**Black Magik Tool Box Automated Analysis Pipeline Manual**

A phenotype-blind, automated, agnostic pipeline that allows high-confidence genotyping in regions of low coverage and/or low variant quality score and permits analysis for undiagnosed diseases in coding and noncoding regions for variant Mendelian models, including homozygous recessive, de novo dominant, compound heterozygous recessive, X-linked, hemizygous, and, in very well-sequenced positions, novel exon deletions. This pipeline uses 7 modules developed by Lukas Vlahos, Kayla Gu, Anchi Wu, Faris Aziz, and Tom Markello to produce a short output list of deleterious variants as candidates for rare diseases.

**Table of Contents**

1. Background
2. Preprocessing
3. Requirements
4. Relevant Programs
5. Inputs
6. Outputs
7. User Guide
8. Configuration Files
9. Ethnicity (Ancestry) Matching Module
10. Salvage Pathway Module
11. KaylaKode Module
12. Automated Broad-level BAM Curation/SNR Calculation Module
13. BAM Curation and Variant Exclusion Filter Module
14. Pedigree-Aware, Multi-Parametric BAM File Noise Evaluator (Confetti Filter) Module
15. Extreme Novel Exon Deletion Filter (Call no Call Filter) Module
16. References
17. **Background**

This monolithic analysis program was the end result of 10 years of trial and revision of the application of genetic inheritance, population (quantitative) genetics and structural implications of mutations (the “protein folding” problem and conservation over species or evidence of selective pressure). The original analysis was manual using the construction of discrete mathematical Boolean operations on a set of variants generated by Next-Gen Sequencing as presented by the NHGRI intramural Sequencing Center (NISC). Eventually, this was evolved into an iterative solution using both VarSifter and VCF formats and data (originally just CADD scores) in a modified version of data originating from NISC. Very important additions were made in 2015-2017 by Lukas Vlahos, Kayla Gu, Anchi Wu, and Faris Aziz. The most important addressed a major problem with false negative heterozygous genotypes being called homozygous reference in a parent which led to a large excess of apparent false positive de novo mutations in the raw NISC data for trio cases of two parents and a child. An almost equally important problem addressed by this work was the automatic (re)interrogation of the BAM file pileups from sequence alignments. This work has required significant manpower in the past and was onerous at best and at worst was left to individuals with minimal and inconsistent genetic experience. Eventually, sufficient knowledge was gained while performing these tasks manually to generate a set of working rules that follow basic segregation genetics and Hardy Weinberg equilibrium principles. Along with the addition of noise analysis reduction tools, this code is now capable of autonomously analyzing genome-wide data in a sensitive and robust manner to detect variants of interest for rare genetic disorders.

1. **Preprocessing**

The following procedures were performed to create a flat file with single annotation metadata per column.

1. Obtain DNA from patients (draw blood).
2. Obtain sequencing raw data (short read FASTQ).
3. Align the data into a sorted BAM file. These must be stored and be available for the analysis part of the process. Different alignment methods are available – graph aligner, dragen, NISC, etc.
4. Genotype the aligned sorted BAM file to obtain a list of called variant genotypes (a vcf file) - using standard genotyping tools that (mostly) genotype each BAM file independently (one at a time individual genotyping), or at most use a local small cohort for allele frequency priors (like haplotype caller in GATK). Jointly genotype all individuals in the file at every place at least one individual has a variant from the reference genotype (i.e. a jointly genotyped vcf file).
5. Unroll into a format that has a single row per variant allele.
6. Annotate the unrolled vcf file using a standard UNIX program (i.e. Annovar or snpEFF) with a global list of known transcripts.
7. Open this file in the VarSifter GUI and save it (unpacks the annotation INFO field). This converts the Vcf file into a VarSifter (.vs) file.
8. Convert VarSifter file to the NISC conventions and cross-correlate the chromosome and variant positions in the VarSifter and VCF formats by running code that performs that task (called "Line by Line").
9. Annotate to a flat file with single annotation metadata per column, using (prior) assembled publicly available data sets, private commercial annotations (HGMD) and send previously unscored variants to CADD and Polyphen2 (nonsynonymous variants from the transcript annotation procedure. HGMD annotation is optional but improves the analysis. A list of publicly available data sets can be provided. The program that can do this with individual files is called the "Editor" and the newer program that can do multiple annotations simultaneously is called the Shebang editor. The editor manual should describe more in detail about how to perform this annotation process.The final polishing for never seen before CADD scoring and Polyphen2 scoring is described in detail elsewhere.

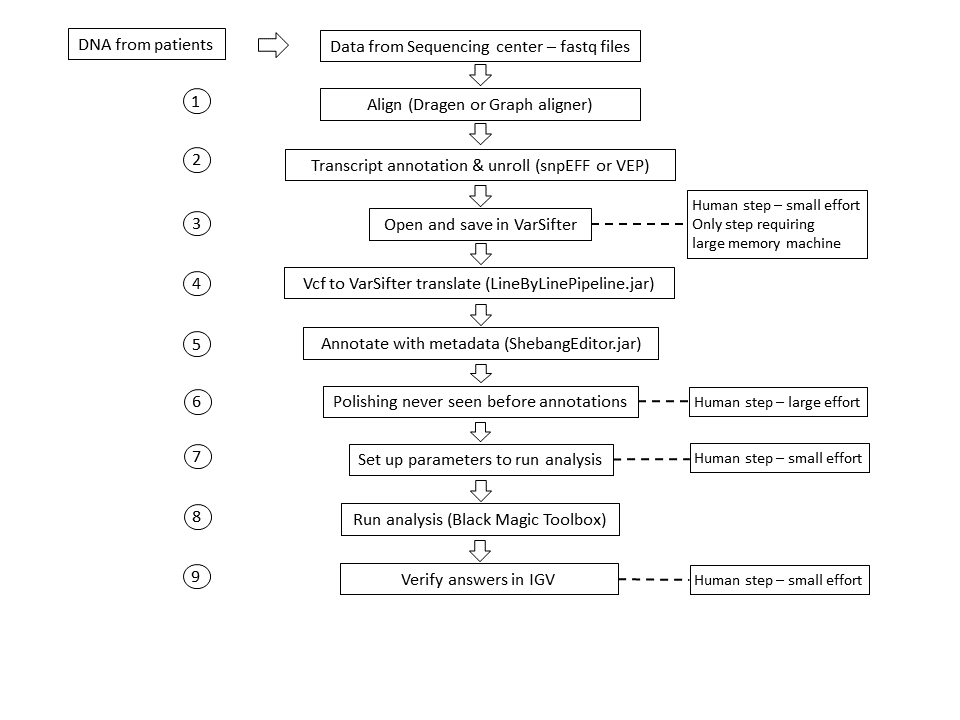
****

Figure 1. *Complete process of analysis pipeline*

1. **Requirements**

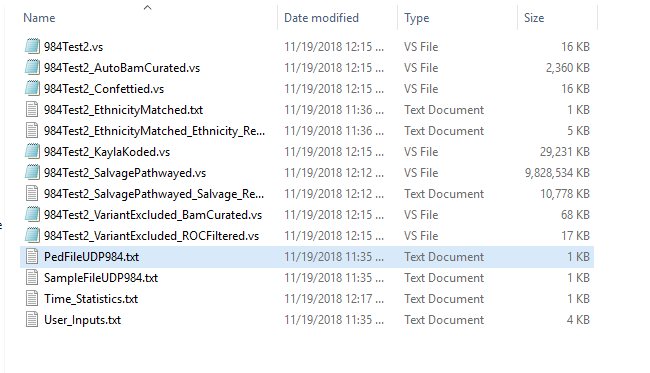
* This code requires Java JDK 1.7 or newer, though Java JDK 1.8 is recommended.
* The SAMTools1 Java utilities must be available for this program to be compiled using the source code.

1. **Relevant Programs**

* Jar File: BlackMagikToolBox.jar
* Source Code: BlackMagikToolBox

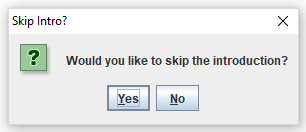
1. **Inputs**
2. A dataset: This can consist of a VarSifter file from NISC, or can be generated from any vcf format file by annotating it with a transcript in a program like snpEff if it is “unrolled” to have one line per variant. To make use of the population analysis parts of this process, an internal dataset should be used. This is available from the “All\_Gahl” statistics of 1400 exomes. VCF formatted files that are annotated by snpEFF can be opened and saved in VarSifter and then are run through the Line By Line program converting them to “nearly the same” as a NISC derived file. An “editor” program can add the statistics from 1400 exomes or the 300 jointly called genomes from the UDP’s internal dataset for example that have been created for this purpose. It is always best if a local cohort of at least 300 other individuals have been sequenced using the same chemistry and technology and aligned and genotyped in the same way. The editor would also annotate the VarSifter file with new metadata from external databases such as CADD and ExAC as well. **The VarSifter file must be preprocessed with Line By Line and the Editor prior to running the Black Box**
3. Output file name. Please do not include the file extension as that will already be taken care of by the program.
4. A “Configs” directory folder containing all the configuration files for all the modules. These configuration files include:
   1. Directory file of the family’s BAM files in the format “ID \t BAM File Location”. This must be named “BAM\_Directory\_Config.txt”.
      1. The **BAM Files themselves must be aligned and indexed and must incorporate the optional MD tags** either during alignment or can be added using SAM tools after alignment. The file paths must be known within the BAM Directory Config and there must be read permissions to access these BAM files.
   2. Ancestry informative marker configuration folders. Please see below:
      1. The config folders are the marker set folders. The EASMarker, AMRMarker, EURMarker, AFRMarker should each include the complete 8 sets of markers for each ethnicity. For example, AMRMarker folder should contain AMR1.txt, AMR2.txt, AMR3.txt, AMR4.txt, unAMR1.txt, unAMR2.txt, unAMR3.txt, unAMR4.txt).
      2. These marker sets are a list of uncorrelated markers very frequently found in exome datasets (likely to be called exome data), which have orthogonal frequencies in one ethnicity used in the 1000 genomes project. There are several independent datasets that are used to cross validate this work. The purpose of this marker set of sets is to determine whether an individual has sufficient likelihood of belonging to a subpopulation that the use of ethnic matched allele frequencies as the prior probability for a Bayesian genotype caller is warranted. This dataset was generated once and is available to be used for this purpose. It’s accuracy is sufficient for this purpose but it is not intended to be used for ancestry informative conclusions.
   3. Strength Configuration file. This must be named “Strength\_Config.txt”.
      1. This is a simple config file that assigns a score for whether a variant causes a more severe change in a codon (like a stop or a nonsynonymous change) or a less severe (such as a synonymous change in a codon) or is intronic. It is used in the compound heterozygous recessive analysis.
   4. Gene Configuration file. This must be named “Gene\_Config.txt”.
      1. This file generates four positions for each gene transcript from the hg19 (GRCh37) reference genome coordinates. The original spacing was 1000bp upstream of the stated 5’ start site and 1000 bp downstream of the end of the 3’UTR. Several manual iterations of compound heterozygous recessive pair analyses were run for a few months and when cases of pairs were made with two different gene names on the transcript information fields, these loci were interrogated and manual changes to the positions were made to remove conflict spacing between two closely apposed gene loci. In addition, a single “gap” could be opened inside a larger locus to accommodate a smaller locus that existed entirely inside of an intron of the larger “outer” locus – often on the opposite strand. This gene config is not perfect but very functional, resulting in very few compound heterozygous recessive pairings of variants residing in two different loci.
   5. ROC Header Configuration file. The config file must be in the format “<category>=<VS header>”. This must be named “BamROC\_HeaderConfig.txt”.
      1. This file lists all the headers in the flat file (the VarSifter file) that can be used for all the different filter functions in the analysis (the module originally called KaylaKode). These headers classify the different types of columns that would be used for different functions as the analysis is performed, and to exclude variants which cannot be solutions for a rare genetic disorder.
   6. Exon Boundary Config.This must be named “Exon\_Boundary\_Config.txt”.
      1. This file contains a complete list of all the exon boundaries in the human genome. If the user chooses to filter non-exonic CNCs (There is a toggle option), this would compare the list of final CNCs that come through the filters to the boundaries. If a CNC fits within any of these bounds, then it will be kept within the final variant list as an extreme novel exon deletion.
   7. BAM Chromosome Config file. This config must be named “BAM\_Chr\_Config.txt”.
      1. NISC generates bam files that have header indexes for chromosomes with the characters “chr” in front of the chromosome numbers and does not use the same exact for gl’s or chrUn notations that many other alignment and genotyping processes produce. This config contains both sets of nomenclature to allow the program (which tests the headers of the BAM files) to select the appropriate data structure index names to fit the BAM files being analyzed.
5. Genotype Coverage Filter Criteria
6. Minor Allele Filter where 1 denotes ref is the minor allele.
7. 1000 genomes allele frequency fields or other similar database that can be inputted manually through the GUI.
8. Pedigree of family. This will be input through the GUI.
9. Population statistics. This will be input through the GUI. Additionally, the required minimum population statistics will differ depending on if the proband is male or female because of extra filters needed for X-Linked conditions.
10. If the proband is male, optional filtering for hemizygous external database columns can be added.
11. Exonic CADD threshold. A CADD score of either 19 or 20 is recommended. This is the value for exonic variants and is scaled for intronic variants and for the VMM scores proportionately in fixed ratios (see below in Variant Exclusion Filter).
12. Cutoff thresholds for Homozygous Variant, Heterozygous, and Hemizygous Variant found in the population.
13. Genotype cutoff amounts for every genotype column as specified in the ROC config file.
14. Population data for homozygous reference, homozygous variant, and genotypes column. Additionally, the program will ask for a genotype threshold for the genotype column that was chosen.
15. Filtering data for the cutoff amount to label a base quality as too low, a threshold for the number of bad base qualities to depict a read as a bad read, and a cutoff ratio for the maximum percentage of bad reads over the total read count.
16. CNC cluster size minimum. This must be a positive integer.
17. **Outputs**

The output of this process is a very small VarSifter file that can be opened using the VarSifter program on a small machine or most likely viewed in a simple text editor or excel spreadsheet containing a candidate list of deleterious variants that could be causing the rare disease in the proband. The output of this pipeline will be included in an output folder located in the same directory as the initial VarSifter file. The outputs after each module of the pipeline are also included in that folder, as well as information about the time statistics for the process, the user inputs, a ped file, and a sample ID table. Some programs may also release a secondary report output such as the Ethnicity Matcher and the Salvage Pathway. The Variant Exclusion Filter releases two outputs in which the \_ROCFiltered.vs output is the correct output to be sent to the next module. The final output file is VarSifter file that was initially inputted when asking for the output file name + “.vs”.



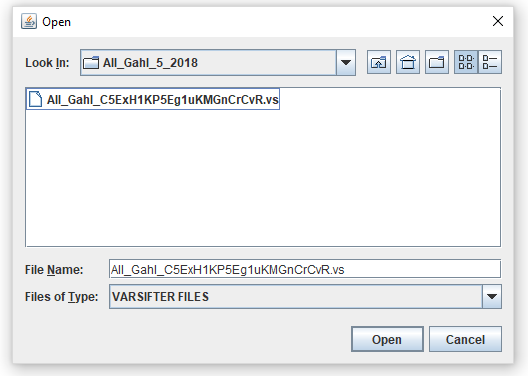
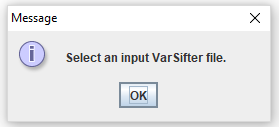
1. **User Guide**

**Step 1. User Introduction.**



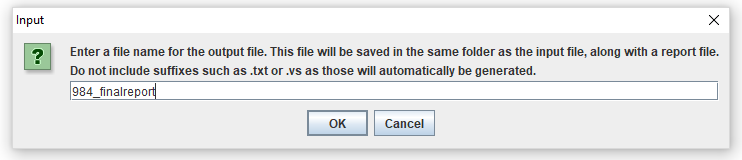
**Step 2. Input the VarSifter File.**

The user will be prompted to input the initial VarSifter File which will be sent through the pipeline.



**Step 3. Input the output file name.**

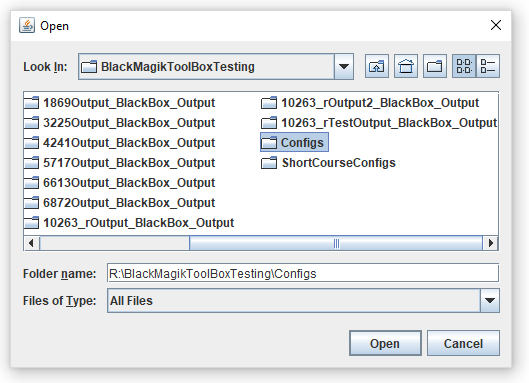
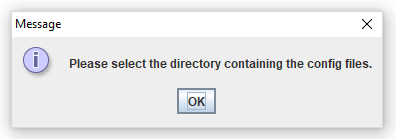
The user will be asked to enter the name for the final output file. This will also be the prefix to the name of the Output folder for the Black Box as well as every single output released by the modules in the Black Box. Please do not include suffixes such as .txt or .vs as those will already be accounted for in the program.

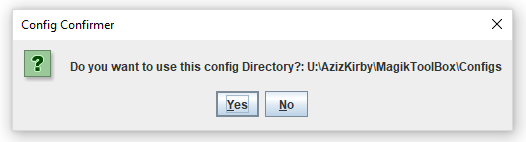


Blah Blah Caption

**Step 4. Input the Config Directory.**

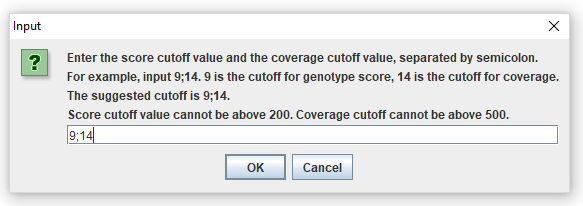
If no config directory with the name “Configs” was placed in the same folder as the jar file, the program will ask the user to navigate to the config directory. If there was a config directory in the same location as the jar, the user will simply have to validate that the config directory being used is the correct one. If not, it will make the user navigate to the correct config directory. More details about the configuration files themselves are in the “Configuration Files” and in the “Inputs” section. Please make sure the information in the Configuration files are updated.





**Step 5. Input the genotype filter criteria.**

The user will then be prompted to enter thresholds for the genotype score and genotype coverage cutoff values, separated by semicolon. The default threshold pairing is “9;14”



**Step 6. Input the minor allele filter.**

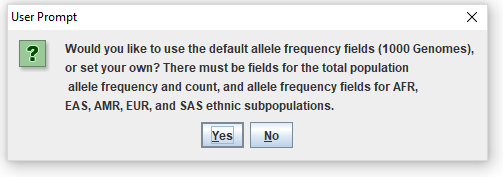
Minor allele filter: the field that indicates if the reference allele is the minor allele. 1 indicates that the reference allele is the minor allele (i.e., the-minor-allele-is-the-ref-allele, is true). 0 indicates that the reference allele is the major allele (i.e., the-minor-allele-is-the-ref-allele, is false). If the reference allele is the minor allele, the frequency filter for reference allele, and the raw data using ref allele counts are used instead of alternate allele.

* + Header examples: “All\_Gahlref\_is\_minor = 1”

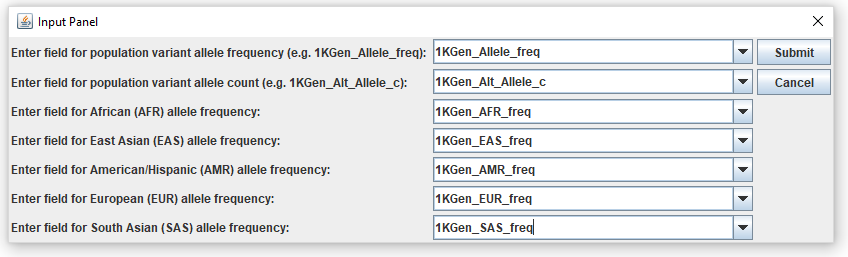


**Step 7. Choose whether to use 1000 genomes for allele frequencies or use your own dataset.**

The Salvage Pathway uses allele frequencies to calculate prior probabilities for regenotyping a variant. The user could either choose to use the default data from 1000 genomes or they can choose to use another dataset for allele frequencies.



If you choose to use your own dataset, the inputs for the allele frequencies look like this when using 1000 genomes.

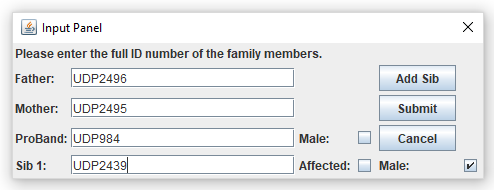


**Step 8. Input Pedigree.**

The user will then be prompted to enter the ID’s of the family members they wish to analyze. Any siblings beyond the proband who are also male and/or affected can also be specified by clicking the “Add Sib” button and clicking to check the appropriate boxes. If the input value is not included in the header of the input file, an error message will appear.

Please input the ID number **without** the .NA suffix. For example, if the ID number is “UDP234.NA” in the input file, enter “UDP234” - do not use an underscore or any change from how the value appears in the lower left-hand box of the VarSifter GUI. Any entry that is not an exact match to some individual will generate an error message when the enter key is pushed, or if you try to mouse to the next prompt. This allows for correction of the mismatched person before the submit button is pushed.

