VIA™ Software Theory of Operations

DOCUMENT NUMBER:
CG-00042

DOCUMENT REVISION:
A

Effective Date:
07/30/2023

bionano

Contents

Legal Notice

For Research Use Only. Not for use in diagnostic procedures.

Revision History

REVISION	NOTES
A	Initial document release.

Introduction

Variant Intelligence Applications™ (VIA) software is a complete and integrated solution for the visualization, interpretation and reporting of genomic variants from multiple technology types. By supporting multiple genome-wide data modalities, VIA software provides the most comprehensive view of genomic variants of any interpretation, annotation, and reporting software tool available. As a platform-agnostic tertiary analysis solution, VIA stores and manages distinct types of genomic data from various platforms (see **Table 1**) enabling the extraction of meaningful insights from a combined analysis. The software includes algorithms to detect copy number variants (CNV) from major microarray vendors, optical genome mapping (OGM), and next generation sequencing (NGS) methodologies as well as Absence of Heterozygosity (AOH), from data types that assess B-allele frequency. VIA also provides intelligent interpretation assistance to analyze CNVs, Loss of Heterozygosity (LOH) and Structural Variants (SV) from OGM data. As a centralized analysis solution spanning technologies and application areas, VIA software provides an efficient environment to keep pace with advancements in technology while retaining access to historical platform data. By being adaptive to whichever technology is used to generate CNV, LOH, or SV genomic variants, VIA software provides rich annotations for the co-analysis of sequence variants from NGS to provide a complete picture of genomic variation and reveal more answers for disease association.

Table 1. Common platforms supported in the software.

PLATFORM	EXAMPLE ASSAYS	ASSOCIATED FILE TYPES
BIONANO	OGM	ogm.bam
		ogm.vcf
THERMO FISHER/ AFFYMETRIX	Affymetrix arrays output .cel file format	.cel
	CytoScan 750K	.cychp
	CytoScan HD	.cyhd.cychp
	CytoScan XON	.xnchp
	OncoScan	.oschp
	CytoScan HT-CMA, SNP6	.cel
ILLUMINA	CytoSNP12, CytoSNP850K, Infinium Omni, GSA, GSA-Cyto, GDA, GDA-Cyto	.txt (Final Report files)
		.gtc
	Infinium HumanMethylation450,MethylationEPIC	.idat
AGILENT	SurePrint G3 CGH + SNP Bundle, 4x180K GenetiSure Cyto 4 x 180K CGH+SNP	.txt
NGS	WGS, WES, Panels	CNV = .bam
		Seq Var = .vcf, .vcf.gz .json.gz (generated by Nirvana)
CUSTOM	Custom CNV with probe or segment values	.txt (tab delimited)
	Custom Seq Var with annotations	.vcf

The principles and algorithms applied across platforms in VIA software are shared and the flexible parameterization capabilities of the software enable customization to address the nuances of each technology. This document provides a description of the workflows and algorithms fundamental to data processing in VIA with a summary of the software's performance for example datasets.

OGM data from a VCF or a BAM file can be uploaded into VIA from Access or from the VIA homepage. The data is processed using the settings determined in the sample type, which may include application of a decision tree, which assigns pre-classifications to the variants detected in the sample, depending on the overlap or similarity with the coordinates of known regions involved in pathogenicity or other regions of interest determined by the user. Users can visualize and re-classify variants, review links to external databases, write detailed variant interpretation text and export these into report templates. These data are stored in the VIA Server and are accessible for review though the UI by other authorized users.

VIA CNV & AOH Segmentation Algorithms

Copy Number Segmentation Concepts

VIA software will arrange the ratios according to their position along the chromosome. Each probe is represented as a small gray dot along the length of a chromosome in the genome and chromosome plots; the user-specified calling thresholds seen as blue and red horizontal lines, call certain regions as a **Gain**, **Loss**, **Amplification**, or **Homozygous Loss**. The most trivial calling algorithm would be to simply use these thresholds and the probe locations, but this can cause noisy output, marking any probe that exceeds these limits as a CN change event. Different approaches can be utilized, and VIA offers three segmentation algorithms: a circular binary segmentation (CBS)-based, SNP Rank, and two iterations of a hidden Markov model (HMM)-based algorithm, SNP-FASST2 and a new algorithm SNP-FASST3.

SNP-FASST SEGMENTATION

Intensity and BAF information are used in tandem to segment the genome and associate the most likely statehood for the segment. Single Nucleotide Polymorphism Fast Adaptive State Segmentation Technology (SNPFASST) defines states that represent all *possible* combinations of LogR and BAF states in a matrix of ninety-six combinations of copy number and allelic changes.

Effective for detection of mosaic and multi-clonal cases, these probability calculations occur throughout the genome based on the user defined minimum probe floating window. Generally, the higher the number of probes to create a region, the more confidence that the copy number change is real. SNP-FASST is based on the Hidden Markov Model, but the defined states are Normal, Gain, Amplification, Loss, and Homozygous Loss as seen in **Figure 1**. The state boundaries are adaptive, meaning the user can adjust the thresholds (blue/red lines) so that the algorithm adheres to the data.

