count
        ii. **Increment** LBL true decision count
    f. **IF** LBL is *Correctly Assessed* **AND** VBT is *Correctly Assessed*
    AND LBL violation count is **LESS THAN** VBT violation count

i. **Increment** VBT false decision count
ii. **Increment** LBL true decision count

2.3 Discussion : Small Region Selection for Validation Pipeline

By Mendelian inheritance rule, one full haplotype string between parent and child chromosome is required to be identical. Using this information, we run the best-path algorithm of vcfeval in shared allele mode between parent and child and obtain a single haplotype string with maximum number of included variants together with the list of *sync point*s.

Performing allele match operation to obtain *sync points* does not always give us the correct splitting. Since it is possible that both variants of the two parents can match to the same child allele, those variants need to be removed from the maximum path and that may change the sync point location for some regions.

During our test, we observed several examples of wrongly divided regions, where two consecutive regions are marked wrong by VBT, but merging them would result in a correct decision. We do not eliminate these regions from the result to keep the validation pipeline simple.