If the “Male” and “Affected” boxes are not checked, the default values are unaffected and female respectively. Remember: phenotype is everything and the analysis fails if affected status is wrong!

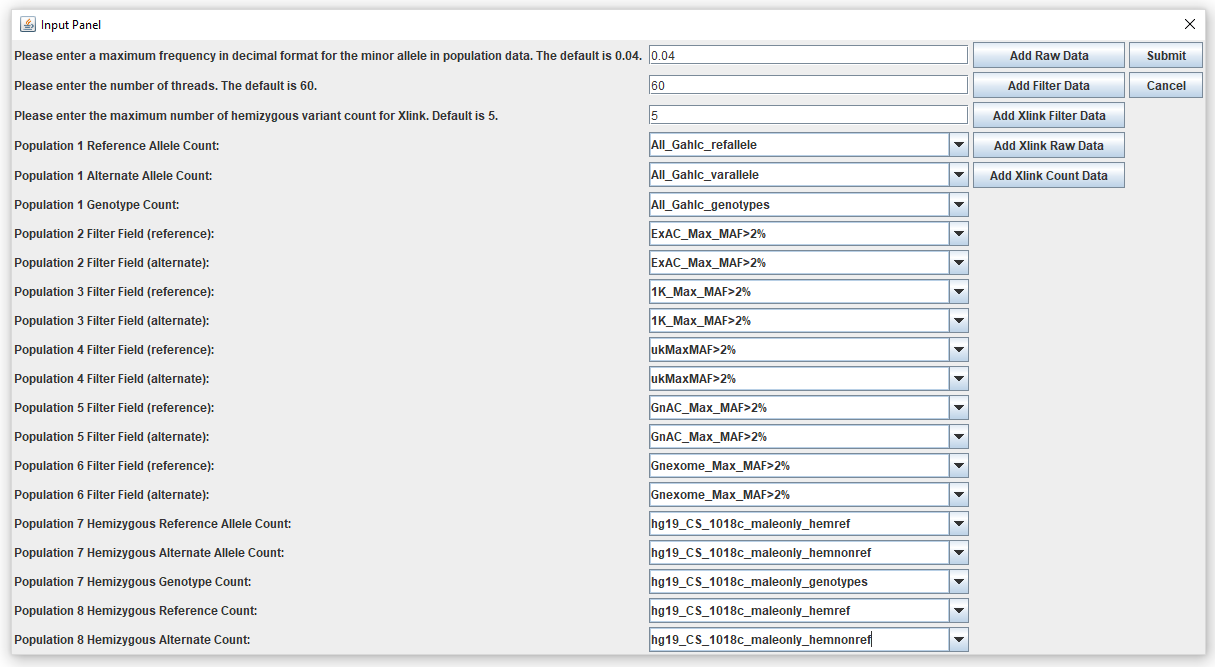


**Step 9. Input population statistics.**

If the proband is not male, X-linked variant analysis is not performed. In that case the following message will appear:

Put message of female population data

If the proband is male, X-linked variant analysis is performed and extra buttons and fields will appear for X-linked related input.

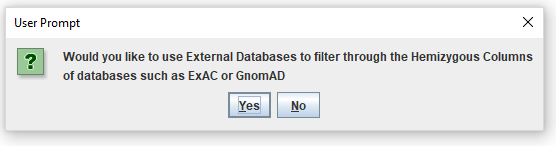


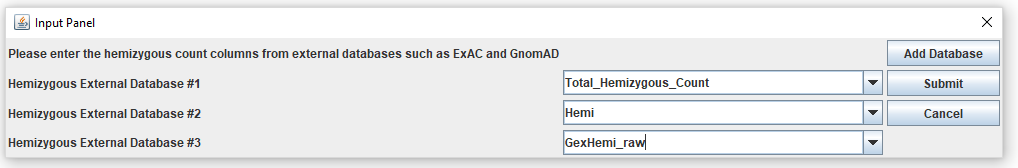
* Please enter a maximum frequency in decimal format for the minor allele in population data. This threshold is used as the cutoff threshold for the Poisson calculation using the added raw data. “0.04” is normally used because in All\_Gahl cohort, genotype counts instead of allele counts are used (0.04 is effectively a 2% allele frequency). A quarter of the entered value (“0.01” for “0.04”) is used for X-linked raw data Poisson calculation because for X-linked variants, the chromosome count is the allele count.
* Please enter the maximum number of hemizygous counts for X-linked; the entered value will be used as the rejection threshold for the X-linked Count Data fields. For example, if the cutoff is 5 counts (meaning 5 well sequenced individuals with MPG Score>10), any variant with more than 5 counts of hemizygous variant is rejected.
* Add Raw Data: the raw data include refallele counts, varallele counts, and genotype counts. Note genotype is not total allele counts, but chromosome counts.
  + Header examples: “All\_Gahlc\_refallele” & “All\_Gahlc\_varallele” & “All\_Gahlc\_genotypes”)
* Add Filter Data: filter data should only include a true or false value. The program only lets fields with “false” value be printed out. These fields represent that the maximum minor allele frequency is clearly 95% confidently less than 2% or 1% (whichever the user chooses). If MAF is not calculated for ref and var alleles separately, put the MAF for var for ref as well.
  + Header examples: “ExAC\_Max\_MAF>2%” or “ExAC\_Max\_MAF\_var>2%”.
* Add X-linked Filter Data: X-linked filter data are also exclusively either a true or false value. And “true” values are thrown out. After the “HemMAF” program is run to complete the hemizygous count annotation, ExAc, Hg(ClinSeq) both have X-linked filter data.
  + Header examples: “ExAC\_Max\_hemMAF\_ref>1%” & “ExAC\_Max\_hemMAF\_var>1%”
* Add X-linked Raw Data: the raw data are used to calculate for Poisson distribution with 95% confidence. After “HemMAF: program which completes the hemizygous count annotation, All\_Gahl and Hg(ClinSeq) both have X-linked raw data.
  + Header examples: “All\_Gahlc\_hemref” & “All\_Gahlc\_hemvar” & “All\_Gahlc\_hem”
* Add X-linked Count Data: this fields requires hemizygous reference count and hemizygous variant count. After MAFixer annotation, both All\_Gahl and Hg(ClinSeq) have those fields. (This often use the same columns as the X-linked Raw Data.) This set of data doesn’t need the All\_Gahlc\_hem column because it is not used to calculate Poisson distribution for MAF.
  + Header examples: “All\_Gahlc\_hemref” & “All\_Gahlc\_hemvar”

Above examples use the UDP2 internal (or “All\_Gahl”) cohort population data or the ExAC data3, but it is possible to add and use data from other population cohorts to the filter to increase the filtering power. Once the user is familiar with those additional columns of data, they can start to use those columns too. Cohorts like 1000 Genomes4, UK10K5, and gnomAD are examples of such cohorts that can be added to the VarSifter file.

**Step 10. X-linked External Database Check.**

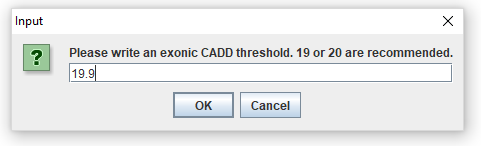
If the proband is male, then the user would be prompted if they would like to use External Databases to filter X-linked diseases based on the maximum number of hemizygous count cutoff inputted in the previous step.



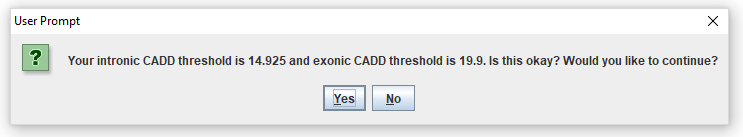


**Step 11. Input deleteriousness scoring threshold.**

The user will be prompted to enter the desired exonic CADD threshold and click “OK” to continue or “Cancel” to cancel the job. Any variants with a CADD Phred score below this value will be excluded in both outputs. While the user may put in any number value (integer or double), a threshold of between 19-20 is recommended.



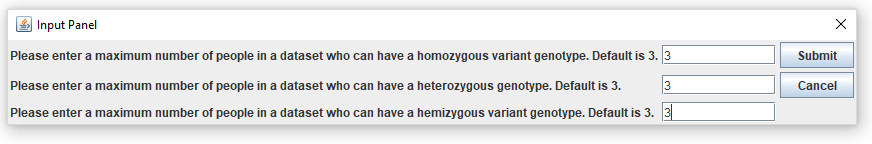
The program will then ask the user to confirm that the user entered the exonic and subsequently calculated intronic CADD.



Clicking “Yes” will continue the job. Clicking “No” will bring back the earlier prompt asking the user to input the desired exonic CADD threshold.

**Step 12. Input minimum cutoff counts for specific genotypes.**

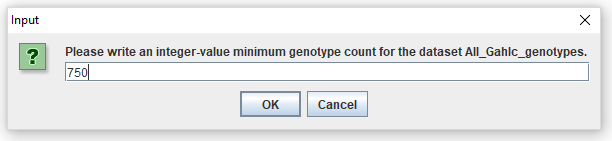
The user will be prompted to enter the minimum cutoff counts for homozygous variant, heterozygous, and hemizygous. The default cutoff values are 3 for each genotype.

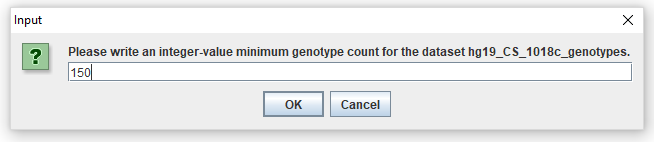


**Step 13. Input minimum genotype counts.**

The user will be prompted to enter the genotype cutoff thresholds for each of the genotype headers found in the ROC Config File. This number will vary much depending on the reliability of the source of the genotypes column. A lower number will accept more through the filter for those genotypes, while a higher number will restrict the variant to be allowed through the filter if more people were sequenced at that location.

In our flat file for example we had All\_Gahlc\_genotypes and hg19\_CS\_1018c\_genotypes. These numbers were created based on our knowledge of the reliability of each of the genotype columns and how much we wanted to filter them by.





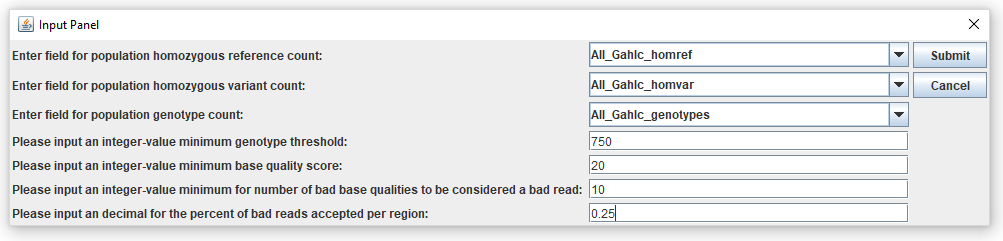
**Step 13. Input Pedigree Aware, Multi-Parametric BAM File Noise Evaluator (Confetti Filter) Parameters.**

The user will be prompted to enter the fields for the number of occurrences in which the genotype statistics data shows the number of homozygous reference counts, homozygous variant counts, and the genotype counts. Then the user will be asked to enter the cutoff threshold for that specific genotype field.

Next the user will have to input the threshold that determines the maximum cutoff for determining if a base has a low base quality score

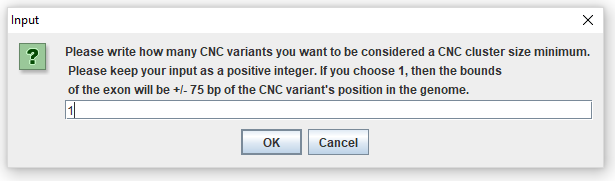
Next the user will have to input the threshold that determines the maximum cutoff for determining the number of low base quality scores that becomes considered as a bad read.

Finally, the user will have to input a decimal for the percent of bad reads accepted per region. If the region has a ratio that is more than the bad read ratio inputted, then it will reject the variant in question.



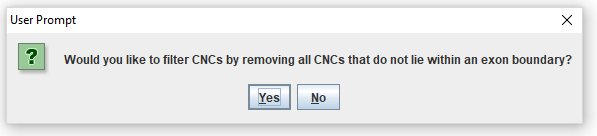
**Step 14. Input CNC Cluster Size Minimum.**

The user will be prompted to choose and enter the minimum number of variants that will be considered a CNC cluster/potential dropped exon (Fig. 5a). This value must be a positive integer. While choosing 1 is an option, it should be noted that the bounds of the potential dropped exon are set to be ±75 bp of the possible CNC variant’s position.



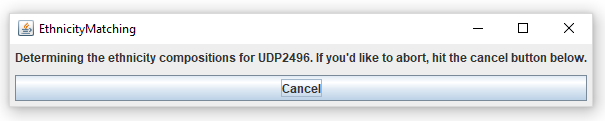
**Step 15. Choose to filter CNCs on exonic regions.**

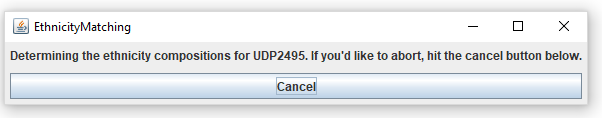
The user is prompted to choose whether they want to remove non-exonic CNC’s. This uses the Exon Boundary Config to establish the exon bounds.

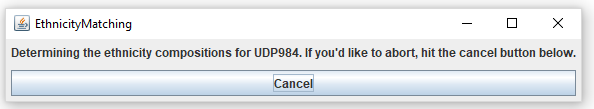


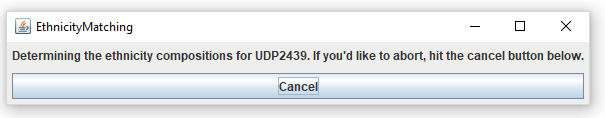
**Step 16. Program is running.**

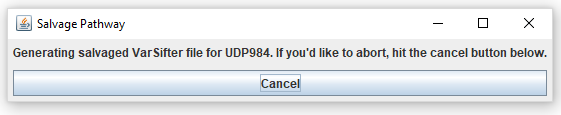
After the completion of all the inputs, the program will begin running. A dialog box will appear for each stage of the pipeline with the option to cancel the job at any point during the run. The dialog boxes changing between each stage will allow the user to have an idea of what stage of the program they are in.

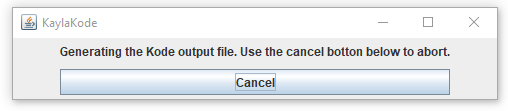


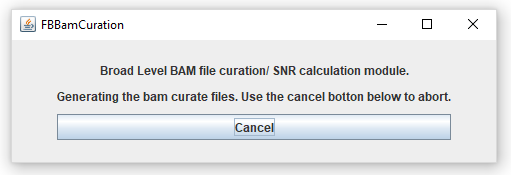


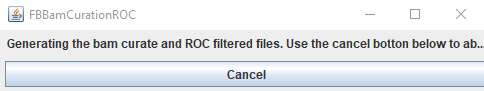


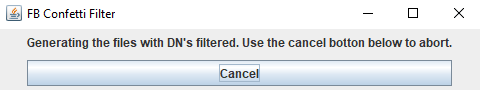


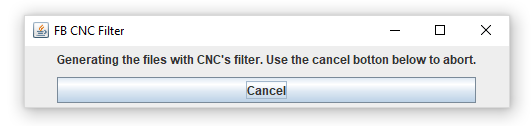






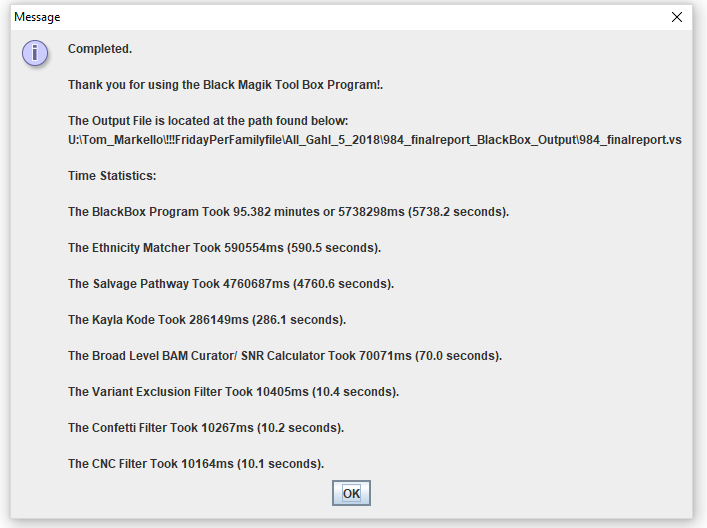






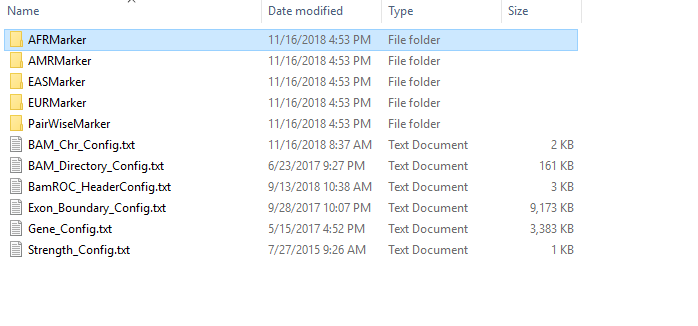
**Step 17. Program Completion.**

Once the run is complete, a dialog box will appear to confirm its completion. The completion screen will also include the path to the output file and the time statistics of the program.



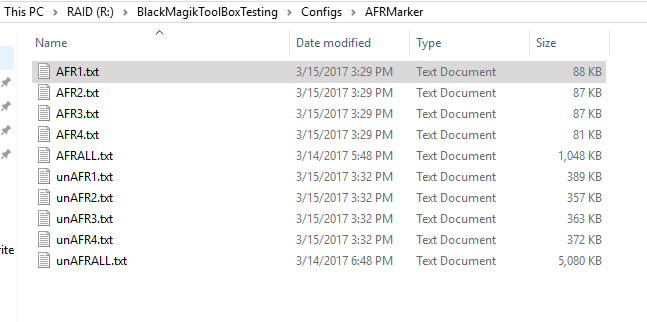
1. **Configuration Files**

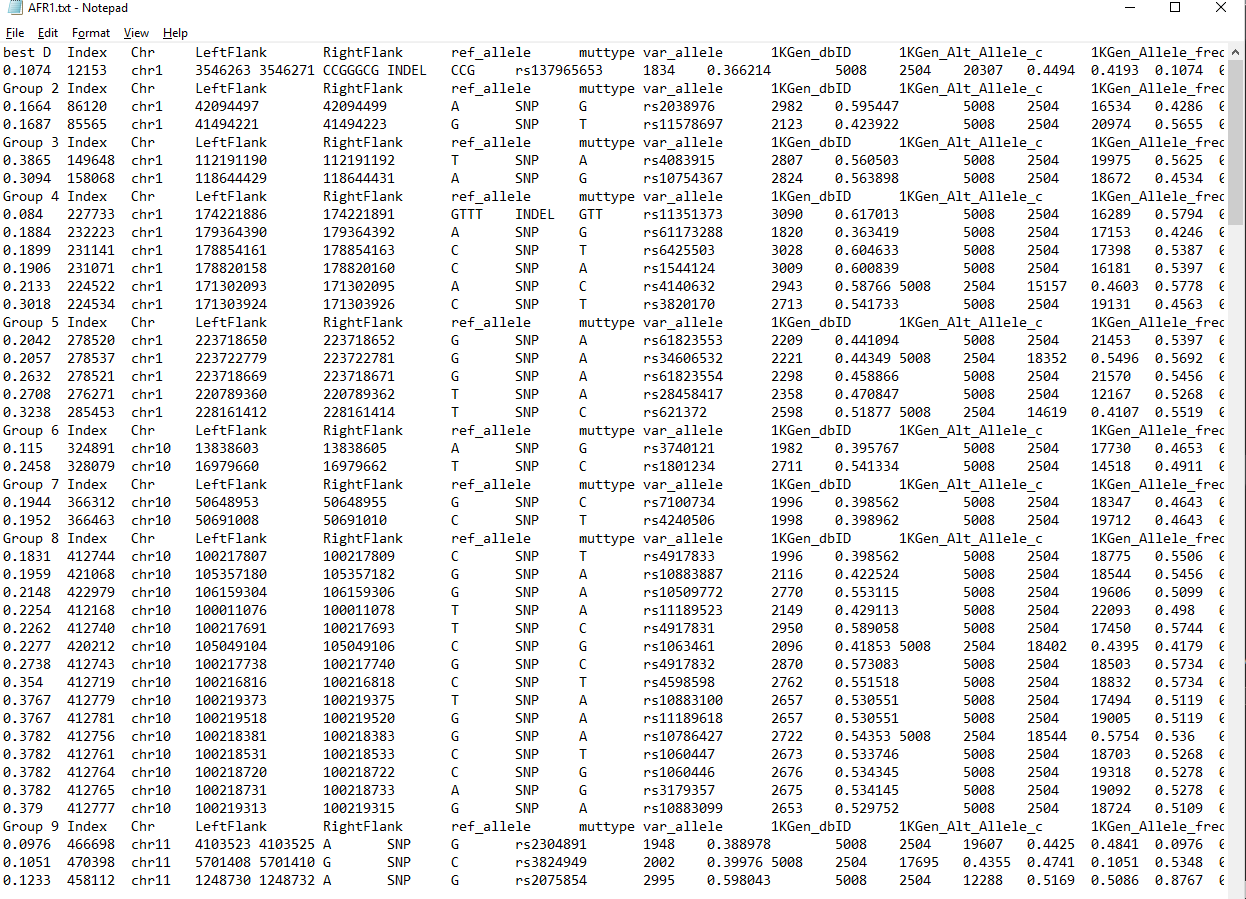
This is the entire “Configs” directory containing all of the relative configuration files for all the modules in the pipeline.