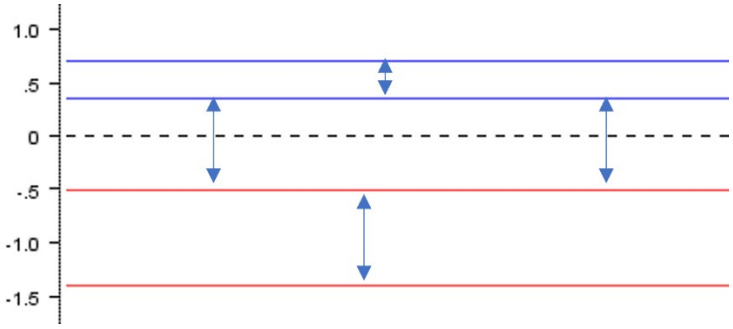


Figure 1. CN state threshold setting for SNPFASST region calling.

SNP-FASST2 Algorithm Parameterization

Parameterization of the segmentation enables flexibility to achieve desired sensitivity and adapt to technology nuances, which are set within the VIA administer panel. The key parameters impacting segmentation for copy number events are:

- Copy number state thresholds for each CN event type, and corresponding settings for autosomes and sex chromosomes wherein the tighter thresholds will increase sensitivity.
- Significance Threshold is a critical value for SNPFASST2 segmentation as functionality and influence on region calling. Increasing the stringency of the p-value to segment the genome will enhance CNV and AOH calling from data patterns with higher confidence.
- The minimum number of probes to make a CN segment.
- Maximum Contiguous Probe Spacing (Kbp) is a parameter used to limit the segments such that if there are any two neighboring probes that are separated from each other by more than the specified distance (in Kilo base-pairs), the segmentation algorithm will stop at the last probe location. This will result in no calls being made in these areas. A good example would be the centromere loci.

This algorithm identifies trends in the B-Allele Frequency for copy neutral events and is highly informative in identifying long contiguous stretches of homozygosity (LCSH), iso uniparental disomy (iUPD), chimerism, or maternal cell contamination (MCC). **Figure 2** illustrates these unique features as described below.

- **Minimum LOH length** - defines the size of the contiguous AOH; it is a key SNP calling parameter.
- **Homozygous Threshold** – defines a homozygous state for the SNPs between the yellow and black lines.
- **Heterozygous Threshold** – defines an SNP between the purple and yellow lines and adjusts sensitivity for calling AI.

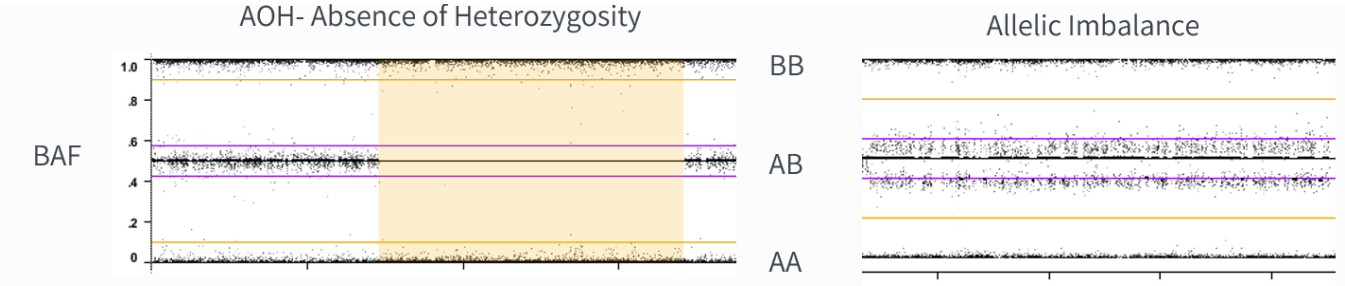


Figure 2. SNP- Allelic Imbalance/AOH Detection

SNP-FASST3 SEGMENTATION

SNP-FASST3 segmentation algorithms build on the previous version, SNP FASST2, and offers improved segmentation particularly for mosaic events. The main enhancements are:

- Support for mosaic calling.
- Improved calling of events when probes fall on the one-copy loss or one-copy gain threshold lines.
- Integration of BAF and Log R for copy number calling with SNP arrays (for SNP-FASST3)

This is achieved through the addition of mosaic states with the processing settings and a modified probability curve for optimal region segmentation. The distribution used, Plateau Pseudo Distribution (PPD), is a modified normal distribution with a plateau in the center, as seen in **Figure 3**.