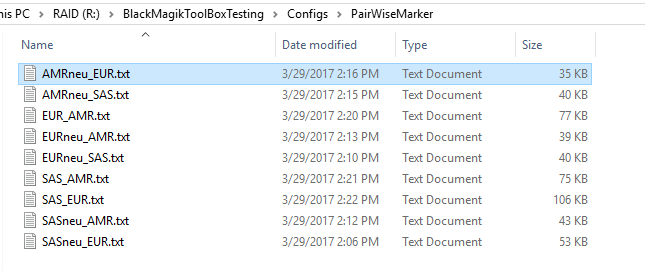


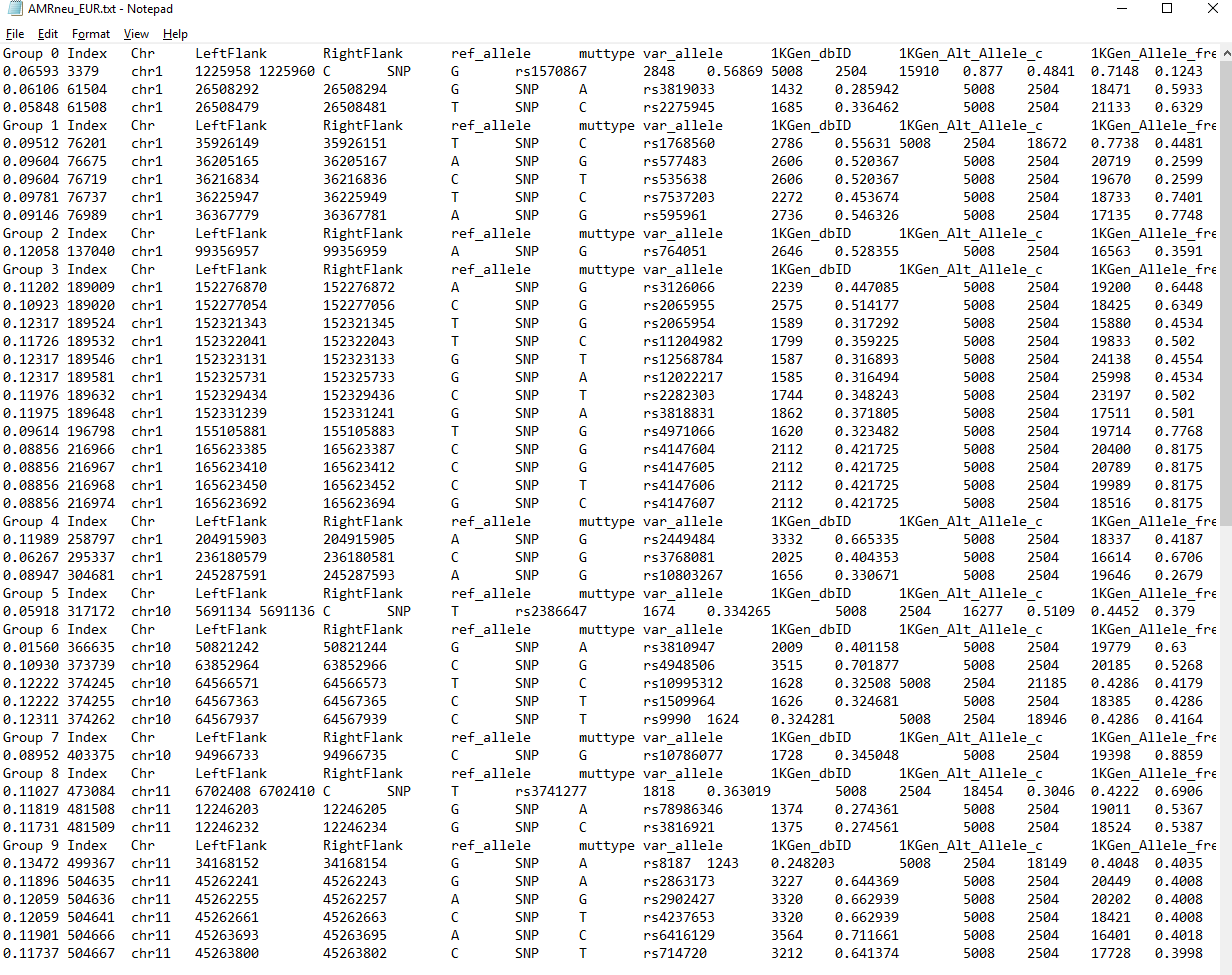
**Ethnicity Marker Configs**

For more information about the Ethnicity Marker configuration files, see the “Inputs” section of the manual.



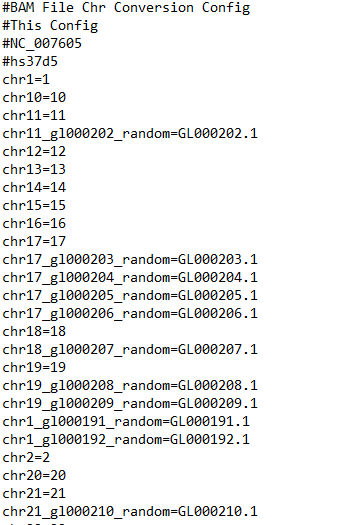






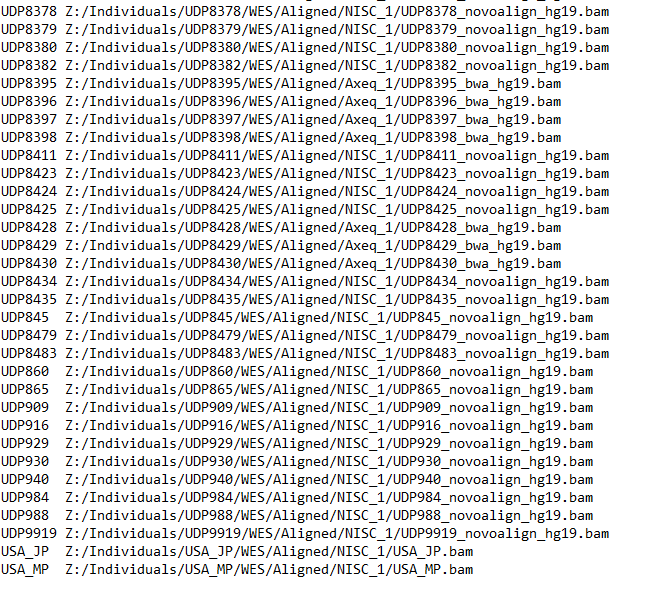
**BAM Chr Config**

This Config serves to essentially translate between the two naming conventions for chrs in the BAM files. The formatting for the config is <NISC chr format (chr1)>=<Other BAM chr format (1)>. NC\_007605 and hs37d5 currently has not been implemented where it needs to be accessed in the BAM file now, but it is mentioned for future’s sake in the config file.



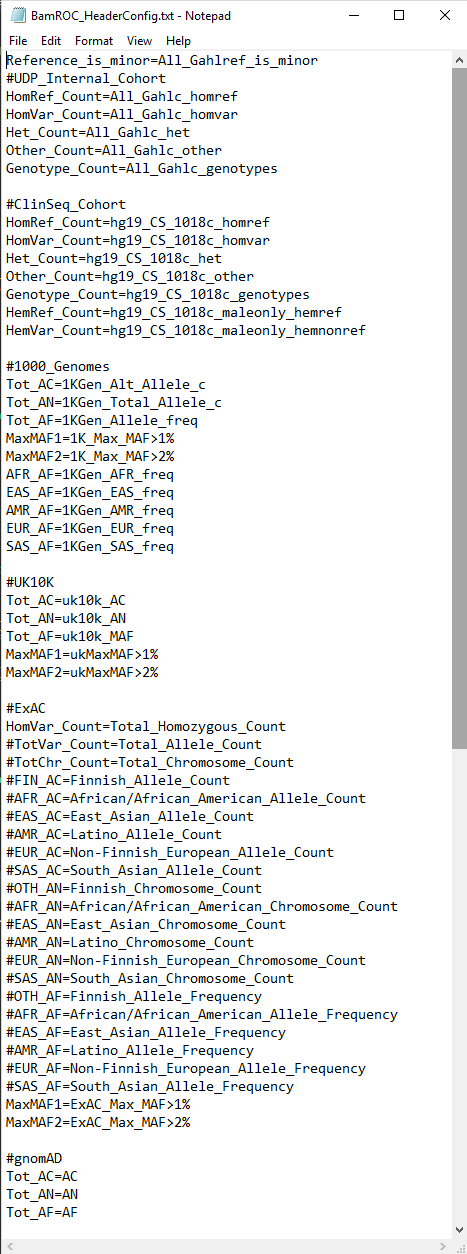
**BAM Directory Config**

This config serves to reference the BAM locations based on the IDs of the person as they appear in the VarSifter file. The formatting is “ID \t BAM File Location” (without spaces).



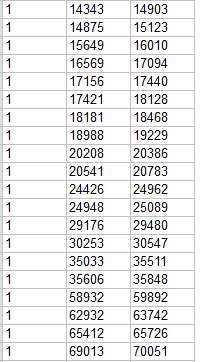
**BAM ROC Header Config**

This config file is used in the variant exclusion filter module and classifies different headers in the VarSifter file to be used for different analyses. The config file must be in the format “<category>=<VS header>”.



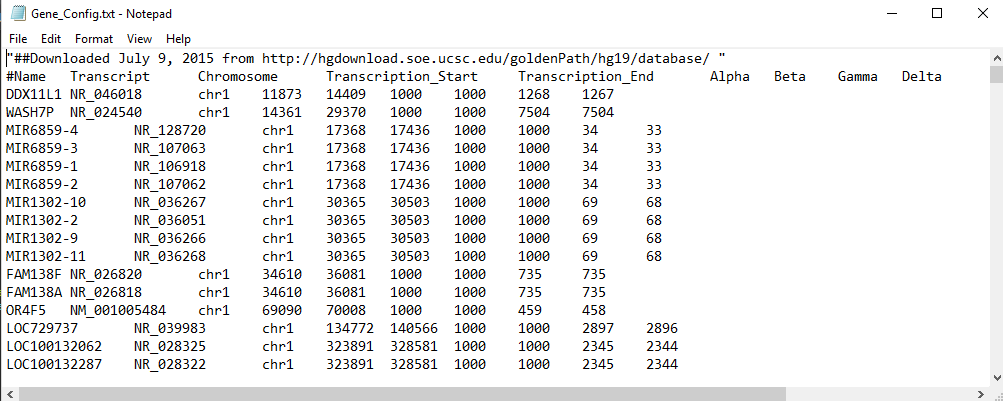
**Exon Boundary Config**

This config file contains a list of all the boundaries for exons in the human genome. This will be used in the CNC filter to filter out all non-exonic CNCs.



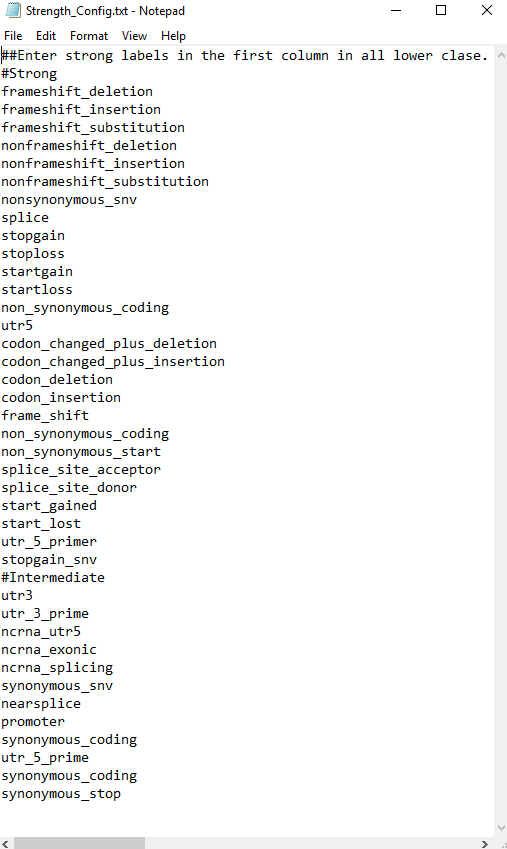
**Gene Config**

For more information about the Gene Config file, see the “inputs” section.



**Strength Config**

This is a simple config file that assigns a score for whether a variant causes a more severe change in a codon (like a stop or a nonsynonymous change) or a less severe (such as a synonymous change in a codon) or is intronic. It is used in the compound heterozygous recessive analysis.



1. **Ethnicity (Ancestry) Matching Module**

**Executive Summary**

This module seeks to define the best set of prior probabilities to resolve the apparent excess of de novo (Mendelian inconsistent) genotypes in the starting flat file. This requires the original BAM files, the annotated flat file, and a pedigree file listing the biologic relationships of the quartet being analyzed. The program rapidly produces an estimate of whether a single 1K genome ethnicity prior probability allele frequency would fit the case, or if the average of two ethnicities would be best, or if the global averaged allele frequency would be the best choice for a starting prior probability for a pedigree aware re-genotyping module run next. The only significant output of this is the designated ethnicity of the parents (for prior probability estimation only).

This module proceeds in a stepwise manner to resolve the question: Is the current individual more like an African allele pattern than any other ethnicity(Y/N)? If no then it goes on to ask is the current individual more like an East Asian allele pattern than any of the rest of the ethnicities(Y/N)? This is repeated until either a best choice is found or a mixed ethnicity (either biethnic or multiethnic) decision is reached. Once the decision is made to use a single ethnicity allele frequency, or the average allele frequency as the prior probability, this module is finished.

**Purpose**

Documentation and rationale of the Ethnicity Matching program for use in the Salvage Pathway, a Bayesian-based, nuclear family pedigree aware, re-genotype caller.

**Introduction**

Ethnicity matching is only used as a component of the Salvage Pathway (Manual 2). This program was written to rapidly and accurately determine an ethnicity (ancestry) that best represents the sample’s overall genetic background solely for choosing a more accurate prior probability for genotyping at positions in the genome where an ethnic population has a previously identified polymorphism. Instead of using a universal constant prior of 1/1000, substituting an ethnic-specific population frequency prior probability for a Bayesian-based genotype caller most likely improves the accuracy of the genotype call, especially when low coverage read depths are present. However, allele frequency priors can and often do vary widely between ethnic groups at the many genome positions where polymorphisms are present. In theory, selecting the wrong ethnicity could potentially worsen the reliability of a genotype call compared to using the average allele frequency as the prior, but would only rarely worsen the accuracy compared to using a simple constant like 1/1000. Thus, the first question to address is what ethnicity best describes a person’s genetic background and is that ethnicity significantly better than any other choice, so that using its allele frequency would be better than using the global allele frequency?

The program uses simple principle axis ancestry informative markers in linkage equilibrium in a conjoint probability estimate. For example, consider that at position X, people of African descent have an alternate variant allele frequency of 0.8 and that all other ethnicities measured at that same position have variant allele frequencies close to 0.5, like a coin toss. The other ethnicities can now be used as a joint control group for a pairwise comparison. If the person of interest’s genome has the variant allele, that person’s odds are 0.8/0.5 to be ethnically African compared to any of the other ethnicities. If we multiply the odds of a few hundred similar positions that are separated in the genome far enough to be in linkage equilibrium, and if all positions share similar property (ancestry informative markers, or AIMs with only the African allele ratio significantly different from 0.5), we can achieve a high confidence regarding whether the sample is African or not African. A logarithm of the odds ratio (LOD) of 3 or -3 is chosen as the cutoff to reject or accept each ethnicity in this process.

Our program evaluates for the following “ethnicities”:

* AFR: African American/African
* AMR: Latino/Native American
* EAS: East Asian
* EUR: European
* SAS: South Asian
* “Unknown” refers to ethnicities that do not match any of the 5 listed above significantly better than any of the other ethnicities.

This is the first stage of the workflow. The output file should be used as the ethnicity configuration folder in the Salvage Pathway

**Program Overview**

In March 2017, we mined hundreds of thousands of ancestry informative markers from 1000 Genomes and constructed a marker profile that can be used to determine ethnicity. Those markers, at the time of this document, are contained in the EthnicityMatchingSoftware folder (Supplement B).

Five folders contain five marker sets. Each is used to determine whether a certain ethnicity should be accepted or rejected over all the others.

* AFRMarker
* AMRMarker
* EASMarker
* EURMarker
* PairWiseMarker

To be more specific, AFRMarker contains 4 sets of markers where the allele frequency in the African ethnic subpopulation is deviant from 0.5, and all other ethnicities are clustered at 0.5 variant allele frequency. The AFRMarker set also contains another 4 sets of markers where the African allele frequency is clustered around 0.5, but all others are deviant away from 0.5 in the same direction. (They are all either above or below.)

In each set, there are about 60 groups of markers that are in linkage equilibrium so the log ratio can be simply added. Within each of those 60 groups, there are 7-8 markers that are close to each other and are certainly in linkage disequilibrium. Those act as substitutes to each other, and only one is used. If the sample under study is not well sequenced in the first marker of that group, the code will look for whether the next marker was well sequenced and continue until it finds a marker in that group where the sample is well sequenced. This ensures a very high likelihood every group will contribute to the global likelihood and thus give 60 independent likelihoods whose log values can be summed.

For those markers, the log ratio for a single position is calculated as:

The log ratio values for every independent marker position are added together to aggregate a high confidence call.

As another confirmation to the AFR marker set, there is a marker set called “unAFR,” which contains markers where the African is around 0.5 and the other ethnicities have an allele frequency deviant from 0.5 in the same direction (by simple selection of the major and minor allele for each ethnicity to give the same direction for selectivity). The log ratio score is calculated as:

All 8 marker sets aim to answer the question, “Is this person of African descent or not?”

Occasionally, a marker set might not have enough marker position hits where the sample is just not well sequenced in many markers in a group. Thus, the marker sets are normalized to a set number of markers, so that when the marker sets are compared together, values are not falsely skewed towards one ethnicity as an artifact of unequal marker sequencing coverage. For example, African marker sets all have their markers scaled to a fixed number 65. It is scaled within each marker set as such:

65 is roughly the number of independent markers in each marker set that are hit using a test sample.

After the 8 marker sets are calculated, the 8 log ratio scores are averaged.

In analogy, all the other ethnicities have their own 8 marker sets, so those log ratio scores can be calculated. AFR (African), EAS (East Asian), AMR (American/Latino), EUR (European), marker sets all contains 65 markers in each group, so that all sets are comparable in their probability magnitude when being evaluated against each other possibility in a fair comparison of the genotype cutoff values. SAS (South Asian) doesn’t have its marker sets, because it’s determined by the above step-wise elimination. The SAS/EUR marker comparison was left until the last step because these two allele frequency sets are most close to each other and a false choice will cause the least inaccuracy in the subsequent use of the priors as used in the next module.

The log ratio score for each ethnicity is used in the following logic tree with a log ratio score of 3 as the cutoff:



Figure 8. *Logic tree for determining ethnicity*

The first two decisions are “final” if there is not a clear decision between AFR and notAFR, or EAS and not EAS, so any in between result (not >1000:1 or <1:1000 likelihood) is immediately called unknown (= global allele frequency as the prior).

When any of the above ethnicities cannot be confidently accepted or rejected, another more nuanced comparison is evoked called “pair-wise comparison” (see below). AFR and EAS have very strong markers so they usually have strong rejection or acceptance log ratio scores. If even those scores are ambiguous, the sample is likely to be a mixed ethnicity that might contain African or East Asian. The pairwise comparison requires that both African and East Asian are already ruled out. If ethnicities cannot be returned in later steps, the program returns “Unknown”. If African or East Asian cannot be ruled out, the program designates the ethnicity “Unknown” and exits.

The second to the last decision in the main path also has a “pairwise” (double elimination) side branch that gives the three remaining options (AMR, EUR and SAS) a second chance if all three pairwise tests can agree that one is better than both of the other two (using different markers that have better power to distinguish between two groups head to head).  Since the last decision (EUR vs SAS) is in fact a simple pairwise decision, that is easiest of all the comparisons, but also the least significant choice since their allele frequencies give the least different priors over all positions.

The pair-wise comparison utilizes 10 other marker sets that compare two ethnicities at a time:

* AMRneu\_EUR (AMR around 0.5, EUR skewed)
* AMRneu\_SAS
* EURneu\_AMR
* EURneu\_SAS
* SASneu\_AMR
* SASneu\_EUR
* EUR\_AMR
* SAS\_AMR
* SAS\_EUR
* EUR\_AMR

These are all in the PairWiseMarker folder in the path documented above.

AMRneu\_EUR and EURneu\_AMR are two supporting markers. They are also scaled to a certain number of markers and averaged between them to give a log ratio score for EUR compared to AMR.

The number of markers used were collected over a test sample of some 100 people. The average number of markers used are employed as the scale value. The scale function is the same as the above scale function for AFR markers.