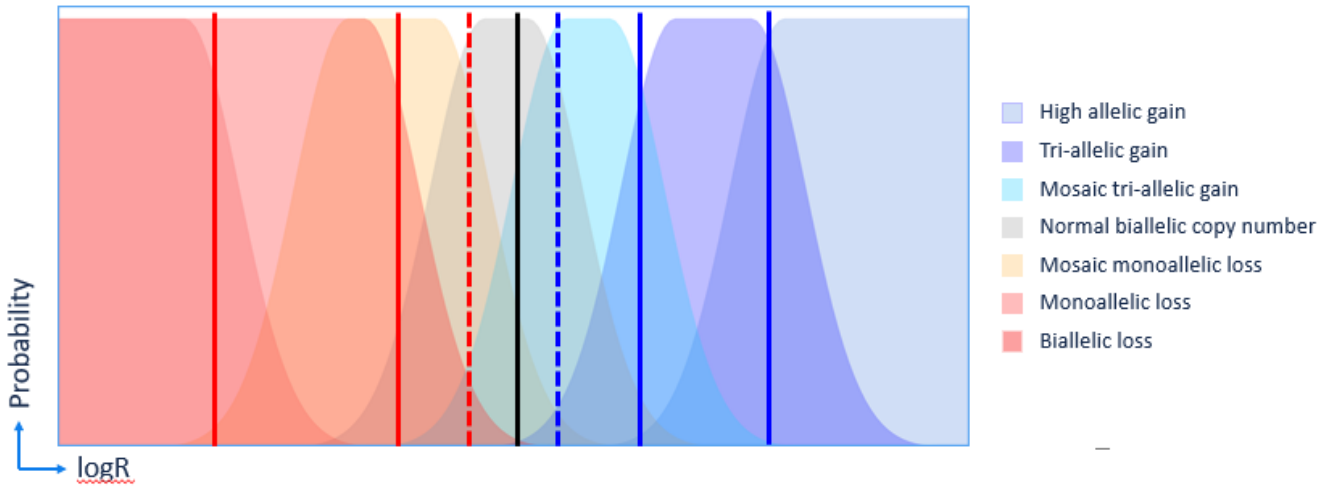


Figure 3. An example Plateau Pseudo Distribution (PPD) plot for loss states.

The addition of mosaic states to the Log R and BAF states, which represent a state between normal and adjacent states allows use of a single settings profile to call copy gains and losses as well as mosaic gains and losses. Entering these mosaic states requires orders of magnitude more significant than entering non-mosaic states (e.g., if the significance threshold for non-mosaic is set to 1E-8 then, by default, the threshold for the mosaic state would be 1E-20. This offset can be modified by the user in the Processing settings).

The algorithm also fills gaps between the loss states and the gain states such that the probability distribution function for homozygous loss/big loss, loss, and mosaic loss (and similarly for high gain, amplification/gain, and mosaic gain) will not have dips between the distributions. This improves calling when probes fall along the threshold lines where these dips were previously located.

In addition, SNP-FASST3 incorporates Log R and BAF data together for single-copy loss and mosaic copy loss states to improve calling, where previously only Log R values were used to determine copy number. This makes the algorithm less likely, for instance, to make a single-copy loss call in a region where the BAF track looks heterozygous. Values for gain and loss are 1/6 of the one-copy gain/loss thresholds respectively and for mosaic allelic imbalance, it is (2.5+allelic imbalance threshold)/6.

Several parameters that affect the behavior of mosaic calling can be adjusted by the admin in Processing settings for SNP-FASST3 algorithm:

- **Minimum mosaic threshold (%)** – level of mosaicism to be detected, corresponding to the desired aberrant cell fraction.
- **Mosaic CN significance offset** – amount by which to increase stringency of the significance threshold for mosaic CN states.
- **Mosaic SNP significance offset** – amount by which to increase stringency of the significance threshold for the mosaic imbalance state.

SNP-FASST3 also decouples the significance threshold, which is the most impactful processing setting, for CN and AOH segmentation to allow for additional parametrization to achieve desired sensitivity for segmenting CN and AOH independently.

RANK AND SNP RANK SEGMENTATION ALGORITHM

The Rank Segmentation algorithm is a statistically based algorithm, similar in concept to the Circular Binary Segmentation (CBS) algorithm developed by Adam Olshen at Sloan-Kettering Institute (Olshen AB, Venkatraman ES, Lucito R, Wigler M. Circular binary segmentation for the analysis of array-based DNA copy number data. Biostatistics. 2004 Oct; 5(4):557-72). The CBS algorithm has been modified to significantly improve processing speed by using a normal distribution function to assess for change points as opposed to the non-parametric permutation-based statistics used in the original CBS algorithm. The result is segmentation of the genome into clusters of uniform ratios. A recursive algorithm, the genome is continuously divided into smaller and smaller units until no region can be further segmented. Significance Threshold is the single parameter that controls whether a region is to be segmented out or not. The logic is to start by rank ordering the log-ratio probe values, consider the distribution of the probe ranks in a region and then compare this information to the distribution of probes in the adjacent segment (to the left and right). If these distributions are significantly different, generating a significance value less than the Significance Threshold, then the segments are divided. The process stops if no segment can be found in an interval that is significantly different than its neighbors. At completion, the entire genome can be represented as a series of segments, each having a cluster value which is the median log-ratio value of all the probes in that region, plotted as horizontal black lines. The calling algorithm then uses the cluster values and the user-defined thresholds to establish regions of copy number variations. The SNP Rank Segmentation is like the Rank Segmentation algorithm but also considers B-Allele frequency values from SNP arrays to segment the genome. Using the B-allele frequency in conjunction with log ratios (providing copy number results) allows for better segmentation.

For the Rank, SNP Rank, FASST2, and SNP-FASST2 Segmentation algorithms, a significance threshold needs to be set in the Analysis panel so the sensitivity can be adjusted. The smaller the number, the less sensitive the algorithm is in creating a new segment. So, if some known aberrations are not being called because they are too small, this value should be increased. This setting is inversely proportional to the number of probes: the larger the number of probes, the smaller the value used for this setting, ensuring valid results. Many probes at a setting of 1E-6 or lower have been processed.

BAM MultiScale Reference concept for NGS

BAM MultiScale Reference (MSR) method functions well with both shallow and targeted sequencing data as well as WGS/WES with normal depth of coverage. It builds a reference file from a set of normal samples and uses adjustable dynamic binning. The method uses a Hidden Markov Model to segment the genome into target areas using the reads in targeted regions and the backbone areas using the off target reads and additional areas. Coarse binning is used in the backbone areas to provide the copy number baseline as well as large copy number events and fine binning is used in target areas to provide high resolution copy number detection in targeted regions. The adjustable dynamic binning is very flexible allowing adjustment of the minimum bin width based on the depth of coverage. The dynamic binning allows the target regions to get more coverage and the backbone regions, less coverage but the backbone still gets coverage. **Figure 4** illustrates the dynamic binning approach.

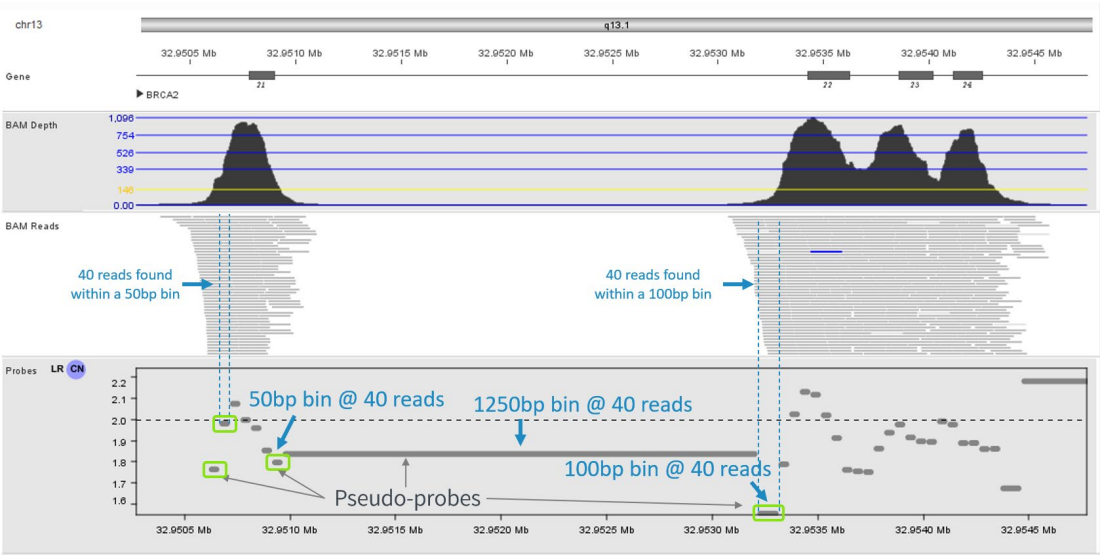


Figure 4. Bin formation using the BAM MSR Builder tool

BAM MSR method is a read-depth method that uses a pooled reference file to generate pseudo-log ratios based on the reads that is packages as a separate utility, the BAM MultiScale Reference Builder program. It also generates B-allele frequencies based on the reads at SNP locations. After logR bins are generated from the NGS methods, segmentation for CNV and AOH/LOH is performed using any of the segmentation algorithms available in VIA. Instructions on the parameterization of the BAM MSR is provided in the *VIA User Guide* (CG-00043).

SNP-FASST3 CNV PERFORMANCE FOR WGS (WHOLE GENOME SEQUENCING)

Copy number variant detection performance of SNP-FASST3 using whole genome sequencing data (WGS) was assessed for both MultiScale Reference and self-reference calling methods using simulated data. Copy number gains and losses were simulated at twelve size ranges between 5 kb to 3.5 Mb at target allele fractions of 10%, 20%, 30%, 40% and 50%. Simulated variants were incorporated into a baseline Illumina WGS sample sequenced to a depth of 45X. Residual CNV calls in the actual sample were identified with SNP-FASST3 and then subtracted from the final results so that performance evaluation was done using only simulated events. Five gains and five losses were simulated for each size bin and variant allele fraction for a total of 600 events with no more than ten events in each simulated sample. Resultant BAM files were analyzed with SNP-FASST3 and CNV calls were assessed for calling accuracy. Centromeric and annotated segmental duplication regions were excluded from the analysis. CNV calls were made using the self-reference method for coverage normalization as well as with an MSR composed of six WGS samples sequenced to a comparable depth with the same methods. See **Figures 5** and **6**.