All 6 sets are scaled to be equivalent to 35 independent marker groupings so they can be directly compared to every other set in a fair comparison of the conjoint groupings compound likelihood.

AMRneu\_EUR and EURneu\_AMR are averaged together, AMRneu\_SAS and SASneu\_AMR are averaged together, EURneu\_SAS and SASneu\_EUR are averaged together.

EURneu\_SAS, AMRneu\_EUR and SASneu\_AMR are triangular scores for one set of pairwise comparisons among the three ethnicities.

EURvSAS, AMRvEUR, SASvAMR are another set of triangular scores for another set of pairwise comparisons among the three ethnicities.

The results from the two sets can be compared to confirm each other.

The logic goes:

if(AMRneuSASscore>3 && AMRneuEURscore>3) {

candidate="AMR";

}

else if(EURneuSASscore>3 && (-AMRneuEURscore)>3) {

candidate="EUR";

}

else if((-EURneuSASscore)>3 &&(-AMRneuSASscore)>3) {

candidate="SAS";

}

Then similarly for another set:

if((-SASvAMR)>3 && (-EURvAMR)>3) {

candidate2="AMR";

}

if((-SASvEUR)>3 && EURvAMR>3) {

candidate2="EUR";

}

if(SASvEUR>3&& SASvAMR<3) {

candidate2="SAS";

}

Then candidate and candidate2 are compared; if they agree, the result is returned. Otherwise, “Unknown” is returned and the average population frequency over all ethnicities are used as the prior population frequency.

*Why choose 0.5 as the control marker frequency and not extreme difference?*

Extreme difference AIMs are those whose allele frequencies are very high in one ethnicity only but can have varying allele frequencies in other ethnicities that are not equivalent. We did not use extreme difference AIMS because they tend to give values that can be circularly contradictory in ethnicity predictions. Worse still they are fewer in number, not found uniformly across the genome and are not equal in number between groups. They also frequently vary even within an ethnic group, being specific for only a sub-ethnic grouping. Using 0.5 as the control marker frequency controls the log ratio score for individual SNPs to a similar scale across all markers and these can be found evenly distributed in equal numbers for every ethnicity across the entire genome space. In fact, multiple sets of markers spaced in linkage equilibrium can be found to form cross validation sets of AIMs. This strategy significantly lessens the risk that a small quantity of SNPs does not unfairly bias the ethnic prediction.

1. **Salvage Pathway Module (Bayesian genotype calling from pedigree information, ethnicity matching, and sequence reads)**

**Executive Summary**

This module evaluates one specific family one offspring at a time at every place one of the offspring has an apparent Mendelian inconsistency in the jointly called flat file. The program resolves the inconsistency by regenotyping using the population prior probability, a chosen mutation rate, and the inheritance information of the quartet in a Mendelian inheritance prior, along with all the reads in every family member's BAM files to produce a better genotype call at these locations. It does not address the tens to hundreds of thousands of places called nonreference in a typical quartet jointly called variant file. The program uses a straightforward classic small pedigree Bayesian caller using an ethnically matched prior at each position and the inheritance conditionals along with a mutation rate conditional probability plus the information in the combined short read sequences in the BAM files of all four family members at that location.

There are two parts to the module. One involves making a "perfamily" variant file list that looks at the global density of variant base calls in all the reads of the entire family's BAM files. This involves taking each read in all four individuals and binning any variant base calls as a function of how far the positions are on either side of the position being genotyped, up to 150bp on either side. This parameter weights the value of each variant base less as it is found further from the position being re-genotyped. This process is only done once for both offspring if at least one offspring and both parents have some reads at that position.

The second half of this module looks at each sibling one at a time, and using the siblings’ conditional probabilities for their own reads, regenotypes the parents and the sibling, testing if the apparent de novo (Mendelian inconsistent) genotype calls are still the most likely, or if in the light of the known Mendelian inheritance relationships the most likely genotype call should be changed to a simple inheritance of a variant from one parent to the offspring being re-genotyped at that location. This is repeated for all apparent Mendelian inconsistent locations for that sibling and then the entire process repeated for the other sibling(s). When this process has been completed for all positions in both siblings, the old and new genotype designations are preserved in the output flat file. Old designations (genotypes) are re-inspected by the Mendelian modeling code (KaylaKode) when testing for de novo variant candidates, and the new designated genotypes are used in the Mendelian modeling code (KaylaKode) when testing for compound heterozygous recessive candidate pairs.

**Purpose**

The Salvage Pathway is a second pass genotyper that attempts to analyze variant calls at the BAM level using automated autonomous code. Although the goal is to improve genotyping scores at mostly noisy and low coverage regions, it has developed into a more general utility, able to function as a genotyper that is pedigree aware and reinterprets the BAM file data. This is used in Ethnicity Matching (typing) of parents, and BAM curation in the Forwards/Backwards project. This re-genotyper, when using the Ethnicity Matching for Bayesian prior estimates, improves calling in both good and bad regions of BAM file pileups.

**Introduction**

Multiple factors challenge accurate variant calling at low-coverage regions, often rendering several thousand variants that are falsely Mendelian-inconsistent. Within those Mendelian-inconsistent variants could be a true positive compound heterozygous half pair, so the most common current approaches that discard low coverage genotype calls using variant call file quality measures like VQSR, may lose the potential to find true positives in these large collections of false positive apparent de novo genotypings. Therefore, to facilitate more complete agnostic analysis, we propose a variant calling approach that, as much as the actual coverage can allow, corrects false positives in Mendelian-inconsistent spots. This method incorporates further pedigree information and ethnicity matching to call the variants in low-coverage regions with higher confidence than available variant calling tools that do not use this additional information.

There are many factors that bias variant calling at low-coverage spots. These begin with basic sequencing and mapping errors. Mapping errors are partly due to non-exhaustive alignment methods, and partly because the reference sequence used in alignment cannot be perfectly representative of all samples sequenced, especially in the highly polymorphic regions of the genome. In fact, low complexity and highly repetitive regions of the reference genome produce regions with high misalignment content, at least for all data aligned to a single monolithic reference genome. We use the term “confetti” regions because those regions have abundant and widely spread-out mismatches that resemble colored confetti dropped onto the grey background of the perfectly matched (reference genotypes called) bases in the short-read pileup when viewed by the Broad Integrated Genome Viewer in the default “show mismatched bases” format. An additional factor that biases some genotype calling happens when genotype callers use an averaged genome frequency prior probability of 1000:1, instead of the more accurate and much higher frequency at known polymorphic sites in the genome. This risks a false negative heterozygous genotype call at positions with low depth of sequencing coverage. Our approach focuses on minimizing these three factors, coupled with pedigree information from parent-parent-child trios, to improve variant calls from genotype callers that evaluate single exomes and use flat error functions and flat priors for frequencies. We do this by estimating the ethnic-specific prior probability for allele frequencies adjusted for each position in the genome and using published ethnicity frequency priors. We also adjust the error estimate across the genome, and finally derive the genotype call as the most parsimonious explanation of the inheritance state among the nuclear family trio.

**Program overview**

Prior to the use of these salvage tools, the genome sequence data are presumed to have been aligned and genotype called with a standard genotype caller. These genotypers are not pedigree aware and use only the local reads for an individual in the genotype calling. Some use a flat prior, some use user defined priors and some calculate priors from a grouped average of BAMs to be genotyped in a single run. Mendelian inconsistent calls from a pedigree naïve genotype caller for a parent-parent-child trio can be processed further by this method to “salvage” the best genotype call that may agree with, or be a revision of, the first caller’s genotype for one or more person in the trio using the Mendelian relationships and the conjoint reads in all three pileups.

**Step 1: Determine the prior probability with ethnicity matching (parents only):**

This step uses robust and reliable ethnicity informative markers in the sequenced data to decide the most appropriate ethnic population frequency, if the ethnicity can be confidently determined.

If there isn’t ethnicity confidence for the parent, the default uses the 1/1000 genome global population frequency. For the places not covered by population frequency, this method uses the unassuming population prior: 50% for reference, 50% for alternate allele. This is because there has already been a pedigree-unaware genotype call in one of the three individuals in the trio that notes the presence of a heterozygous state.

A subroutine scans the whole file to extract ethnicity information. It then reads the config file of population frequency downloaded from the 1000 Genomes website1.

Ethnicity determination is described in the Ethnicity Matcher Manual 1.

*Denotation*:

A is reference, B is variant. AA denotes homozygous reference. From population frequency, we extract P(A) and P(B) population allele frequency for the determined ethnicity, and perform the following calculation:

Assuming Hardy-Weinberg Equilibrium:

(1)

**Step 2: Calculate the conditional probability with BAM file:**

Then using the sequence information in the individual bam file, we calculate the likelihood of seeing those reads given a certain genotype. If we assume that errors are independent, a simple Ehrenfest model would be:

(2)

Where D is observed reads with total n reads and k errors. L(D|AA) denotes the likelihood of observing D reads if the genotype is assumed to be homozygous reference AA. is the probability that the read does not come from the position mapped.

However, neither mapping error nor sequencing error is independent, so these errors have codependent crossterms. When a read is seen to be mis-mapped to a position, it becomes likely to observe other reads mis-mapped to the same position in subsequent or prior reads that map to the region and overlap that position. The details of how the dependency is modeled is borrowed from Heng Li’s approach2. An alternative approach utilizes a heuristic estimate that reflects the dependencies between errors.

**Step 3: The posterior probabilities are calculated as follows for the parents:**

Putting the first two steps together to calculate the Bayesian posterior probability.

(3)

P(AA|D) denotes the probability that the genotype is AA given the observed sequencing reads.

**Step 4: The parents’ information is used to calculate the child’s prior probabilities:**

Parents’ genotype probability vector (denoted as G) is used as the prior for the child’s genotype estimates.

corresponds to the genotype probability of AA, P(AA), for the mother.

(4)

is the Mendelian coefficient given for the child to be AA ()

is the mutation coefficient. is 2e-7.

is a binary function. It returns 1 when the statement is true.

is the mutation variable. It is one when the child’s genotype requires mutation from the allele from the mother.

Mendelian coefficient is the Mendelian inheritance probability. For example, if the parents are both homozygous reference, the Mendelian inheritance probability for the child to be homozygous reference is or ~1.

**Step 5: Recursive genotype calling on the parents (does not include hemizygosity for the X chromosome):**

Similarly, the parent and the child’s genotypes can be used to calculate genotype probabilities of the other parent. In summary, parents’ genotype probability vectors are calculated with equations (1) to (3). Then the child’s genotype probability vector is calculated with equation (4). Taking the most probable genotype from the parents and the child, if they are still Mendelian inconsistent, the child and one parent (usually the one with the highest read depth) are used to repeat equation (4) to re-evaluate the genotype probability vector of the other parent. The same procedure is performed on the other parent. If the most probable genotypes of the trio are still in Mendelian inconsistent state, we consider the trio genotype calls to be truly Mendelian inconsistent.

Lastly, a new genotype score is calculated with the probability of the most probable genotype in a Phred scale. This results in an output file, along with a report file that summarizes all the positions changed, and the genotype probability vectors for each person in the trio.

**The workflow of the Salvage Pathway perfamily file maker:**

* Unindicated in the workflow diagram is the fact that the perfamily considers the minor condition where the reference genotype call is actually the minor allele, and can be a candidate for being a deleterious cause of a recessive disease. A symmetrical image would show this alternative.
* Unindicated in the workflow: If the population frequency is listed as 0%, it is fixed at 1/15000 as an empirical estimate of a minimum potential allele frequency. This default is chosen since there is always a non-zero probability that an allele is present in a non-exhaustive population allele sample, even when 100 of thousands of individuals are sampled. This lower limit is conservatively “high” to avoid a prior Bayesian starting point that cannot be overcome by a deep read depth of the conditional probability achieved by the experimental sequencing data.





\*\* Variants that are originally determined to be de novo are also printed in the perfamily file even if the algorithm determines that they are no longer de novo. This is part of our effort to be as conservative as possible, and is especially aimed to rectify some of the false negative salvaged spots caused by extreme allele skews. These original de novos will be evaluated later for noise using the Pedigree Aware, Multiparametric BAM File Noise Evaluator (see Manual 6).

**Detailed workflow:**

1. *SalvagePipeline method*

This method is the overall workflow of the salvage pathway. It takes in the input VarSifter file, the family pedigree, the BAM file locations of the pedigree’s members, the output perfamily file path, and the ethnicity of the parents. The whole module outputs a VarSifter file that includes re-genotyped variants that were previously Mendelian inconsistent. It also generates a report file that documents the positions that were re-genotyped and some other details of the salvage process.

1. *FindBAM method*

The FindBAM method uses hash mapping of each sample’s BAM file paths to allow a quick reference to different individuals’ BAM files.

1. *SNR (Signal-to-Noise Ratio) method*

The Signal-to-Noise Ratio (SNR), described in Manual 4, is the core method of collecting characteristics of a region within a person’s BAM file. It also assigns an error term and collects variants mapped under the regions of interest. The purpose of this method is to look at the region ±150bp of the variant position of interest(VOI) and evaluate the distribution of mismatched variants in the region.

The program first reads every read mapped within that 300bp region, and then finds the mismatches within each read by looking at their respective MD tags and CIGAR strings. Then it indexes those mismatches by their reference genome positions. At the end, it generates a hash map where the position of each mismatch is mapped to the number of mismatches at that position.

1. *Callgenotype method*

The callgenotype method takes in the reads, reference allele, alternate allele, variant frequency, and BAM file region’s error rate and returns the probabilities for each genotype. Potential hemizygous genotypes are assigned a higher mutation rate to more highly weight hemizygous candidates than simple heterozygous genotypes weights. Hemizygous genotypes are reported twice as frequently as simple de novo calls in this weighting scale, to avoid losing potentially true positive hemizygous candidates. To date, the small number of hemizygous genotype calls in the UDP cases that have been discovered have all been captured using this prioritization. Another nuance is that total read depth is scaled to maximum of 50 reads. This is because the inherent properties of an independent Bayesian Ehrenfest model are not as functional as a model, when used at higher depth of coverage values. This type of Bayesian calculation presumes complete independence of each read in the pileup, which is not strictly true for real world NextGen sequencing data and is less functionally true as read depth increases. Scaling back total read depth to allow the genotyper to stay within the range where it is a better approximation of the real data is one way to avoid this risk for inaccurate genotype calling.

**10) Example Results**

The Salvage Pathway program was run for one test trio. UDP984 (proband), UDP2495 (father), UDP2496 (mother) give a practical example of the function of this process. For this trio, there are ~51,000 Mendelian inconsistent positions (chrY and chrX are excluded) out of a total of 3,172,855 positions. The Mendelian inconsistent positions in this example do not consider sibling information.

* ~250 positions have CADD3 scores equal to or greater than 20.
* ~38,000 (~72.5%) are covered by 1000 Genome (all with the same number of samples successfully reported by 1000 Genomes).
* 4,156 are covered by ExAc4 with the number of chromosomes genotyped in their cohort ≥80.
* 4,169 are covered with ExAc with at least 1 chromosome in their cohort genotyped.

Using default population reference allele frequency 0.1%, the re-genotyping results are as follows:

**Table 1.** Results of re-genotyping (salvage) of the UDP984 family.

|  |  |
| --- | --- |
| **Salvage Status** | **Number of alleles (%)** |
| Salvaged alleles | 50744 (98.8) |
| Unsalvaged alleles | 636 (1.2) |

**Table 2.** Distribution of inheritance models in identified Mendelian inconsistent positions in the UDP984 family.

|  |  |
| --- | --- |
| **Inheritance model** | **Number identified (%)** |
| Compound Heterozygous Recessive | 13360 (26.0) |
| De Novo | 524 (10.2) |
| Positions in areas with high noise density | 1018 (2.0) |
| TOTAL | 51380 |

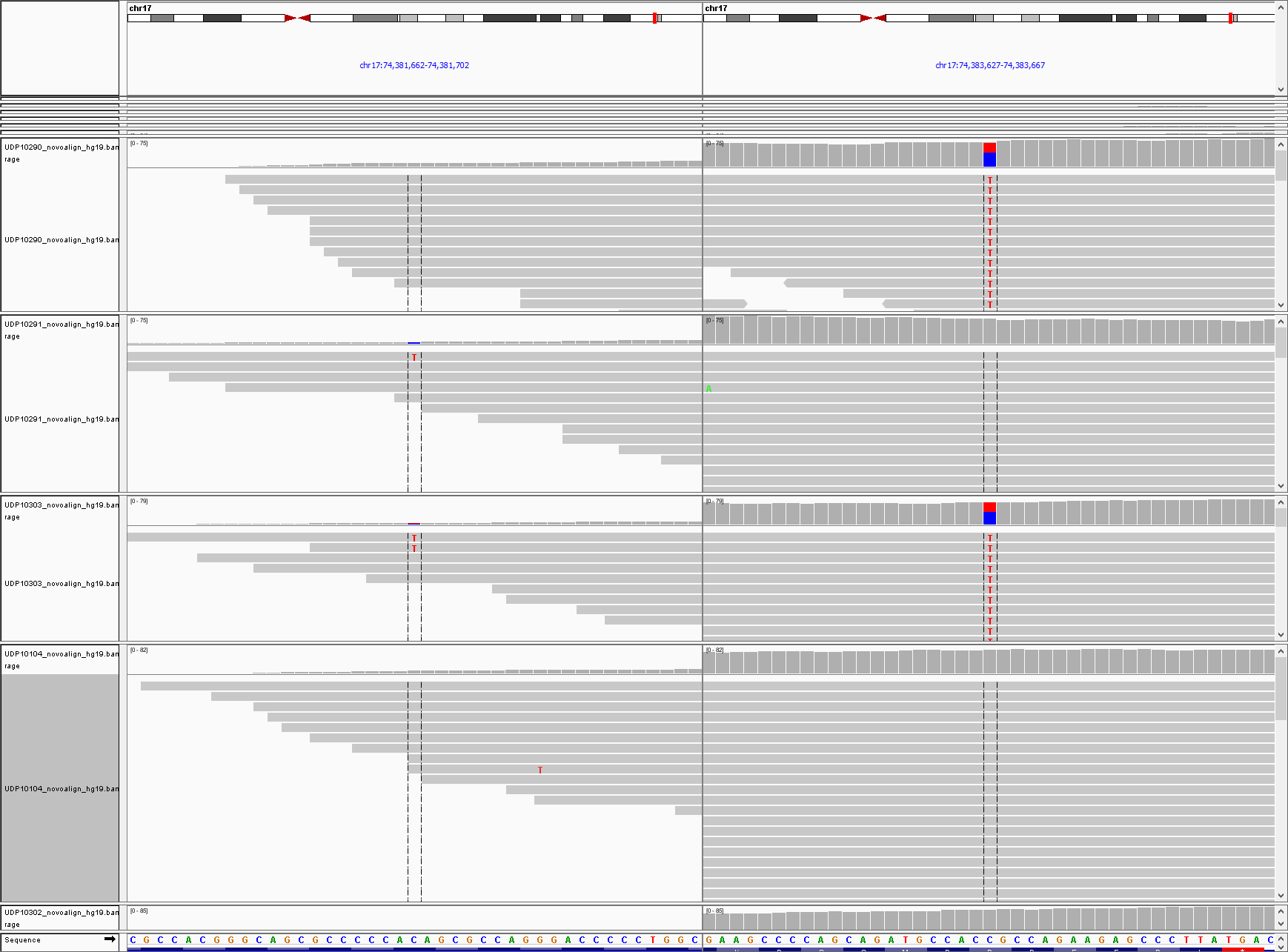
Among all the salvaged compound heterozygous recessive variants, 70 have CADD Phred scores ≥20.

Among all the remaining re-genotyped apparent de novos, 5 variants have CADD≥20.

After using the Salvage Pathway to process nine other families in the UDP internal cohort5, the rate of salvaged/re-called genotypes compared to the total number of Mendelian inconsistent variants in a family’s exome appears fairly consistent and high (Table 3). The average salvage rate is 91.4%.

**Table 3.** Comparison of salvaged alleles vs. unsalvaged alleles for 9 families in the UDP internal cohort.

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Salvage status** | **Family ID** | | | | | | | | |
| UDP10352 | UDP4245 | UDP909 | UDP2752 | UDP1869 | UDP5243 | UDP6870 | UDP6595 | UDP865 |
| Salvaged alleles (%) | 48948 (92.4) | 25749 (91.3) | 24449 (91.1) | 29029 (91.1) | 28091 (91.0) | 30708 (91.8) | 28829 (91.7) | 26654 (90.8) | 26531 (91.6) |
| Unsalvaged alleles ($) | 4028 (7.6) | 2443 (8.7) | 2401 (8.9) | 2820 (8.9) | 2777 (9.0) | 2738 (8.2) | 2611 (8.3) | 2710 (9.2) | 2404 (8.4) |



**Figure 8.** *An example of a salvaged candidate position that could be real and form a properly phased compound heterozygous recessive pairing.*

1. **KaylaKode Module**

**Executive Summary**

This module evaluates the flat file using Mendelian models, the inheritance state of the genotypes in a nuclear pedigree, the population frequencies and the predictions of deleteriousness, which are mostly based on CADD scores, but also retains known pathologic variants from HGMD, ClinVar, and other scoring systems, even if they have a low CADD score. The module employs each common Mendelian model (homozygous recessive, de novo dominant, X-linked, compound heterozygous recessive) and two special evaluations. One involves apparent total deletions and another addresses Mendelian apparent inconsistent homozygous variants, i.e., possible hemizygous compound heterozygous recessive pairing of a paternal variant in a heterozygous parent and a deletion from a hemizygous parent whose other allele is reference.