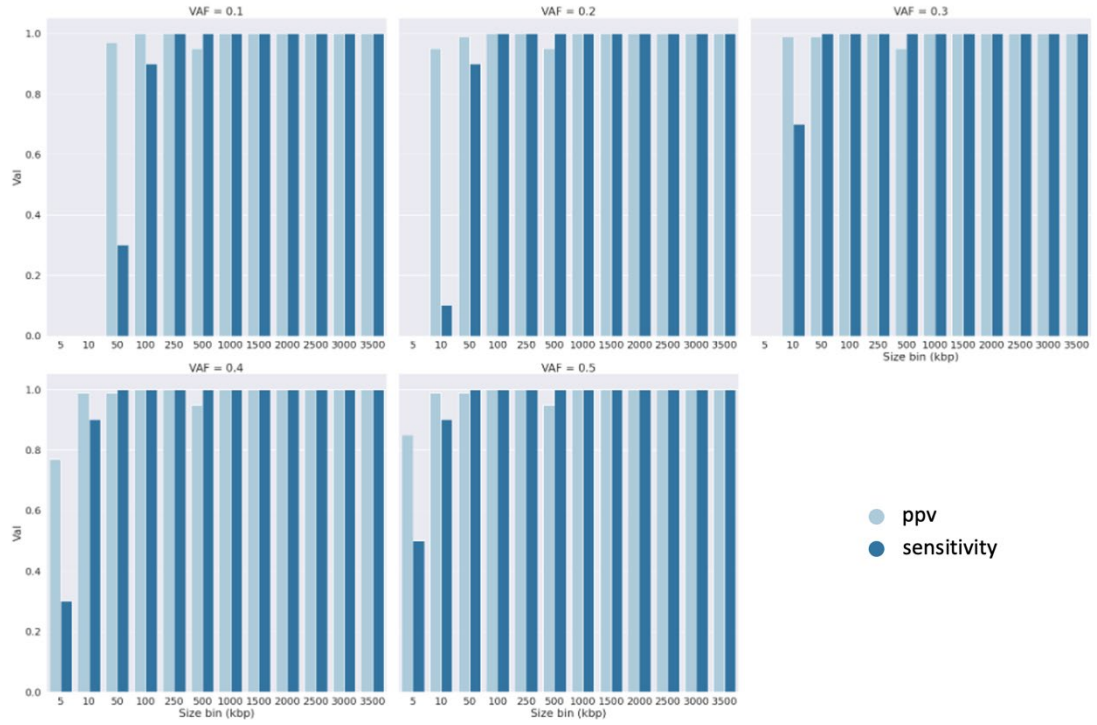


Figure 5. Sensitivity and PPV of simulated CNV events detected using SNP-FASST3 self-reference method

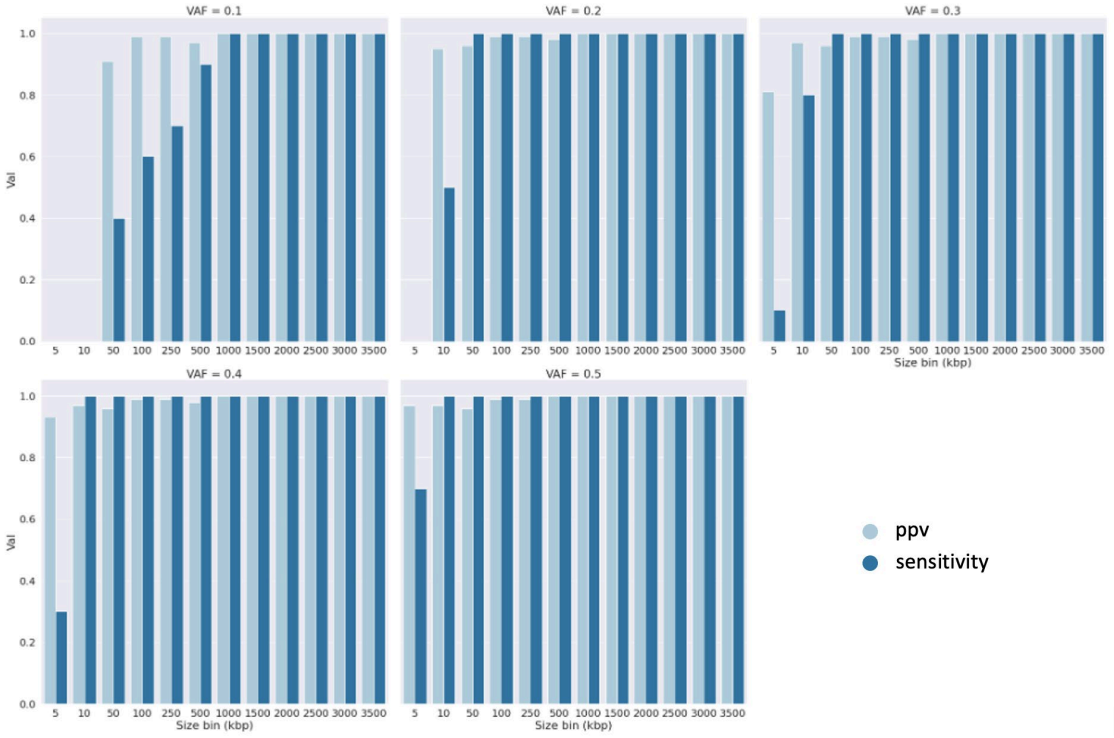


Figure 6. Sensitivity and PPV of simulated CNV events detected using SNP-FASST3 MSR method

OGM Data Integration into VIA

OGM Data Workflow for Segmenting CNVs with SNP-FASST3

OGM samples can be uploaded directly from Access by selecting the option to 'Upload to VIA' when submitting a sample for processing or the data can be manually uploaded into VIA through the Data or the Batch Import method leveraging the ogm.bam (with the accompanying ogm.bam.bai) and ogm.vcf files for each sample.

It is recommended to process OGM BAM files with an OGM BAM MultiScale Reference (MSR) matching the effective coverage, sex, and genome build. The OGM MSR files are created with a set of cytogenetically normal samples through the BAM MultiScale Reference Builder. Like the BAM MultiScale method for NGS, a set of normal OGM BAM data files are used to construct dynamic bins of the coverage profile for the reference dataset to generate a MSR file inclusive of bin positions with expected coverage levels. The MSR file is applied as an *in silico* reference for an experimental data file for CNV segmentation processing. Bionano has generated shareable OGM MSR reference files from male and females control samples at 80x, 160x, and 300x effective coverage levels for each genome build leveraging the following settings.