This program adds new designations in the metadata section of the flat file, specific to the family being analyzed and only for the individual being analyzed. It flags any rows (any single variant in the unrolled flat file) as being a legitimate candidate for any particular type of Mendelian model to explain the affected status of the individual. The new column in the flat file contains the VMM score for the most deleterious pairing for any variant that can be a viable half of a compound heterozygous recessive pair of variants present in the same genetic locus, to the most deleterious other half of the pair.

The compound heterozygous analysis requires a configuration file for the starting and stopping positions for any single locus that bounds the positions in which pairing for compound heterozygous recessive pairing can be made. Loci boundaries were determined by genomic position values 1000bp upstream and downstream of coding transcripts, but allowing for a single gap within each locus for a second locus of a different gene on the same or opposite strand. Initially this was run heuristically for several hundred cases. Each time a pair was made from two adjoining different genes this was flagged and the locus boundaries reset to resolve those incorrect pairings. After several months, very few illegitimate cross boundary pairs were made and the boundaries considered adequate for utilitarian use in analysis. The locus boundaries are included in the code as a configuration file that can be improved from this current build. All viable “half of a pair” variants inside the gene locus were tested to see if they could be combined with another candidate anywhere in the locus region with proper phasing by transmission from parental genotypes. Phasing was performed by the boolean XOR for heterozygosity of a variant genotype in one but not both parents. The possibility of a parental crossover within a locus that would be an exception to the XOR boolean relationship is an extremely unlikely event and has been ignored in the current analysis.

In addition to the configuration file for the locus boundaries, there is an additional "strength file" that allows simple scoring of pairs of variants in the compound heterozygous pairing based on types of variants as interpreted by the transcript annotation. This is not used to filter variants but has been found to be useful to human interpreters who are reading the final lists. These scores are weighted to give higher strength to a pair of nonsynonymous or nonsense variants over pairs where one or both variants are in noncoding regions of the locus.

**Introduction**

Variations within the genome contribute to evolutionary adaptability in response to changing selective environmental pressures. While these variations are beneficial to the population in an evolutionary sense, they make interpretation of Next Generation (NextGen) sequencing data very difficult for truly deleterious changes that can be causing a genetic medical disorder in a single individual. This confusion is even harder when there is only a very small dataset, rather than the large multigenerational pedigrees used in traditional plant, animal, and human linkage experiments by classical geneticists in the era prior to NextGen sequencing. The current discussion is meant to specifically apply to single families with only one or a few affected individuals, and a nuclear pedigree of parents and siblings.

An important issue that is beneficial to life, but complicates genetic data analysis, was the evolution of a very compact genetic code with multiple repetitions of the same sequence in separate and re-arrangeable modules. This is enormously complicated by the development of the diploid genome that contains two nearly but not exactly identical sets of genomic information in eukaryotic cells. This state of (almost identical but dual) information content is not well modeled by other fields of computational science that have studied and developed highly efficient theoretical solutions for single information systems like the text editing or spell checking of a manuscript (whose text has only one language). A rough analogy would be trying to write a spell checking code for a text written in two different languages but which tells a single story and has the words and grammar randomly switching between the two languages on a single word or sentence level.

One of the most common inherited causes of human disease follows the classical inheritance pattern first described by Gregor Mendel as autosomal recessive. Under these conditions, each genetic locus could have at most two alleles per individual, although a population could have a very large number of alleles. The normal or wild type was said to be functional, while all others were said to be mutant. Though mutant alleles were originally thought to be of null function, it was discovered that mutant alleles could have their own effects even when manifested as a recessive trait relative to the wild type allele in a heterozygous individual. This was explained by the diploidy of the chromosome count and the understanding of the nature of the double helix.

Later, with the understanding of the linear nature of the genetic code and that there was a finite number of genes, it became clear that multiple genetic changes could exist within a single genetic locus on either of the two copies of DNA in a single diploid organism. This is now designated as the inheritance state for an individual genotype at any one locus. For the classic recessive genetic inheritance states there now exists a discrete number of conditions: homozygous wild type (normal), homozygous recessive (mutant with both alleles identical at one position), a carrier of one mutant and one wild type allele (still normal), and finally a combination of two different mutant states in the same locus at different positions, which has the same effect as a homozygous recessive pair of alleles but is now designated as a compound heterozygous recessive inheritance state.

A very rare allele is often eliminated through negative selection in a population but can still appear occasionally. This appears at the highest rate when consanguineous mating produces a homozygous recessive inheritance state for a contiguous set of loci in an offspring. However, the vast majority of deleterious pairs of alleles inherited recessively for an outbred population will be expected to occur as a compound heterozygous recessive pair (halfHet pair).

The analysis of compound heterozygous recessive pairs thus must be undertaken with a dataset of all the variants that can be sequenced in the individual whose medical condition (i.e., the genetic trait) is being analyzed. These variants must then be compared to as many informative family members as can also be affordably sequenced. This group of sequences can be used to form patterns of inheritance states that can include (or more commonly exclude) entire lists of loci from being considered recessive.

There are three fundamental, direct tools for identifying candidates from a large list of all the variants in a single individual’s genome: Mendelian consistency, population statistics, and in silico predictions of deleteriousness.

The purpose of the following documentation is to explain the code written in Java that performs a compound heterozygous recessive filter function as well as several other filters on a set of NextGen data in the format of the VarSifter files (.vs format).

The basic strategy for the compound heterozygous filter is to avoid all use of predictions of deleteriousness in the actual filter and attempt to form all viable compound heterozygous recessive pairs, primarily using Mendelian consistency and secondarily by using population allele frequency. This is because these parameters are much less subject to false negative errors that reject potential real answers, and can be verified by reliable means.

An abridged summary is as follows:

1. The overall strategy is to define N genetic loci in a master list, where each locus is defined by a known transcript in the human genome that performs a different genetic function. Compound heterozygous recessive pairs must be at two different positions within that locus to be a legitimate pair.
2. Use three stages.
   1. First, go position by position through the entire genome looking for all the variants that can conceivably form a pair within a compound heterozygous recessive inheritance state. This involves three rejection criteria, with the third coming from any of several sources.
      1. The first criterion is that all affected siblings in a family must be heterozygous at the candidate position. All positions where this fails can be rejected.
      2. The second criterion is that one of two, but not both parents must be a heterozygote at the same position. Likewise, all siblings cannot be homozygous for the variant. False negative heterozygous genotype calls in a parent and false negative de novo mutations in all affected siblings are a rare exception to this criterion, which can be handled using a re-genotyping program.
      3. The third criterion is that – for any well studied human subpopulation – the allele frequency cannot be so high that this variant would have a substantial homozygous recessive frequency in a population large enough that it should have been noticed and already recorded in medical literature as a known genetic disease. The last criterion requires that sufficiently unrelated individuals are sampled at every locus to allow an accurate estimate of the true population frequency. This is possible for cases where there are fewer numbers of called genotypes (i.e., at the edges of exons and within introns during exome sequencing, poor amplification and poor capture in genome sequencing) by taking into account the probability that the population was not accurately sampled at low sample sizes that can occur in current databases at some positions. This utility uses the cumulative Poisson distribution 95%ile estimate that the allele frequency is greater than 1.2% or greater than 2% to avoid including or excluding a variant due to under-sampling.
   2. The second stage takes all the variants that fall “alpha” bases upstream of the start location of the transcript and “beta” bases downstream of the stop location for each gene locus (on a transcript by transcript basis). Every locus can have an adjustable alpha and beta parameter to allow more or fewer intronic variants to be within a locus designation. If fewer that two variants fall in an interval, the locus is determined to be rejected. When more than two “halfHet” variants exist in a locus, all combinations are interrogated for every permutation of all possible pairings to see which if any pairings can be found that meet criteria, meaning to be properly phased. One of the pair must come from each parent and no unaffected sib may be found to also be heterozygous for both halves of the candidate pair, but again every affected sib must be heterozygous for both variants.
   3. The final stage is to calculate a combined score for each successful pairing that is based on a heuristic function that weights predicted deleteriousness to favor two severely deleterious (deleterious-deleterious) half pairs and disfavor a benign-deleterious variant pairing, which would be merely a carrier state for a recessive condition and not an affected state. This score is compared between pairs to find a maximum score for the best pair at a locus that contains a pair of variants that meet the criteria of Mendelian consistency, population rarity and phase correctly in a small nuclear family pedigree. This is the output list for the code. This list is in the form of a smaller VarSifter file that has all the old annotations and the new pairing annotations. The “legacy” compound het button in the original VarSifter GUI as well as other compound heterozygous recessive variant-related filter functions are now active and available. Since there are rarely more than a couple of thousand variants that come out of this algorithm, this file can easily be viewed in Excel or other flat file spreadsheets as an alternative for viewing the results.
3. The list of viable candidate pairs can be processed further to remove duplicate variants and to allow more comprehensive interrogation of results at the BAM file level and additional Boolean logic. This process is entailed in the rest of the full Forwards Backwards pipeline, and should immediately start with Broad-level BAM Curation (see Manual 4).

Similar agnostic approaches were used to construct X-linked, Mendelian inconsistent, homozygous recessive, and de novo dominant filters. The details of these logical constructs are included below.

**Data Structures**

The data for the code presumes that the input is a flat file of data containing annotated variants with the structure of one variant per line (e.g., a VarSifter file). This flat file must also be annotated with the necessary population statistics that allow filtering at 1.2% and 2.4% allele frequencies (conservative estimates to avoid throwing out viable candidates).

*Standard contiguous gene compound het code:*

The data structure also requires an array that contains 4 elements for each locus: alpha; transcript start position; transcript end position; and beta. For the first approximation all alpha and beta values are 1000 bases long, and the start and stop positions for hg19 are taken from the UCSC genome data browser8. An example file is included in Supplement B, with the file name “Gene\_Config\_USE.txt.”

A separate algorithm allows for a gene to be analyzed for compound het pairings when the gene is split into two discontinuous regions with an intervening region containing other genes. Additionally, the halfHets from the internal gene(s) are also considered unlikely to cause a functional compound het recessive pair with any halfHet in the “split gene”. These are problematic loci and generate excessive numbers of false positive compound het pairings due to the large number of candidate halfHets and consequent combinatorial explosion.

To avoid this, a second array is made that has four adjustable parameters per locus (alpha, delta, gamma and beta). The split locus is now parsed for candidate halfHets in the regions covered by alpha minus the transcription start site to the transcription start site plus gamma, then also from delta minus the transcription end site to the transcription end plus beta. For that split locus, only the halfHets from both regions and not the interval between delta and gamma. This second compound het config file must have these 6 values for every locus interrogated for compound hets, and stored in a file named “Gene\_Config\_Splits\_USE.txt” which must be in the same folder as the .jar file (CompHetFilter\_Splits.jar). This algorithm is the one preferred over the simple first algorithm.

Currently the local NIH UDP cohort (*n =* 1341 exomes) is used as the raw data for frequency statistics. The variant allele count is listed but the total allele count is not present in the current data. To account for this program, we use the genotype count data (which is ½ the total allele count) and specify a 0.02 or 0.04 percent allele frequency. For data from the Appistry Inc. (GATK) alignment pipeline, there is a true "total alleles count" metric that can be used. For this data, a 0.01 or 0.02 percent allele frequency should be entered for the raw data in the “Please enter a maximum frequency in decimal format for the minor allele in population data” window.

The other data are values that yield a True or False value if the maximum subpopulation or maximum total population allele frequency is either >1.2% or >2% for any ethnic category, as determined by calculating a cumulative Poisson distribution (see Appendix). This can be determined for the five subpopulations in the ExAC database3, the 1000 Genomes project4, and the UK10K data5. Additionally, any external population-type data (e.g., ExAC) can be selected along with the local population in the compound het “halfHet” stage of the filter.

The memory requirements are reasonable for any modern workstation that can run Java, including UNIX, Mac and Windows-based PC. The data are parsed line by line and locus by locus and fit into a 4GB RAM space, much smaller than the requirements to open the source file (or even the result file) in VarSifter. Older versions of the compound het filter using VarSifter files with variant lists containing 3 million variants and 1000 exomes can run in under 13 minutes and typically produces fewer than 3000 lines of output, with no more than 10-20 viable pairs that have combined scores over 10 on the heuristic scale that looks at both the halfHets’ individual CADD scores. After implementation of multithreading, speed improvement is observed. Less than 10 minutes is the expected time for a typical 8 GB perfamily file. The speed improvement is expected to be more prominent on computers with multiple core processors. See VMM scoring for the best way to look at these result files for compound heterozygous recessive pairings.

**Compound heterozygous**

There are two subclasses used to implement the Compound Het Filter:

\* HalfHetNode: this subclass serves as container for a line containing a legitimate half het.

- The Java constructor takes in a VarSifter line split by tabs as an array. The Index, reference allele, variant allele, and PHRED score will all be stored in variables within the object. Additionally, the object will calculate the strength of the individual half het based on the type field, which of the two parents the half het comes from, and an Array List storing the genotype for each unaffected sibling.

- The score method takes in a second HalfHetNode and calculates the Best Strength, Best VMM, and pairs list. Scores are created for each call to the score method, but are only stored if they are greater than the value currently stored in each object (which is initialized to the smallest value). The Index of the pair will be added to the PriorityQueue pairs, which is sorted first by Strength and then by VMM.

- The getLine method returns a line ready to be printed to the output.

\* IntervalNode: this subclass contains intervals from the gene list config file as well as the halfHets that occurred within the locus.

- The constructor takes in a line from the config file. The gene name, transcript name, and chromosome are stored. The alpha value is subtracted from the transcription start position and stored as the front end of the interval, while the beta value is added to the transcription stop and stored at the end of the interval. Additionally, a reference to the next node is stored, creating a linked list once these nodes are placed in an Array List.

- The add method adds a halfHet to the Array List of HalfHetNodes stored within the interval node.

- The addTranscript method appends an additional transcript that appeared on a different line of the config file. The minimum start and maximum stop are stored as the new values, and the transcript name is appended to the transcript variable for the IntervalNode.

- The printLines method prints out all the half hets within the IntervalNode’s ArrayList into the input PrintWriter.

- The interval nodes for the split compound het are generated in the same way.

a. Selection of half heterozygous recessive candidates

The halfHet filter is inherently problematic, as it cannot be done with a single line-by-line iteration through the data. Individual lines can be judged for Mendelian consistency and population frequencies with a line-by-line iteration, which is used to select halfHets (potential alleles that could be paired with a second allele at the same locus. However, a second iteration is required to complete halfHet phasing within each single gene locus, to form a complete compound heterozygous recessive pair candidate for the genetic cause in an agnostic genome wide anaylsis.

The boundary positions of loci present another problem. There is no published database that has a consensus of where any one gene locus starts or ends (or if there are internal positions that are not truly part of that gene’s locus – i.e., a split locus with another separate gene locus entirely internal to the outside parts of a surrounding gene locus). Since halfHets can only complement within the same gene, these boundaries must be well defined prior to filtering or many false pairs will be potentially made and can easily overwhelm this type of analysis. The first implementation program uses a config file, “Gene\_Config\_USE.txt,” which contains transcript names and their transcription start and stop values, and an alpha and beta value for each gene. As a first approximation, these alpha/beta values extend the locus boundaries up and downstream, respectively, into noncoding regions. The alpha/beta regions are added on to the start/stop positions, with all alpha and beta values initially set at 1000 base pairs. Since this config file is not hard coded and can be modified, these values can be changed as the topology of the genome becomes better understood at each locus in an ongoing heuristic attempt to minimize false pairings of two alleles in a compound heterozygous recessive combination. Ultimately, these values will be modified for every genetic locus to allow accurate pairings of all appropriate halfHets without any illegitimate pairings from two different loci. For the current use, however, enough editing has already been done to avoid the great majority of false pairing and this data set is used in the current analysis with apparently sufficiently good effect.

This config file must be prepared prior to running the code and be present in the same folder as the code .jar file. Intervals will be stored in the IntervalNode class discussed above, which will contain the gene name, the appropriate list of transcripts, start and stop positions, and an ArrayList that will contain the halfHets for the interval. The IntervalNodes are stored in a LinkedList, which will be tested every time a halfHet is found. This allows halfHets to be placed in both IntervalNodes of an overlapping interval if the positions are appropriate. Thus, different transcripts can have their own “compound het pairings” performed, even with a nearly identical overlap of positions in the genome.

Once these IntervalNodes are set up, the filter begins the process of detecting and sorting halfHets within the input VarSifter file. Lines are first judged on Mendelian consistency, where several criteria must be met:

- One but not both parents must be heterozygous. If both parents are heterozygous at a single halfHet position, then that position would have to phase with another halfHet at some other position. However, that would make one parent affected, which this filter assumes is not the case. Thus, the condition is a XOR of the parents’ heterozygosity at every halfHet position.

- The proband must be heterozygous.

- Unaffected siblings cannot be homozygous recessive. This would result in the same gene being damaged in an unaffected sibling, meaning that said sibling should also be affected. Non-penetrance is presumed to have been excluded at the phenotyping stage prior to the sequencing.

Second, the population data are tested based on the fields the user selected at the start of the program. Preexisting filter fields within the ExAC, 1000 Genomes, and UK10K master datasets can simply be read as true or false – true values imply that there is a greater than 95% chance the variant is more common than 1% or 2% (depending on the field selected). Additionally, the user may select raw data, which will be tested using the Poisson cumulative probability function (for more details, see Appendix 1). If the calculation shows that it is more than 95% likely that the variant is present in a value greater than the rejection threshold input by the user, then the variant line is rejected.

If a given line passes both the Mendelian consistency and population frequency data tests, it is determined to be a legitimate halfHet. The line is used to create a new halfHetNode object (see above), which will store variables used later for fast access. This halfHetNode object will also have a strength score generated based on the ‘type’ field. Strong values are given a score of 3, intermediate values are given a 2, and the weak values are given a score of 1. The specific labels that fall into the strong and intermediate categories are specified by a config file, “Strength\_Config.txt.” This file can be modified based on what labels the user wants to place in each category so long as the ‘#Strong’ and ‘#Intermediate’ fields are included. Weak values do not need to be specified because a weak score is the default value. A typical strong example is a stop mutation or a nonsynonymous codon change for a transcript. An intermediate example would be a synonymous codon change, or a 3-prime UTR coding change. An example of a weak change would be an intronic change more than 20bp from an intron-exon boundary.

This halfHetNode object will then be added to the ArrayList within the current IntervalNode, as well as any subsequent IntervalNode with appropriate boundaries.

b. Phasing halfHets using parents and siblings

Once the entire VarSifter file has been scanned for halfHets, the IntervalNode objects will each contain an ArrayList of halfHetNodes, which contains the halfHets found within that interval. Many of these ArrayLists will be empty, and that means that they can be skipped in the next processing step.

The program will loop over the linked list of IntervalNodes and enter each object’s ArrayList into the findCHetPairs function. This function will take each maternal halfHet within the input ArrayList and attempt to phase it with each halfHet in the ArrayList. The criteria for proper phasing are as follows:

- A maternal halfHet can only phase with a paternal halfHet.

- Unaffected siblings cannot have both halves of a compound het pair.

Each time a compound het pair is made, the two halfHetNodes that form the pair are scored in three steps. First, the strength scores of each halfHet are multiplied to generate the score for the compound het pair. If that score is higher than the current ‘Max\_Strength’ score in either halfHetNode object, it is stored.

Second, a combined prediction of deleteriousness is calculated using the CADD Phred scores from each of the halfHets (also now done with Eigen and EigenPC scores for SNPs and the CADD Phred scores for indels). The goal is to have a score that will favor two moderately damaging variants over a single extremely damaging variant paired with a non-damaging variant (carrier state). This is achieved with a VMM score, defined by the following function (P1):

(P1)

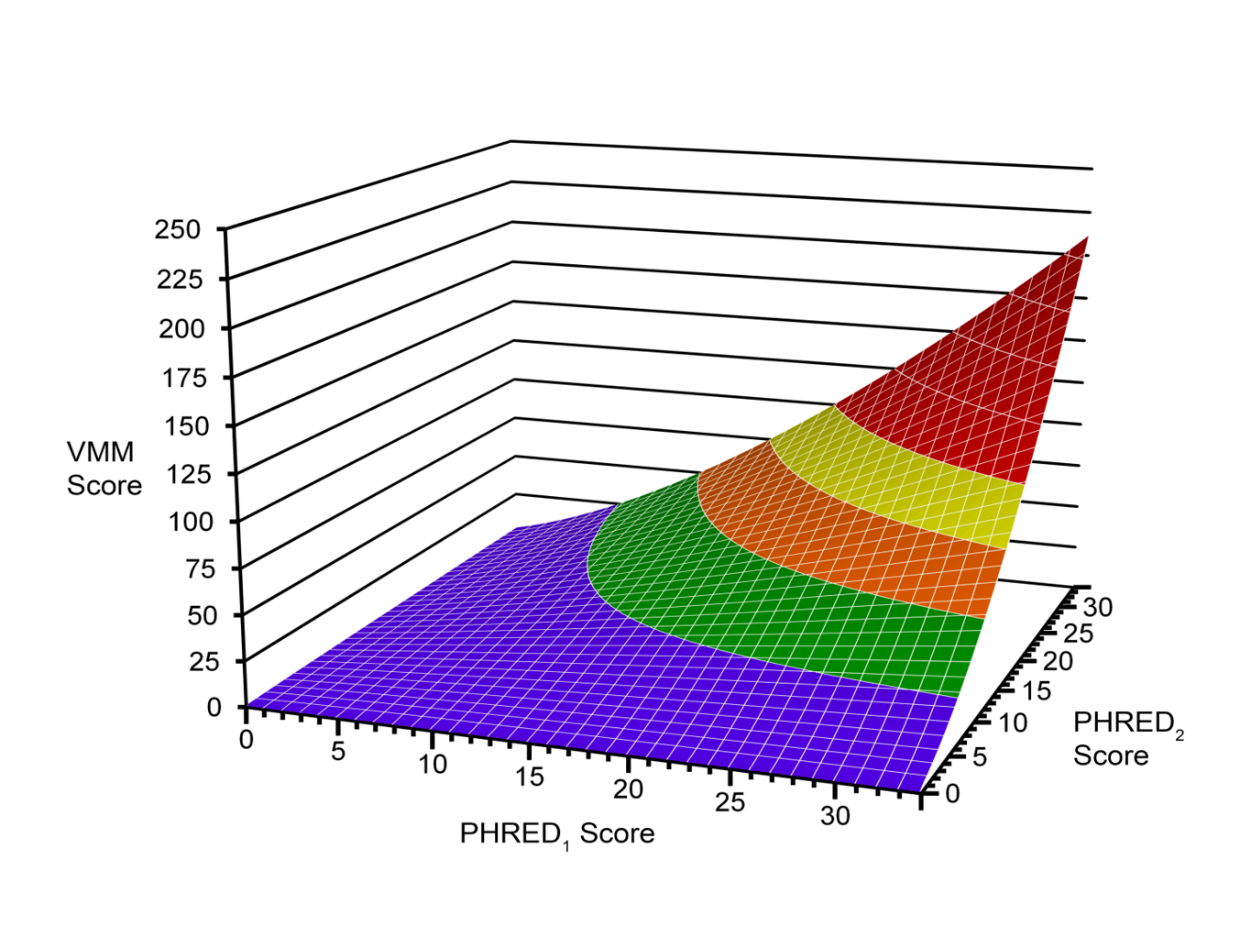


Figure 10: *The x- and y-axes represent the two CADD (Phred) scores and the z-axis is the associated VMM score. The longer wavelength colors denote regions more likely to be truly deleterious for the pairs of variants.*

c. Ranking pairs of halfHets by conjoint scoring

The less purple and redder the VMM score, the more likely the pair is going to be a truly deleterious pair of compound het recessive alleles at that locus (Fig. 10). The other figure has the reverse coloring system that goes from cool purple to hot red as the VMM score rises.

As the graph shows, this function achieves the desired effect, i.e., scoring two intermediate variants very strongly while scoring the combination of an extremely weak and extremely strong variant very poorly. If the calculated VMM score is greater than the current ‘Max\_VMM’ in either halfHetNode object, then that value is stored in place of the previous score.

The last step is to add the index of the pair to each halfHetNodes’s pair’s variable. The pair’s variable is a PriorityQueue containing the indices of each line the given halfHetNode pairs with sorted first by the strength of the pair and then by the VMM score. When the halfHetNode is printed out, the indices will be printed in a comma-delimited list in the MendHetRec field, which will allow the VarSifter GUI to view the compound het pairs.

Any halfHetNodes within the ArrayList that have no pairs are removed.

Some apparent de novo genotype relationships are false negative potential halfHets for an inherited allele. These occur when a child is a het and both parents are apparently homozygous reference but one of the parents have low coverage. These few cases can be salvaged (see Manual 2). There are also some genes that have so many variants that there is a combinatorial explosion of compound het pairings. By applying the Mendelian consistency and population frequency filters first, the duration of this process is greatly lessened, and appears to have made this problem tractable for an agnostic genome-wide search of exome data for all but the largest transcript loci, and all but the most poorly aligned loci. Adding intronic variants may or may not cause this problem to reappear depending on how stringent the filters are for deleteriousness or population frequency.

**Homozygous Recessive Filter:**

A Homozygous Recessive Filter is also used in the program. This filter utilizes the same pedigree information, frequency data, and user inputted file as the Compound Heterozygous Filter.

The filter, HomRecDetection, is a Boolean function and will return true if the following conditions are met:

Both parents’ alleles are heterozygous while the proband is a homozygous variant at the same position.

If the proband has an unaffected sibling:

- The sibling cannot be a homozygous variant.

- This means they can have any other genotype including “NA”

If the proband has an affected sibling:

- The sibling must be a homozygous variant or “NA”

The filter also considers the HGMD value of that variant if “DM.”, but not for “DM?” or other less confidently scored variants. If so, the function returns true regardless of population statistics. Additionally, the program will return false if any of the population data indicates that the allele is too common (i.e. > 5% for a conservative rejection value).

After the program reads each line of the VarSifter file and stores the value in an array (separating each with a tab), the filter is applied. The array is passed through the filter and any values that return true are printed in a new file.

In the outputted VarSifter file, the values that are *true* or are homozygous recessive will have a “1” and the values that returned false will hold a “0”, per the original VarSifter notation concept and thus activating the selection buttons in the VarSifter GUI.

**Mendelian Inconsistencies**

The program also checks for any Mendelian Inconsistencies or errors, which occurs when an allele in an individual does not follow normal Mendelian inheritance.

The Mendelian Inconsistency checker uses the same pedigree information as the other filters in the program and filters any inheritance inconsistency within the family. The program assumes two purposes for this filer: One is to verify parental consistency. For this purpose, any line that exhibits inconsistency in the proband or the sibling is returned. The second purpose is to examine any parental inconsistent lines that are likely candidates to cause diseases. For this purpose, the same population filter used in the compound heterozygous filter is used - to only pass through lines that are inconsistent and rare as judged by the Poisson distribution maximum minor allele frequency (MAX-MAF) criterion.

The program also asks the users to choose whether they would like to apply the extra population filter to the Mendelian inconsistency check. The Mendelian inconsistency is only checked on the proband, not his/her siblings.

Within the program, the following phrases are used:

-homvar (meaning homozygous variant), homref (meaning homozygous reference, het (meaning heterozygous), and NA which is the same in the VarSifter file.

The following situations will be returned as a Mendelian inconsistency:

* Both parents are homvar or homref, the proband is het
* Both parents are homvar, the proband is not homref nor NA
* Both parents are homref, the proband is not homvar nor NA
* One parent is homvar and the other is het, while the proband is homref
* One parent is homref and the other is homvar, while the proband is not a het or NA

The filter is applied like the Homozygous Recessive filter. It uses the function that reads each line of the VarSifter file and stores the values in a tab-separated array. Any lines in the file that are found to match the Mendelian inconsistencies will be written in the new file.

Adding this filter to the program also modifies the VarSifter file because a new column, “MendInconsis” is added, with values indicating the type of apparently inconsistent inheritance:

* 0 indicates Mendelian consistent inheritance in the pedigree.
* 1 indicates Mendelian inconsistency in the pedigree.
* 2 indicates de novo dominance.
* 3 indicates hemizygosity type of apparent Mendelian inconsistency.
* 4 indicates an apparent extreme novel deleted exon (CNC).
* 5 indicates the case where an apparent deletion is not extremely novel.

An important note is that the Mendelian inconsistency button on VarSifter is activated in the KaylaKode output file but does not perform accurately when selected in the VarSifter GUI. The custom query field should be used to look at Mendelian inconsistent changes in exome analysis as has been used by the authors for >5 years ever since the older buttons on the VarSifter GUI were deactivated.

**De novo dominant**

Any line with parental inconsistency is also checked for de novo (Mendelian dominant) events. Population frequency is always considered when examining de novo dominant candidates. This explains the situation when there are lines that display both inconsistency and de novo dominant events, but return false for de novo because it means the line did not pass the population frequency filter (possibly a true de novo, but too common to cause disease). For searching for these Mendelian inconsistency, it is optional to use the population frequency values from multiple population samples if they are annotated into the flat file used during analysis.

De novo dominance occurs in the following situations:

* Both parents are homref or homvar, the proband and affect sibling are both hets, and the unaffected sibling expresses the same genotype as the parents or are NA.

The de novo dominance filter is found in the same method as the Mendelian inconsistency filter and will print to the output file the same way. A new column will appear specific to de novo dominance. However, the filter can only be applied through the custom query unlike the other filters that are part of VarSifter and are shown as a checkbox.

Additional comments about the filters: All of the filters are applied to the same file and are outputted in the same file. The information regarding each filter is found in a column in the metadata of the VarSifter file.

**X-linked**

The Boolean logic for X-linked variants is as follows: mother is heterozygous, male affected is hemizygous for the minor allele while the unaffected male is hemizygous for the major allele. The gender information is input by the user so no additional pedigree file is required. AA (A denotes reference allele) or A for males at chrX, non-autosomal regions are counted as hemizygous reference. The autosomal region is position<2699520, position>154931043 and position between 88400000 and 92000000 on chrX.These autosomal regions are treated in the same manner as autosomal loci during the KaylaKode processing.

**The reference has a minor alllele completeness problem and how this code addresses that possibility.**

This addition to the analytic tool considers the case where the reference allele is the minor allele. The MAFixer annotation program provides filter fields that calculated the minor allele frequency for the reference allele. Thus, when a reference allele is determined to be the minor allele, the population frequency filter examines the reference allele frequency instead of the variance allele frequency. The field that determines whether the reference allele is the minor allele is input by the user. In All\_Gahl cohort, it is the field titled “All\_Gahlref\_is\_minor” which was produced by the NHGRI intramural sequencing center for those exomes. For snpEFF annotated flat files, the appropriate column would be different and must be present in the flat file for this analytic module to function properly. It is assumed to be a numeric field, with “1” indicating reference allele being minor and “0” indicating reference allele not minor.

**Multi-threading**

To better accommodate users with different hardware, multi-threading is employed. Even though multithreading is still limited by the line-by-line model, which requires sequential reading of the input file, it parallelizes compound het matching and line writing. However, it requires a more stringent input file format: chromosomes with underscore or chrUn or chrM in the gene config or contained in the VarSifter input files are not considered in this current version. In other words, only chromosomes chr1 to chr22, chrX, and chrY are considered at this time. Both files must be sorted by chromosome (as string) and ordered by position. The wrong chromosome order in the input VarSifter file is another way the currently written program will hang without reporting an error message. The Mitochondrial chromosome can be analyzed very easily by the older semi manual methods (including using the VarSifter GUI in the custom query Boolean fields)due to its small size.

General implementation of the multithreading is as such: the gene config file is read in line by line, an IntervalNode structure is constructed for each chromosome; as soon as an entire chromosome IntervalNode structure is complete, a thread is sent to process line by line the VarSifter input file. While one thread is reading, a latch is opened so that another thread cannot access the input file. Each thread reads exclusively lines of the same chromosome. As soon as the file reaches the next chromosome, the latch opens and the next thread takes over the processing. At the same time, the compound het candidates from the previous thread are sent to be matched. When compound het candidates are numerous, the multithreading advantages are obvious. It also requires less memory space as the gene\_config file is not saved entirely but divided by chromosomes and generated or discarded as the VarSifter file is being read.

1. **Automated Broad-level BAM Curation/SNR Calculation Module**

**Executive Summary**

This module resembles the first half of the Salvage Pathway module. It calculates the same density pattern values around the candidate positions that come from the candidate variants derived from the KaylaKode. Most of these candidate positions are not mendelian inconsistent places that were evaluated by the first half of the Salvage module, so these locations now require the same analysis for their local region variant density as the ones evaluated in the Salvage module. At this stage, the variant candidate list is at most a few hundred to a few thousand, making this intensive evaluation of the local BAM file regions computationally feasible.

Next, the program calculates the Signal-to-Noise Ratio (SNR), which is based upon a heuristically determined formula (see Program Overview). An Error term is derived from the SNR, which is used in various ways in the next module.

The two numerical values for SNR and Error are added to each variant’s metadata as new columns in a per family file flat file. In addition, two elements are added to every family member’s genotype data (All count and Good Count). Of these four descriptions, two (SNR and error) are for the local region of all the family’s BAM file reads, and two (All count and Good Count) describe how many good reads are present for each family member and how many total reads are present in the region for each family member.

These terms are fed into the next two modules and contribute to a complex set of decisions made for each mendelian model that incorporates these values. Those decisions ultimately determine whether to accept or exclude those variants as candidates for being real (accurate, true) variants that will be printed in the final list. The processes are called “BAM curation” (this Module 4) and Exclusion Filtering (Module 5).

**Program overview**

The program takes in the VarSifter and pedigree files (inputs 1 and 2), and also locates the family’s BAM files based on the BAM file directory (input 3). The output of the program is a flat (non-VarSifter) file with “SNR” and “Error” columns appended to the end of the metadata section, along with the variant genotypes for each family member, “AllCount” and “GoodCount” columns specific to each family member, and re-genotyped status of each family member’s genotype call. All other sample’s genotype data (i.e., non-family members) are removed at this stage, since they will not be used by the later modules of the pipeline. The output of this program is NOT readable by the VarSifter (GUI) program, because each family member has “AllCount” and “GoodCount” columns, which violates the VarSifter (GUI) requirement that no two columns can have the same name. This next module, the variant exclusion filter, uses the data from these AllCount and GoodCount columns, but does not include them in its VarSifter-compatible output flat file, returning the final output of the next module and all subsequent modules as a file readable with the VarSifter GUI.

*Calculate SNR* *and error values*

The intent of the program is to examine how variant bases are distributed within a local BAM file region. This is accomplished by determining a Signal-to-Noise Ratio (SNR), which can be considered a type of “speckle” and which is calculated from the Spatial Distribution Score a and the Mismatch Density (see formula below).

The Spatial Distribution Score is similar to a covariance of the variant base call density function within the region. It is generated by the following procedure:

First, we examine the region -150 bases to +150 bases on either side of the variant (in the reference genome) whose local region is being evaluated.  All short reads that were mapped into this interval are used for all positions that aligned to each base in the reference genome.  The individual bases that align to each of these positions are binned and the number of mismatches counted for that position (X).  All of the 300 positions are then collected and sorted by putting the highest number of mismatches in the middle position and alternating down the list to generate a two-tailed distribution of mismatch counts centered in the middle position. The positional distance of this reordered list is now considered a weighted distance (designated loc) spanning from -150 to +150.  The region is scaled (normalized) to a total of 15000 bases (300 x 50) to give a scaling coefficient (scale) that is equal to 15000 divided by the actual number of total base calls in this interval in the BAM file.  Finally, the weighted distance is squared, then multiplied by the scaling coefficient and multiplied by the number of mismatched bases at that distance from the central position.  The sum of each of these individual elements gives the summed Spatial Distribution Score squared:

A spatial distribution value of 0 represents a pileup with no mismatches or no reads mapped in the region of interest. The higher the spatial distribution metric, the higher and more random the mismatch distribution.

The Mismatch Density is the total number of variant called bases divided by all of the called bases in all the reads, for bases falling +/- 150bp on either side of the variant of interest whose region is being evaluated.

The SNR is the (log base 10 of one divided by the Mismatch density) divided by the Spatial Distribution Score, i.e.,

The top part of SNR captures signal (so lower the mismatch density, higher the signal), the bottom part of SNR captures noise (the greater the Spatial Distribution Score, the greater the noise).

We then define a term, “error”, empirically as 25% of the inverse of SNR.

A typical error rate of a salvaged file is shown below:

The curve moves left or right depending on how noisy the overall data is. (For example, UDP984 curve is to the right, but a good data sample could be much to the left

These models were developed heuristically based on approximations of noise.

*Read BAM files to determine genotype calls and call quality*

Once the error values have been calculated, the sequences in genotypemap and goodgenotypemap are divided based on if they contain the reference genotype or the variant at the VOI position. These reference and variant genotypes are identified by the “ref\_allele” and “var\_allele” columns in the input VS file.

From each family member’s pileup, the following array is generated:

{[ref, var], [goodref, goodvar]}

Where “ref” refers to reads in genotypemap that contain the reference, “var” refers to reads in genotypemap that contain the VOI, “goodref” refers to reads in goodgenotypemap that contain the reference, and “goodvar” refers to reads in genotypemap that contain the reference.

These values are appended to the genotype data for each family member in the output flat file under the “AllCount” and “GoodCount” columns.

*Use of SNR, Error, AllCount, and GoodCount*

These four terms are employed in a complex set of decisions to yield a final determination of whether a variant will be included or excluded by the BAM Curation and the Variant Exclusion filters. The code provides details of the contribution of the Error term, which varies with the inheritance model and the filter.

*Determine “salvaged” status of each variant*

All variants in the input VS file should have been processed by the Salvage Pathway earlier in the pipeline (see Manual 2). In the process, either the genotype score or genotype coverage values were multiplied by -1 to mark the variant’s re-genotyped or “salvaged” status. This module makes each variant’s salvage status more immediately accessible by modules further in the pipeline by marking each variant with “NN,” “NP,” or “PP,” where “N” indicates negative values and “P” indicates positive values. For example, “NP” means that the genotype coverage value was made negative while the genotype score value remained positive. Variants marked with “NN” or “NP” are thus results of the re-genotyping process. These salvage status values are also appended to the genotype data for each family member in the output flat file.

1. **BAM Curation and Variant Exclusion Filtering Module**

**Executive Summary**

This module is a complex filter that uses different criteria for different mendelian models. It includes filters for compound heterozygous recessive, de novo, X linked, hemizygous deletion, and homozygous recessive models. For each model, variants are split into two categories: 1) variants where average reads per family member are <20 in the BAM files, i.e., a low coverage region; and 2) variants where the average reads are equal or greater than 20 reads in the BAM files. After that splitting, each model has separate filters for groups 1 and 2 but most differences are at the threshold values and not the actual structure of the filters (quantitative but not qualitative differences). The first concept exploited is the difference of good quality reads versus lesser quality reads, defined by base quality PHRED score equal or greater than 20 in a short read with a map quality PHRED score greater than or equal to 30. If the expected read at the position matches the reference call it is added to the family pileup (respectively). Then the pileups are evaluated to see if they meet the criteria to stay in the candidate list; they are excluded i the BAM quality is insufficient and the candidate is not likely to be real, even though it appears to follow a mendelian model and be deleterious (i.e. passed through the KaylaKode logic). In addition to the model-specific exclusion rules, this model uses the BamRegionQuality value from Module 4 by excluding variants with regional scores >0.02 or >0.01 if the average read depth per family member is <4 reads in that region.

**Program overview**

This module is the main filter of the pipeline, and excludes variants based on population frequency, noise level, and predictions of deleteriousness. Previous modules in the pipeline—specifically, KaylaKode and broad-level BAM file curation (Manuals 3 and 4)—phase and annotate the flat file in preparation for variant filtration.

Four columns—PopFlags, BamFlags, MendFlags, and MiscFlags—are appended to the end of the flat file. The PopFlags column contains annotations relating to population frequency based on datasets included in the input file (ExAC, gnomAD, 1000 Genomes, etc.). The BamFlags column includes annotations relating to noise and error evaluations derived mostly from the Broad-level BAM File Curation module. The MendFlags column denotes the type of inheritance model previously identified by KaylaKode, and the MiscFlags column contains all other annotations. The annotations are described in Tables S1 and S2. Each family member’s AllCount and GoodCount columns (from BroadBamCurate) are not included in this module’s output flat file. This makes this module’s output compatible with the VarSifter flat file program.

The full population frequency filter criteria specific for each model of inheritance are described in Table 3 (below), and the full exclusion criteria requiring BAM file evaluation for each model of inheritance are described in Table 4 (below).

In the case of compound heterozygous recessive variants, if half of a compound heterozygous recessive pairing were excluded by the criteria detailed in Tables 3 or 4, then a new VMM score (Manual 3) was derived from the half of the original pairing that was not excluded, along with the next best candidate variant that satisfies the phasing for a compound heterozygous recessive pairing.

The final filter in this module is an array of deleteriousness prediction thresholds. The exonic CADD Phred score threshold is set by the user (input 4). The intronic CADD threshold is calculated as follows:

This threshold used for the VMM threshold is derived from the exonic CADD threshold using the formula:

The exonic CADD threshold is used for all non-compound heterozygous variants with locus type of “exonic” and not “ncRNA.”, based on annotations in the VarSifter file inserted by a transcript aware annotation code such as ANNOVAR or snpEFF. The intronic CADD threshold is used for almost all other locus types, including “intronic,” “intergenic,” “UTR3,” “UTR5,” “ncRNA exonic,” “ncRNA intronic,” “upstream,” “downstream,” and “splicing.” The VMM threshold score was used for compound heterozygous variants. Any variants with CADD or VMM scores below their appropriate threshold are excluded from both programs’ outputs.

All variants with any of the flags marked with an asterisk in Table 1 are used to exclude them from the ROC output. These variants are still included in Output 2 for user review.