OGM MSR Reference Builder Settings

OGM Effective Coverage	80x	160x	300x
Minimum Bin Width	1000	1000	1000
Average Read Length	100000	100000	100000
Target Reads per Bin	80	200	200
Maximum Neighbor Bin Gap	100	100	100

B-Allele Frequency (BAF) from OGM with VIA & SNP-FASST3

The B-allele frequency is calculated as the ratio of molecules with missing labels to the total number of molecules aligned to the position. Label sites are filtered to those overlapping known SNPs with a minor allele frequency greater than 5% in the population. Data from 180 control samples are used to identify labels for which the BAF values cluster into three well-formed clusters corresponding to the AA, AB, and BB alleles, indicating that heterozygous and homozygous SNPs can be differentiated. Next, BAF values are normalized for the query sample at each SNP position. Normally, OGM data will have BAF values clustering around 0, 0.45, and 0.9, for the homozygous absent, heterozygous, and homozygous present states, respectively, due to the labeling efficiency being at ~90%. After normalization, the clusters are centered around 0, 0.5, and 1, and the data can be handled in the same way as SNP microarray data for the display of B-Allele Frequency Chart. Finally, AOH regions are called using VIA software's SNP-FASST3 algorithm for segmenting and calling allelic events. See **Figure 7**.

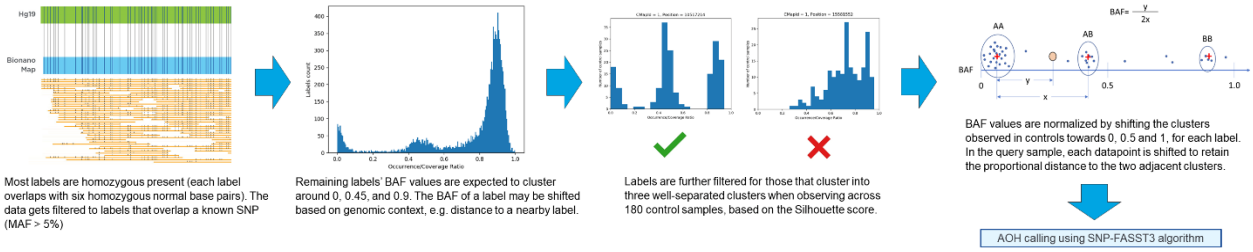


Figure 7. Summary of BAF-based method for filtering and normalizing labels for AOH calling.

SNP-FASST3 CNV & AOH Performance for OGM

CNV and AOH calling performance was evaluated in samples with known events, as well as simulated samples. For AOH evaluation, 230 AOH/LOH events of sizes ranging from 1Mb to 100Mb were simulated at various aberrant cell fractions as low as 5% aberrant cell fraction (ACF), for a total of 1,350 AOH/LOH events evaluated. Sensitivity and precision were calculated for each size range and cell fraction separately. It was observed that recall for 20-25 Mbp AOH events is 92% at 25% ACF. Additional validation was performed using a cohort of constitutional and cancer samples for which orthogonal testing had been performed and for which cell counts and LOH events were available. In 15 samples containing 37 known AOH events, all events were called in 14/15 samples. One 14.6 Mbp AOH event was not called; other AOH events that were greater in size were called. Most false positive calls were under 10 Mbps, and false positive calls in the 40-10 Mbps range could be distinguished using manual review. See **Figure 8**.

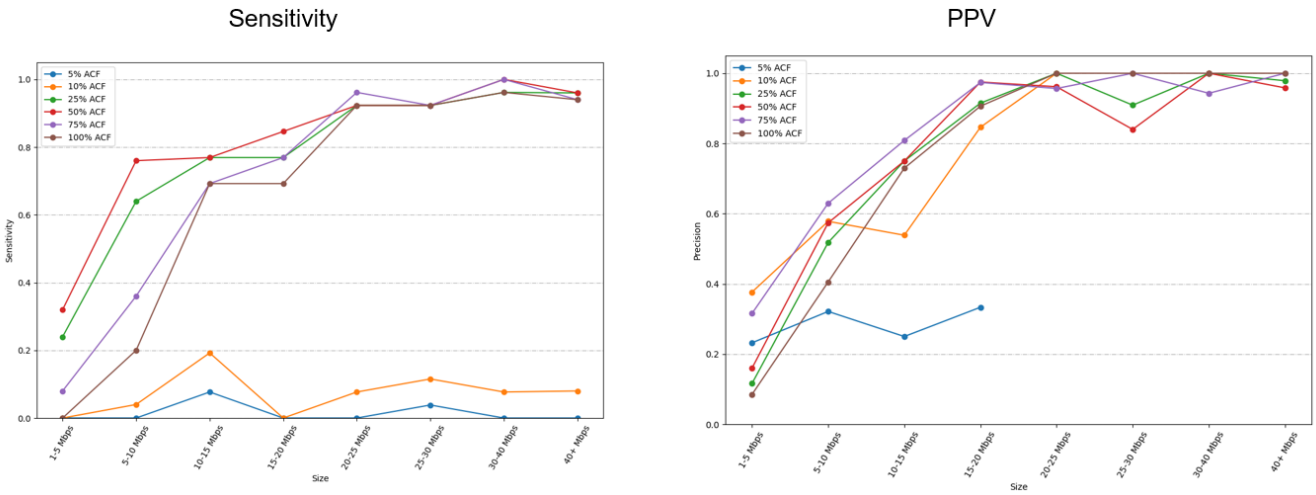


Figure 8. AOH calling performance using simulated data. True positives are defined as 50% overlap required between a simulated event and an AOH call.

For CNV evaluation, 280 CNV events ranging in size from 175kb to 8Mb were simulated on the 22 autosomal chromosomes at multiple coverage depths and allele fractions. Data, seen in **Figures 9, 10** and **11**, were simulated to represent 50% VAF at the 400 Gb (80x) level, 20%, 30% and 50% at the 800 Gbp (160x) level and 5%, 10%, 20%, 30% and 50% at the 1.5 Tb (300x) level. Sensitivity and PPV were calculated for each size range and cell fraction separately. True positives were defined as events with 80% overlap required between simulated event and CNV call. Variants overlapping the OGM CNV Mask region by 45% or more were excluded from

analysis (for a detailed description of the creation and content of the CNV mask, see *Bionano Solve Theory of Operation Structural Variant Calling* (P/N 30110).

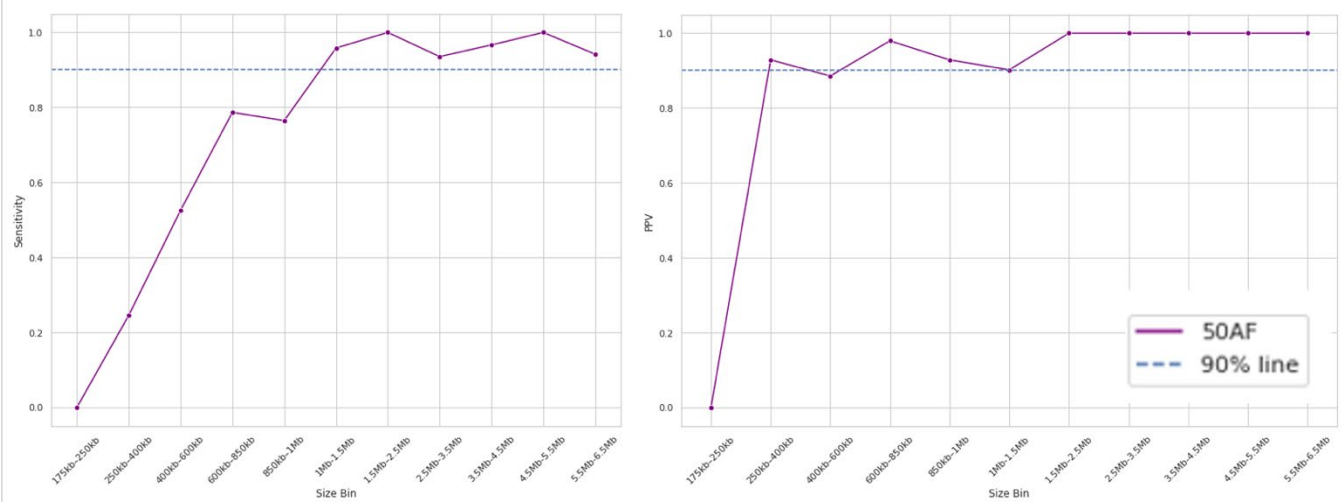


Figure 9. CNV detection performance at 400 Gbp/80x coverage

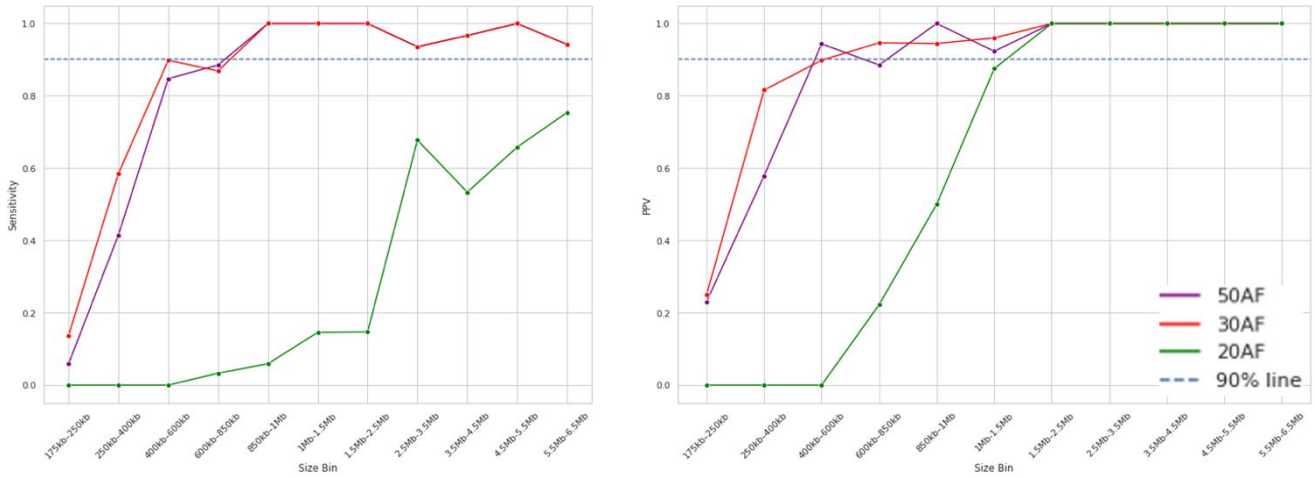


Figure 10. CNV detection performance at 800 Gbp/160x coverage

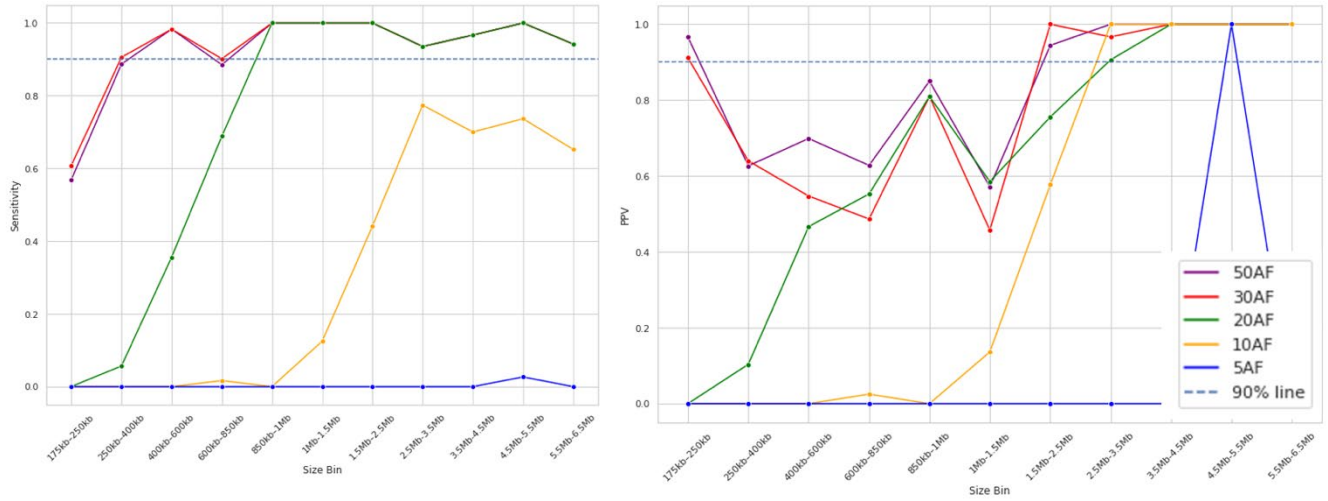


Figure 81. CNV detection performance at 1.5 TB/300x coverage

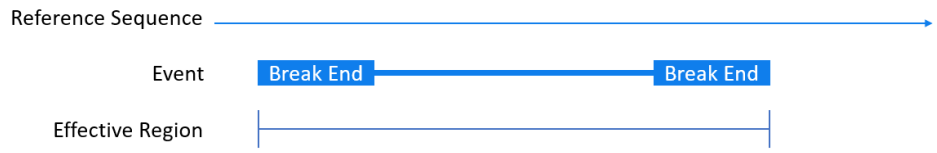
Concepts of Structural Variants (SV) in VIA

Region Types

SV Events in VIA are typically defined by multiple breakend positions plus confidence region with relation to the reference sequence. The region types are combinations of these positions/regions as illustrated below in **Figure 12**.

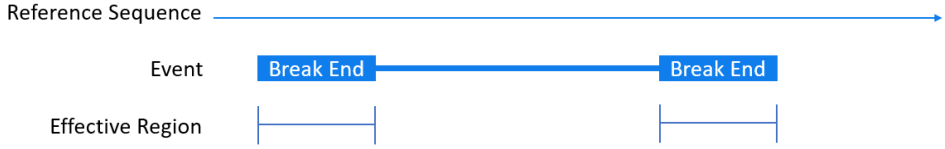
FULL EXTENT REGION

This region is defined by the most upstream point of the confidence region of the start position, and the most downstream point of the confidence region of the end position. This is typically used for Dosage Effect events.