**Table 1.** *Explanation of the different flags used in the filtration algorithm. Each of these flag categories (e.g. “****BamFlags****”) is a column annotated to the end of the VarSifter file. Flags with asterisks were used by the program to determine which variants get excluded or included into the ROC output and, ultimately, into the proband’s final deleterious variant output list. Flags with double asterisk means DM tags that are less than 5% allele frequency for all of the databases used in this analysis.*

|  |  |  |  |
| --- | --- | --- | --- |
| **Category** | **Flag** | **Meaning** | **Filter/flag criteria** |
| BAM Curation and Quality (**BamFlags**) | BR\* | Poor BAM region | See Table 3 |
|  | OU\* | Poor BAD curation | See Table 3 |
| Salvage Status Indicators (“N” = negative/salvaged; “P” = positive/salvaged) | NP | Salvaged: This was in a noisy region |  |
|  | NN | Salvaged: This was not in a noisy region |  |
|  | PP | Not salvaged |  |
|  | UD | Unsalvaged de novo |  |
| Population frequency  (**PopFlags**) | HP\* | High population | MaxMAF>2% or 1% (Conditions differ for each Mendelian model), or count based (See Table 3) |
|  | HF\* | High population rejected due to population frequency | MaxMAF>2% or 1% |
|  | HC\* | High population rejected due to absolute count | For compound heterozygous recessive, if at least one parent’s variant is MaxMAF<1%, the variant passes the filter. |
| Miscellaneous (**MiscFlags**) | BG\* | Big gene | Variant was in the loci *NEB*, *TTN*, or *OBSCN*, and any *KRT, OR, or TAS* genes |
|  | GP\* | Gahl Count above 1 | Internal UDP cohort count > 1 |
|  | DM\*\* | Deleterious mutant | HGMD DM, or M-CAP > 0.5 |
|  | LC\* | Low count | Internal UDP cohort genotyped count < 300; 1000 Genomes cohort allele number < 1500 |
|  | SA | Salvaged |  |
|  | SR | Salvaged homozygous recessive |  |
|  | SH | Salvaged compound heterozygous recessive |  |
|  | Oth | Other genotypes at the variant position available beyond ref and var |  |
| Inheritance model (**MendFlags**) | CM | Compound heterozygous recessive |  |
|  | DN | De novo |  |
|  | HR | Homozygous recessive |  |
|  | HM | Hemizygous |  |
|  | XL | X-linked |  |
|  | CNC | Call no call (extreme novel exon deletion) |  |

**Table 2.** *Explanation of the key configuration categories in the header configuration file.*

|  |  |
| --- | --- |
| **Category** | **Definition** |
| Tot\_AC | Variant allele count |
| Tot\_AN | Total allele count (reference + variant +other) |
| Tot\_AF | Variant allele frequency |
| HomRef\_Count | Homozygous reference genotype count |
| HomVar\_Count | Homozygous variant genotype count |
| Het\_Count | Heterozygous variant genotype count |
| Other\_Count | Other genotype count |
| Genotype\_Count | Total # individuals genotyped at the variant in the cohort |
| HemRef\_Count | Hemizygous reference genotype count |
| HemVar\_Count | Hemizygous variant genotype count |
| MaxMAF1 | MaxMAF>1% |
| MaxMAF2 | MaxMAF>2% |
| AFR\_AF | African/African American variant allele frequency |
| EAS\_AF | East Asian variant allele frequency |
| AMR\_AF | Native American/Hispanic variant allele frequency |
| EUR\_AF | European variant allele frequency |
| SAS\_AF | South Asian variant allele frequency |
| OTH\_AF | Variant allele frequency for other ethnicities (e.g. Ashkenazi Jewish, Finnish European) |
| AFR\_AC | African/African American variant allele count |
| EAS\_AC | East Asian variant allele count |
| AMR\_AC | Native American/Hispanic variant allele count |
| EUR\_AC | European variant allele count |
| SAS\_AC | South Asian variant allele count |
| OTH\_AC | Variant allele count for other ethnicities (e.g. Ashkenazi Jewish, Finnish European) |
| AFR\_AN | African/African American total allele count |
| EAS\_AN | East Asian total allele count |
| AMR\_AN | Native American/Hispanic total allele count |
| EUR\_AN | European total allele count |
| SAS\_AN | South Asian total allele count |
| OTH\_AN | Total allele count for other ethnicities (e.g. Ashkenazi Jewish, Finnish European) |

**Table 3.** *Variant exclusion criteria requiring population frequency and count data.*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Mendelian model** | **UDP** | **ClinSeq** | **ExAC** | **MaxMAF>1%\*** |
| **CM** | HomVar count≥3 | HomVar count≥3 | Homozygous count≥3 | X (neither halfHets meets this) |
| **DN** |  | Het count>2 | Total allele count>3 | X |
| **XL** |  | Male-only variant allele count>2 |  |  |
| **Hemi** |  | HomVar count>2 | Total homozygous allele count>3 |  |
| **HR** |  | HomVar count>2 | HomVar count>3 |  |

**\*Includes ethnic subpopulations (1000 Genomes, UK10K, ExAC, gnomAD, gnomAD exome).**

**Table 4. Variant exclusion criteria requiring BAM file evaluation.** Full criteria for automated BAM file evaluation specific to each Mendelian model of disease inheritance. Full de novo and extreme novel deleted exon (CNC) criteria are shown in the manuals for the pedigree-aware, multiparametric BAM file noise evaluator and the CNC filter, respectively. These apply for all thresholds determined by predictions of deleteriousness. CM = compound het. DN = de novo. XL = X-linked. Hemi = hemizygous. HR = homozygous recessive.

|  |  |  |
| --- | --- | --- |
| **Mendelian model** | **BAM file evaluation criteria** | |
| **Average reads per family member < 20** | **Average reads per family member ≥ 20** |
| **CM** | Less than 2 good quality minor allele reads per parent | Less than 5 good quality minor allele reads per parent |
| OR  0 good quality minor allele reads in the proband | OR  0 good quality minor allele reads in the proband |
| **DN** | Minor allele count in proband - Minor allele count in rest of family < 3 | Minor allele count in proband - Minor allele count in rest of family < 3 |
| **XL** | Proband major read count>2  OR  Anyone else’s major read count < 2 | Proband major read count>5  OR  Anyone else’s major read count < 5 |
| **Hemi** | Proband major read count>2  OR  Anyone else’s major read count < 2 | Proband major read count>5  OR  Anyone else’s major read count < 5 |
| **HR** | Proband major read count>2  OR  Anyone else’s major read count < 2 | Proband major read count>5  OR  Anyone else’s major read count < 5 |

1. **Pedigree-Aware, Multi-Parametric BAM File Noise Evaluator (Confetti Filter) Module**

**Executive Summary**

This module, called the pedigree aware, multiparametric BAM file noise evaluator, consists of 3 different subroutines to determine if a local BAM file region is unlikely to be a real variant.

*Horizontal*

The "horizontal" method involves interrogating along each read in a given BAM file within a range of +/-50 bp of the variant of interest's (VOI) position. If the number of poor quality bases (base quality < 20) exceeds 10 or if the map quality of the entire region is below 30, then that individual read gets stored as a bad read. If the ratio of bad reads to the total number of reads in the region for the person is greater than or equal to 0.25 (so 1/4 of the reads in the region are of poor quality), then the region is considered poor quality and rejected.

*Vertical*

The "vertical" method involves going down each read at only the position of the VOI for each person and comparing the number of times a variant call is made for each person. For example, if only the proband has variant calls at the VOI position, then the potential that it's a de novo (for an adequate read depth), increases with every variant seen only in the proband, and with every reference read seen in the other family members. On the other hand, if each of the other family members ALSO have 2+ variant calls at the VOI position, then it's likely that all the variant reads were miss-mapped and probably the apparent de novo in the proband was miss-genotyped. The same goes for the condition where any other family member, besides the proband, has multiple (3+) variant calls, even if the other family members don’t have any variant calls. The exception to this would be a deep reference base call (only) set of reads at the VOI in both parents but with a few variant reads in more than one offspring. That could be a germline mosaic condition.

**Note**: It is necessary to consider the quality of the base call at the VOI position. It is necessary to consider and account for the possibility that the position was miss-genotyped due to poor quality variant calls being (almost) the only variant calls in that position. However, the presence of high quality reads only in the proband, and yet only low-quality variant bases present in the other family members, is a truly improbable coincidence and this is a nonzero but presumably very low possibility not currently accounted for by the current code.

*Both - haplotype counting*

Because each person has only two chromosomes and therefore (assuming no mosaicism or deletions/duplications), there should (except directly at crossover regions) be at most 2 haplotypes in a given region, one for each chromosome--ref and alt/variant. If there are more (i.e., multiple reads don't match up with each other), then it is likely the read was mis-aligned/mis-mapped, which can cause low quality BAM file regions that still pass Mendelian filters and yet can be excluded by testing for haplotype inconsistency.

The way the program works is to first store each read in one of two (key & value) hash maps, with the determinant being whether or not, at the VOI position, the base call is reference (refMap) or anything else (altMap). The key is the read sequence itself (readString) and the value(s) are the VOI position's coordinates within the readString ([posVarStart, posVarEnd]). This becomes further complicated due to the potential of soft clipping and/or indels occurring within the raw read sequence. In order to mark where indels have occurred in the sequence, the readStrings were padded with "D"'s for deletions and "I"'s for insertions to account for these complications that would interfere with using the read sequence as a correct key value in a key/value dictionary hash map. These do not change the readString sequence overall, but rather mark where inside the readString those indels occur.

Once all the reads have been mapped to their respective maps (refMap or altMap), then all reads with variant reads at the VOI position (altMap) are exhaustively compared, one at a time, with one another at all the remaining positions in the two short reads as aligned in the pileup. If the two reads differ from each other within the positions where they align (as anchored by the VOI position), then they represent different haplotypes. On the other hand, if they agree within all those positions, then they are part of the same haplotype. For example, the most likely condition encountered would be for all the (non-VOI) positions in both short reads to have all remaining positions be called reference at each position in the region until the 20bp limit. In that case those two reads would both have the same haplotype over that region of the pile up.

If a read differs from every other read, then there's a potential that the former is just a badly aligned read and the others form their own correct haplotype. However, if there are too many of those "one off reads" (here determined to be either >=10 or at least the total number of reads - 2 (provided there are at least 5 reads)), then the region is just poorly aligned overall, and very likely full of mis-mapped reads. Notice that as the number of “one off” short reads increases, this type of region evolves from a region that is “ghost like” to one that is “confetti like”. If the number of disagreeing query reads is at least twice the number of agreeing query reads for a given (subject) read being compared to the rest, then the subject read forms a unique haplotype. If there are at least 3 different haplotypes within one region for any single person, then the region is considered confetti. This is well correlated with the visual inspection of the same BAM file views as empirically studied by the developers of this code.

**Purpose**

Documentation and rationale of the filters for evaluating potential de novos left by the Salvage Pathway that are most likely not true de novo variants by the recognition of artifacts in their position in BAM file regions as artifacts that generate false genotype calls.

**Background/Introduction**

The problem that prompted the creation of this “confetti filter” comes from the decision to keep unsalvaged de novos in the varSifter files after they passed through the Salvage Pathway and through the KaylaKode. This was done in order to be conservative, and to not accidentally discount de novos that were not salvaged, but also were in fact true de novos. The issue that arose in being conservative in this manner was the finding that many of these regions are of poor quality in their BAM files ("confetti regions"). While trained human analysts can easily identify a confetti region by sight, it is much harder for a computer to do so, especially without use of explicit image processing techniques. Therefore, the question becomes: How can we mimic the visually processed signs that trained human analysts use to identify confetti regions in a BAM file in a Java-based program, i.e., Pseudo-image processing?

A de novo variant is defined as a genetic change that occurs for the first time in a family member (e.g., the proband) due to a mutation in a parent’s germ cell or a variant that occurred in the zygote during embryogenesis. Such variants are inherently Mendelian inconsistent since the proband did not inherit the de novo variant from any grandparental chromosome seen by sequencing the parents’ blood (somatic) genome. However, a genotyping process that is not pedigree-aware is strongly biased to call the reference allele when a parent has low coverage. Consequently, many variants that originally appear de novo in the raw clinical exome or genome can be corrected by re-genotyping using the Salvage Pathway when parents and siblings are considered together along with ethnic-specific prior probabilities and careful interrogation of different types of noise in the sequencing chemistry and alignment. Because the Salvage Pathway only attempts to automatically re-genotype Mendelian inconsistent variants, it can spend more computational time and effort on these few thousand places and efficiently perform a task that would make a generalized genotyper inefficient.

False positive variants, originally categorized as de novo (before re-genotyping) were sometimes kept as de novo because of a desire to be conservative about risking the exclusion of any true de novo variant. However, many of these apparent de novo genotype calls are actually noise. This makes the task a matter of separating true de novos in the remaining list of apparent de novos from false positives. To perform this selection, a series of noise filters with criteria described below were applied to those candidate de novos, after the ROC filtering stage.

This is the 6th step of the full Forwards Backwards analysis pipeline. Ideally, this would be run after the MakeBamROC filtering stage.

For the sake of space, this filter is called the “**confetti filter**” in these manuals. This is because the noisy areas in false positive de novos appear as if someone threw a handful of confetti over the pileup, resulting in many colorful bits/variants in the pileup when viewed by the Integrated Genome Viewer.

**Abbreviations**

**VOI – variant position of interest.** The place in each short read that, when aligned in a short read pileup, corresponds to one base position in the human reference genome. This is the position that was identified in the genotyped variant call file and passed the KaylaKode filters as a potential variant solution. Whether this is a real variant or an artifact of the alignment process is the question this module tries to discern from the analysis of short reads.

**Confetti regions** – regions in exome/genome alignments that, when viewed by a genome viewer such as the Broad Institute’s IGV (while in the show variant bases only display mode), visually appears to display a grey background onto which is sprinkled a random pattern of colored pixels that correspond to variant base calls within the short-read sequences that are shown in a small interval of the reference genome positions. This pattern is roughly the equivalent to what would be seen if a freshly painted grey surface had colored confetti thrown at it; because of this analogy the term “confetti region” was applied. The most likely reason for this type of pattern is a combination of mis-mapping, low total coverage and artifacts of the sequencing chemistry (PCR artifacts, fluorescence detection errors, sequencing signal degradation that occurs at the ends of short read sequencing by synthesis methods, etc.) An example is appended to the end of this section.

**Ghost regions** – These are regions in an exome/genome alignment that, when viewed by a genome viewer such as the Broad Institutes IGV (while in the show variant bases only display mode), visually look like the superposition of a normal region and a very faded or faint impression of a second pattern that has several reads with aligned variant bases, far below the number typically seen when a true diploid genotype is present. This pattern is not one of random speckle (see confetti region above); it appears to be a faded superposition of a second region of short reads that are only a few percent of the total (from 1-15%) number of short reads in the region, but appear to have the same variants as each other. Also, these variants are a minor percentage of the total number of bases in the local region of the reference genome (no more than one per 5 bases). The reason for this is most likely the mis-mapping of reads that should be correctly mapped to a different region of the reference genome that is very similar to the general region into which these reads were placed by any of the “non-exhaustive” alignment procedures that are the only alignment programs practical for genomes as large as the human genome. An example is appended to the end of this section.

**altMap** and **refMap** – both are key/value(s) hash maps made from the (modified – see below “*Both - haplotype counting*”) read strings and the start and stop positions for where each read was mapped to a reference genome.

**readString** – standard variable name in alignment metadata that refers to a single letter base call in the order of bases in a short-read sequence that was returned from a Sequencing by Synthesis type of DNA sequencing machine. The most common current type of machine is the Illumina NextGen sequencing technology.

**posVarStart** and **posVarEnd** - values in a hash map that locate the start and stop positions for a variant. There is a requirement for two values to handle insertions and deletions in the short-read relative to reference genome coordinates for aligned reads containing more than a single base change (i.e., a SNP variant).

**Program Overview**

***General***

**Relevant subroutine(s):**

* readDepthEval
* extremePopNovel
* genoCoverageEval

**Overview**:

There are three main criteria identified that were stringently applied during filtering and were based on sequencing quality as well as population metadata. These are: read depth; extreme novelty; and genotype coverage. Threshold parameters were developed heuristically, and they function adequately to accurately distinguish true signal from noise. These values, based on empirical testing with at least 10 proband/sibs’ ROC outputs, are prime targets for further optimization. The heuristic but functional parameter choice also applies for the confetti region filters described below.

*Read Depth* (readDepthEval)

In next-generation sequencing, DNA is fragmented into multiple pieces, which are then each amplified, sequenced, and then algorithmically mapped and aligned to a reference sequence. These aligned sequence fragments (“reads”) are stored in a Sequence Alignment Map (SAM) format, a text-based format that also includes metadata about each read such as mapping quality. The compressed, binary configuration of a SAM file is called a Binary Alignment Map (BAM) file. The more deeply sequenced a region of the genome is, the greater the number of independent overlapping reads will be in that section of the SAM/BAM file under any single position in the reference (i.e., a deeper pile up).

Genotype calling is based on Bayesian probability, with an assumed prior probability that is often an average value estimated population variant allele frequency, and a conditional probability derived from the BAM file using the number of reads that were sequenced at that position—the read depth, but only if these reads exceed a base-call quality at that position.

If there are too few reads in the region of interest, the likelihood of a correct genotype call decreases. This is a sampling issue, that asks how accurately the reads for a section of the genome represent the actual genotype that was present in the subject’s DNA used to generate the sequence pileup. The more reads one has at a given position, the greater the power there is for accurately calling the genotype of that position if there is only random noise. One of the heuristic rules applied states that if the read depth is below 8 reads under any genomic position, then the overall region is considered poor in read depth and any de novo genotype made at that location is excluded from the final gene list. Systematic errors do not improve with read depth, mainly due to mis-mapping in next-generation sequencing in low-entropy regions of the reference sequence (i.e., highly repetitive sections of the reference genome).

*Extreme Novelty* (extremePopNovel)

For de novos from patients with rare/undiagnosed diseases, it is relatively safe to presume these events are very uncommon (i.e., novel relative to a large population base). It can be further presumed to be more confidently novel the larger the number of other sequencing experiments that reliably sequenced that same position and called those other genotypes homref. The state of a novel event in a location that is reliably well sequenced as homref is designated an extreme novel case. This means that *very few* people—essentially no one else—in the population have these same variants in their genome. Each VarSifter file in the current work contains additional metadata about the all Gahl and ClinSeq (CS) cohorts, including information on the number of people in each cohort who were homozygous at a given variant of interest (VOI) position. Since these databases were from individuals whose DNA samples were prepared in the same lab with the same chemistry and sequenced on the same machines in the same center, they constitute an excellent control group to designate a novel variant as an extreme novel event. If there are any individuals who are homozygous for the variant at the VOI position (looking at specifically the “All\_Gahlc\_homvar” (if All\_Gahlref\_isminor = 0), “All\_Gahlc\_homref” (if All\_Gahlref\_isminor = 1), “hg19\_CS\_938c\_homvar” metadata), then their parents must be hets and also possess a copy of the VOI. This violates our assumption of the VOI being an extremely novel de novo and is another of the heuristic rules to exclude a false positive de novo. While the rationale is subtle, the presence of sufficient parents that two heterozygous individuals were found to have mated and produced a homozygous variant offspring, suggests that a suspected de novo is more likely to be due to a false negative heterozygous parent (given the suggested high population frequency that yield a heterozygous mating in the control cohort), relative to the intrinsic new mutation rate which is 1.18 X 10e-8 per site per generation.

*Genotype Coverage in the Population* (genoCoverageEval)

Each VS file contains additional metadata from the all Gahl cohort (n = 1310) and the CS cohort (n = 938), including information on how many people in each cohort were sequenced at the relevant VOI position. The more people that had a confirmed genotype at this position, the more confident we can be about the variant’s extreme novelty. This is an inverse Poisson problem.

If X = # ppl with het at the VOI position, k = variable, λ = sample size (n)

P(X>=k) = e-k/k! = c

1 - c = P\* (the probability that we see no one else with hets).

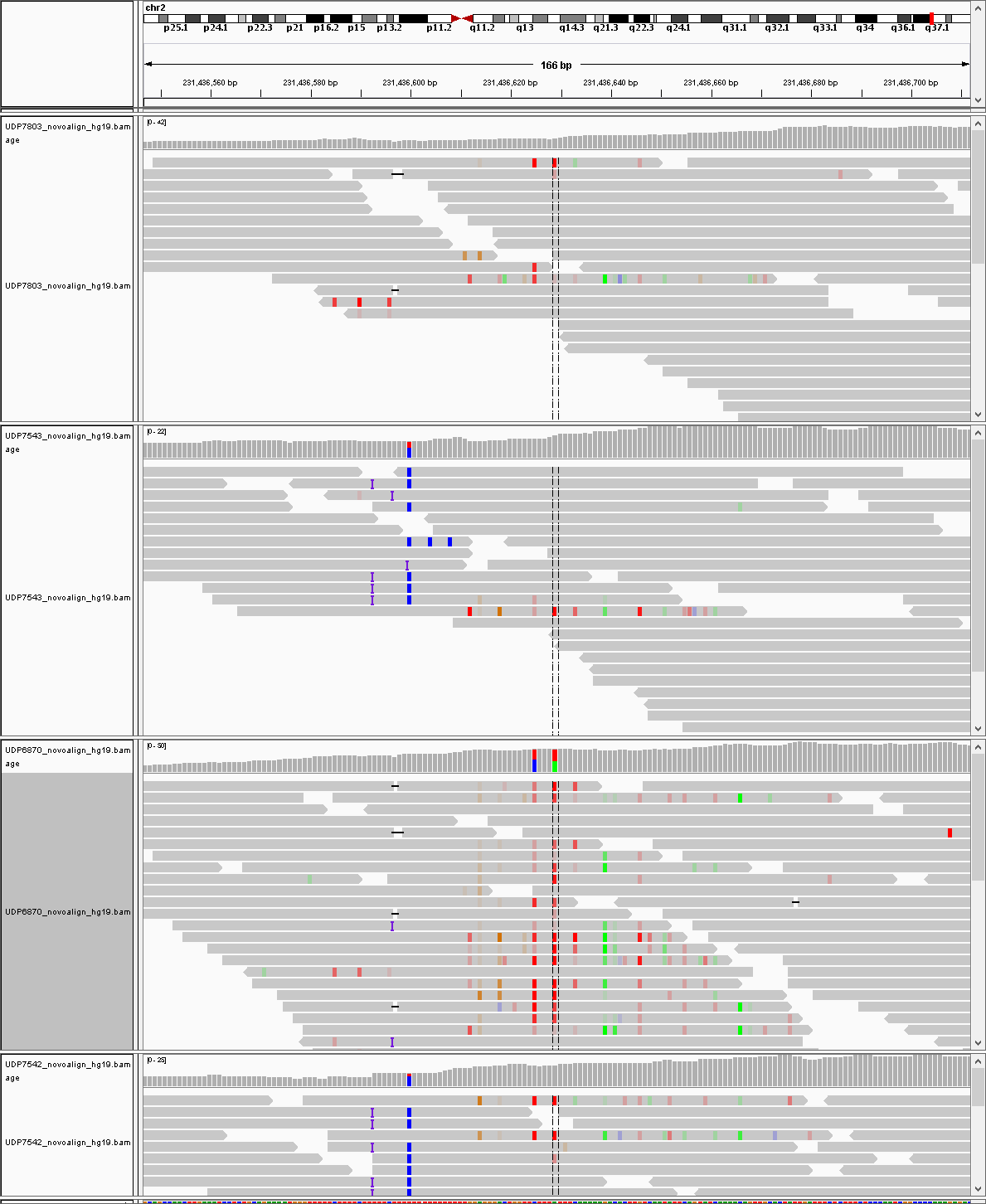
The smaller the sample size (n), the smaller P\* will be.

More intuitively speaking, if only half the cohort (nsub,1 = 655) was genotyped at the VOI position in their respective genomes, it is still possible that in the rest of the cohort (nsub,2 = 655) that didn’t get genotyped there may be homvars or hets at for the VOI. Therefore, the greater the number of people in the all Gahl and CS cohorts who were unsequenced at the VOI position, the lower the confidence that the variant is a true extremely novel de novo, especially if either parent has a particularly low read depth at that VOI position.

***Confetti-region specific***

*Background:*

Certain regions of the genome are more difficult to sequence than average. These regions often result in mismapping and misalignments in any one BAM file pileup. This can be readily visualized by looking at BAM file pileups using programs like the Integrative Genome Viewer (IGV) developed by the Broad Institute. Typical mismapped regions look like someone took a paint gun or a handful of confetti and sprayed it all over the region, hence the colloquial designation used by our analysts: “confetti regions.” In almost every case it is easy for a minimally trained human analyst to visually distinguish confetti regions from cleanly sequenced regions. However, it is far more difficult to computational systems to do so. It was found to be necessary to automate the process of detecting confetti regions within a variant list, not only because it is more time- and manpower-efficient, but also because it removes any chance of human bias in identifying an area of poor sequencing quality. This is critical to the logical assertion of blindness with respect to which group is the true affected group and which group is the unaffected siblings for the forwards/backwards analysis.



*An example of a confetti region.*

**Relevant subroutine(s)**:

* horizontalPPDN
* verticalPPDN
* haplotypeCount

**Overview:**

There are three subroutines and methods that identify a confetti region: horizontal, vertical, and bidirectional.

*Horizontal* (horizontalPPDN)

According to the SAM/BAM Format Specification Working Group, the mapping quality (MAPQ) value is equal to -10log10Pr{mapping position is wrong}, rounded to the closest integer [1]. So, for example, if there is a 50% likelihood the mapping position was incorrect, then the MAPQ value should be 0. Similarly, if the MAPQ value is 30, then the probability that the mapping position was incorrect is 0.1%. Thus, the higher the MAPQ value, the less likely the mapping position of the read in the alignment was incorrectly positioned, with each increase in increments of 10 in the MAPQ value corresponding to a tenfold decrease in the probability of mismapping.

Similarly, a base quality value (QUAL) is the phred-scaled probability a base call is wrong –-10log10Pr{base is wrong} and is used to evaluate the reliability of a base call made by the sequencer after the aligner positions all the bases in the read onto the reference. Base errors are typically due to errors in sequencing chemistry. In the SAM/BAM format, the Phred quality scores are scaled by adding 33 to the calculated Phred quality so that the value can be read in ASCII format. In other words, a Phred quality score range from 0 to 93 will use ASCII 33 (“!,” the lowest printable ASCII character value) to 126 (“~”, the highest printable ASCII character value).

The "horizontal" method involves evaluating the base-call quality array and overall map quality score for all the reads that contain the VOI position within a given BAM file. The CIGAR string for each read, along with the starting and ending positions, are extracted from the metadata that the aligner added to the header for each read queried by the SAMTools subroutine. If the VOI position is inside the read’s start and end positions, and if the sequence isn’t empty (CIGAR isn’t “\*”), then the program extracts the read’s mapping quality (MAPQ) and constructs an array of base qualities for every base in the query read.

An individual read will be stored as a poor quality read if the following heuristic rules apply:

* The number of poor quality bases in the read > 10
  + A poor quality base was defined as a base for which QUAL < 20, Pr{base call is incorrect} > 5.0%.
    - If we assume each base’s call is independent, then having at least 10 bases with QUAL scores of <20 each gives a probability of <0.0510, or <9.8E-14. It is extremely likely in this case that the overall read is poor quality. (This initial assumption about independence among bases is actually unrealistic due to things like alignment errors, but we make this assumption for simplicity’s sake. Also, given that the actual probability of a poor QUAL score for a given base is actually dependent on its neighbors, then if there is a string of poorly called bases in a read, it is likely that the read overall is of poor sequencing quality.)
* MAPQ < 30, Pr{map position is incorrect} > 0.1%
* Or the total number of bases is less than 10.
  + This is an issue of extreme soft clipping.

If 25% or more of the reads in the queried region are of poor quality, then the region is considered confetti.

*Vertical* (verticalPPDN)

A de novo variant in the proband should not be found in the proband’s parents and unaffected siblings. The "vertical" method involves evaluating each read in a person’s BAM file at the VOI position. The reference and variant genotypes are determined by the variant metadata in the VarSifter file (“ref\_allele” and “var\_allele” values, respectively). The number of times a variant is called at that position is then compared among family members. The following heuristic rules apply: If each family member has ≥2 variant calls at the VOI position, then the definition of a de novo variant is violated, and this so-called variant cannot be a true de novo. The same is true if any other member of the family besides the proband has multiple (≥3) variant calls.

*“Bidirectional” - haplotype counting* (haplotypeCount)

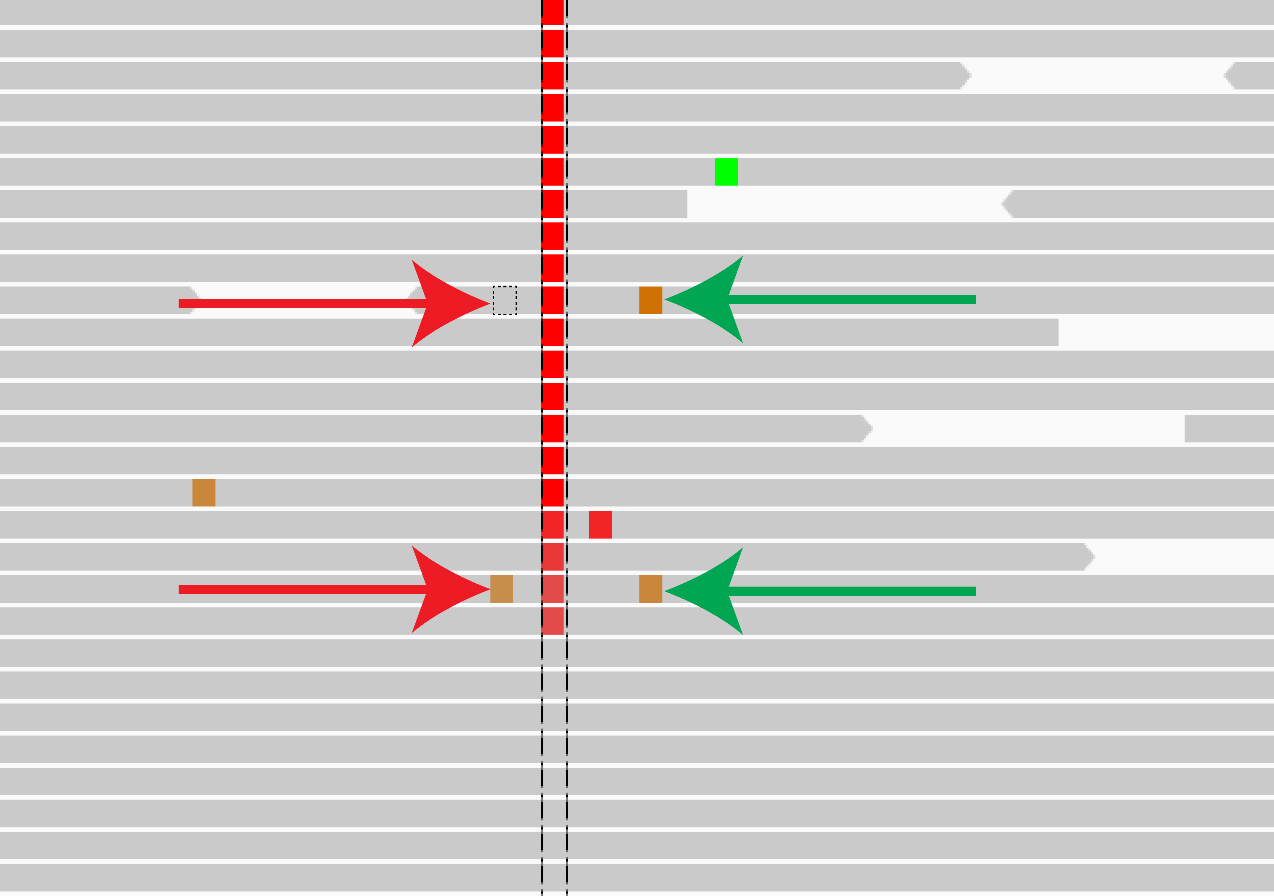
A haplotype is a group of DNA polymorphisms that are inherited together from one parent. This is any region of the parental chromosome segregated into the gamete that has not had a crossing over. This can refer to several different groupings of genes, from half the genes on a chromosome inherited together from one parent, to a cluster of single nucleotide polymorphisms (SNPs) that were inherited together in a very small interval. Haplotypes occur due to genetic linkage, in which genes close to each other on the same chromosome are more likely to be passed down together for many generations without any crossing over occurring inside the interval.

Because each individual has two parents, there are expected to be (almost exclusively) at most two haplotypes in any given very small region. We also assume that there are no issues with polyploidy (which for large regions should have been identified with previous karyotyping or SNP chip analysis) or somatic post zygote mosaicism. If there are more than two apparent haplotypes in a very small region, then it is very likely that there are reads mis-mapped/misaligned, which can signal a confetti region. Mis-mapping occurs when a read is mapped to the wrong region in the reference genome. This can be due to several factors, like mononucleotide repeat tracts and areas of low complexity. Other misalignment errors include PCR artifacts due to errors during amplification and sequencing errors due to physical errors in the sequencer and/or properties of the DNA sequence itself (i.e., homopolymeric regions).

The program first extracts the Compact Idiopathic Gapped Alignment Record (CIGAR) string for each read along with the start and end positions and sequence (SEQ, readString) in order to store each read in one of two maps. If the base at the VOI position is ref, then the read will be put in the reference map (refMap). Otherwise, the read will be put in the variant map (altMap). The keys for both maps are the readStrings and the value(s) are the VOI position's coordinates within the readString ([posVarStart, posVarEnd]). This is complex because of the potential of soft clipping and/or indels within the raw read sequence. To mark where indels have occurred in the sequence, the readString has to be padded with "D"'s for deletions and "I"'s for insertions. These dummy designations do not change the readString sequence overall. Soft clipped regions of the sequence are also removed from the readString. It is possible to infer a risk of a large indel from excess numbers of reads with aligned soft clip start positions, but this is not yet implemented in the heuristic rules for calling a region “confetti”.

Once all the reads have been binned into their respective maps (refMap or altMap), then all reads are compared with one another for each map. The program first compares two reads at the VOI position. If they match, then the bases at the preceding position are compared, and this continues until a difference between the two sequences is found, 50 bases have been read, or until the end of one sequence is reached. The comparison then moves in the proceeding (forwards) direction until the same stopping criteria are met (difference, 50 bases, or end of either read being compared). If at any point the two reads differ from each other, the calls at those positions for each read are stored and associated with those reads (PutativeVarCall class). Additionally, if the reads overlap for at least ±15 bases of the VOI position (an overall frame of 30 bases) and they have no differences, then they are considered part of the same potential haploblock. This continues until all reads have been compared with each other within the region and grouped into haploblocks. Next the following heuristic rules are applied: If the size of the potential haploblock is at least 20% of the overall pileup, then the potential haploblock is considered an apparent haploblock. If there are more than 2 apparent haploblocks within the pileup, then the overall pileup violates the concept of diploidy for each chromosome and is considered noise.

If one read differs from every other read (there are no other agreeing reads), then it is possible that the former is just a badly aligned read and the others form their own haplotype. If there are too many of those "one-off-reads" (more than 10 one-off-reads per (at least) 20% of the overall pileup), then the overall pileup is considered too dense with spurious (one-off-read) noise.



***An example of multiple apparent haplotypes****. Though the bases marked by the green arrows appear to match, the bases on those same reads marked by the red arrows are different. Thus, the reads marked by the arrows represent two separate apparent haplotypes (Ref – G in the first read and G – G in the second read). Combined with the other two main haplotypes apparent (Ref – Variant of interest – Ref, Ref – Ref – Ref) and the other three spuriously erroneous variant calls in the other reads, this image represents at least 7 different haplotypes.*

1. **Extreme Novel Exon Deletion Filter (Call No Call Filter) Module**

**Executive Summary**

This module detects, incompletely but reliably, regions where one person in the large local cohort had a variant and yet only the proband was missing all coverage in her/his BAM file. This is given the label CNC for Called variant (in someone else) but with No Coverage (in the proband). Only genome positions where someone has a variant are included in a jointly called variant file. However, since all the other individuals in the jointly called file also have been genotyped at all those positions, it is possible to ask: Are there any of those positions where only the proband had zero reads aligned at that called variant position? If there is a place like that for the proband, is the local region really missing expected reads, or is this an artifact of a region with very low coverage in most people but the proband was the only one so unlucky that there were actually no reads at that position? However, while another individual in the jointly called file has deep enough coverage for the genotyper to make the call, it is still to be determined whether that position is truly well sequenced in most others, based on the experimental sequencing conditions (i.e., DNA extraction, DNA capture enrichment of exons, DNA sequencing chemistry and fluorescent detection, mapping and alignment). These candidate CNC positions can be filtered by examining the BAM file of the proband and her/his nuclear pedigree family members in the general region 100bp on either side of the candidate position and comparing the proband’s local read count to the count of the rest of the family members. This filter also determines if the person with the next fewest reads at the called variant where the proband has zero reads is a person with only one or a handful of reads, or if the next least coverage is very deep. These criteria are used to confidently include or exclude a potential CNC finding in the final candidate list.

**Purpose**

"Call no call" variants require an additional level of filtering, in part because they are not subject to the same population and subpopulation filters as the other inheritance models used the BamROC step. CNCFilter.java specifically deals with variants that have been called "CNC" in previous steps.

**Premise**

This is the 7th and final step of the full Forwards Backwards analysis pipeline. "Call no call" (CNC) variants are variants that supposedly represent a dropped/deleted exon in only the proband's genome. The way we initially identify CNC's is that the proband has no coverage in the candidate region whereas everyone else in the cohort does. In other words, a CNC is an extreme novel deleted exon. The issue with identifying true CNC's is that: a) They require looking at multiple people's bam file coverage; and b) They require multiple CNC variants to be called near each other (a Poisson problem).

**Program overview**

The primary initial identifier for a candidate CNC is if, at the MakeBamROC step, the Mendelian model flag ("mendFlags") was labeled as "CNC" or, at the KaylaKode, if the mendInconsis value is 4. (These should be equivalent, since the mendFlag annotation was based on the mendInconsis value.) If the variant is not a potential CNC, then its information is written to the output file.

After identifying potential CNC variants, the next step is to see how many of those variants fall in the same region of the genome (in other words, how large a cluster of CNC's is). If an exon were truly deleted in a proband's genome, then multiple CNC variants should cluster by position in the same region, presuming that other variants are seen in someone else in a large cohort nearby to the first one. For a jointly called cohort of around 1500 exomes, this will be very likely and there should be a handful of variants found in most of the exons in coding genes. The program first groups ID'd CNC variants by chromosome. Then, it sorts those variants by index, assuming that index corresponds with position in the chromosome. Afterwards, the program iterates through each variant, asking if two adjacent variants are less than or equal to 250 bp apart. If they are not, then the program groups them together and evaluates the next variant. If they are more than 250 bp apart, then the program creates a group, resets its counters, and evaluates the next variant pair for that chromosome. Again, if there is only one variant in a cluster, then it is considered the result of a poor sequencing job and is excluded from further analysis.

After identifying clusters of CNC's, the program sorts the variants in each cluster by index. It then takes the starting position of the first CNC variant and the ending position of the last CNC variant for the sorted cluster. This is to establish the boundaries of the regions subsequently analyzed in each person's BAM file.

Once the boundaries for analysis are identified, the BAM files of each family member (excluding the proband) are called. The reason why the family members are being analyzed is that they should share the most genes with the proband, compared with any other individual (assuming no issues with non-paternity or uniparental disomy). If the candidate deleted regions in the proband are poorly sequenced in the family members (i.e., poor read depth), then it becomes more likely that the lack of coverage in the proband in that region is a consequence of poor overall sequencing in that section of the genome (false positive CNC). Therefore, the program evaluates the read depth at every position within the previously established boundaries of the genome in each family member's BAM file. Each read is identified by its QNAME in the BAM file and stored once for each position. An array of the number of unique reads at each position is generated. If there are multiple (say, 5+) spots in the region of interest where there are fewer than 5 reads, then the "variant" is marked as a false positive CNC.

An additional filter is applied to the CNC's that make it past this first level of filtering. This filter looks at the regions immediately flanking the originally called bounds of a CNC variant cluster, which were determined earlier. Where the earlier filter looked at the "depth" at a single position, this filter looks at the "breadth" of reads (or rather, the lack thereof) across multiple positions. This range is determined by looking at both parents’ BAM files. The program identifies reads in the parents that overlap with the previously identified CNC cluster boundaries. After taking the start and end positions of each of those parents’ reads, the program then defines a flanking region range with the start being the minimum read start position and the end being the maximum read end position. If a proband has any reads (> 0) within this region, then the variant is considered a false positive.

If the variant makes it past both these filters, then its information is written to the output file.

**References**

1. Consortium GP. An integrated map of genetic variation from 1,092 human genomes. *Nature.* 2012;491(7422):56-65.
2. Li H, Handsaker B, Wysoker A, et al. The sequence alignment/map format and SAMtools. *Bioinformatics.* 2009;25(16):2078-2079.
3. Kircher M, Witten DM, Jain P, O'roak BJ, Cooper GM, Shendure J. A general framework for estimating the relative pathogenicity of human genetic variants. *Nature genetics.* 2014;46(3):310-315.
4. Lek M, Karczewski KJ, Minikel EV, et al. Analysis of protein-coding genetic variation in 60,706 humans. *Nature.* 2016;536(7616):285-291.
5. Gahl WA, Mulvihill JJ, Toro C, et al. The NIH Undiagnosed Diseases Program and Network: applications to modern medicine. *Molecular genetics and metabolism.* 2016;117(4):393-400.
6. Consortium UK. The UK10K project identifies rare variants in health and disease. *Nature.* 2015;526(7571):82-90.
7. Biesecker LG, Mullikin JC, Facio FM, et al. The ClinSeq Project: piloting large-scale genome sequencing for research in genomic medicine. *Genome research.* 2009.
8. Teer JK, Green ED, Mullikin JC, Biesecker LG. VarSifter: visualizing and analyzing exome-scale sequence variation data on a desktop computer. *Bioinformatics.* 2011;28(4):599-600.
9. Cingolani P, Platts A, Wang LL, et al. A program for annotating and predicting the effects of single nucleotide polymorphisms, SnpEff: SNPs in the genome of Drosophila melanogaster strain w1118; iso-2; iso-3. *Fly.* 2012;6(2):80-92.
10. Speir ML, Zweig AS, Rosenbloom KR, et al. The UCSC genome browser database: 2016 update. *Nucleic acids research.* 2015;44(D1):D717-D725.