BREAK END REGIONS

This type of region is defined by only the region of the break ends and their confidence interval. This is typically used for gene disruption events, as well as translocation and fusions.



REGION PADDING

Padding may be added to the regions defined above as needed.

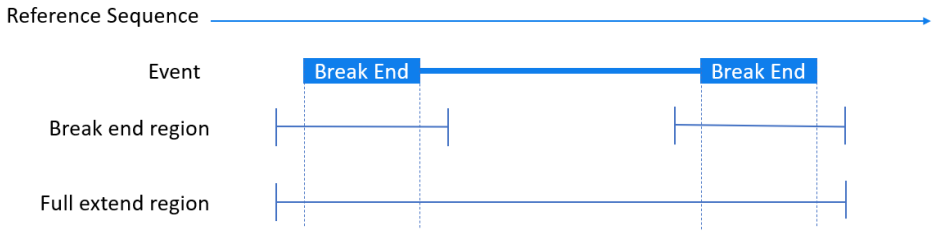


Figure 92. Regions

SV EVENT TYPES

Definitions

Visualization Region: Region used to display the SV event in the SV Track in VIA.
Column Region: Region displayed in the sample event table, variant details tab. This region is also used for region overlap calculations for various annotation regions/lists in the sample event table.
Variant Details Gene Table: Region used to generate the gene table in the variant details tab.

DUPLICATION

This is considered a Dosage Effect event.

- Visualization: Full Extent Region.
- Column Region: Full Extent Region.
- Gene Panel Evaluation: Full Extent Region.
- Variant Details Gene Table: Full Extent Region.

DELETION

This event is considered a Dosage Effect event.

- Visualization: **Full Extent** Region.
- Column Region: **Full Extent** Region.
- Gene Panel Evaluation: **Full Extent** Region.
- Variant Details Gene Table: **Full Extent** Region.

INSERTION

This event is considered a Dosage Effect event.

- Visualization: **Full Extent** Region.
- Column Region: **Full Extent** Region.
- Gene Panel Evaluation: **Full Extent** Region.
- Variant Details Gene Table: **Full Extent** Region.

INVERSION

This event is considered a Gene Disruption event.

- Visualization: **Full Extent** Region.
- Column Region: **Break end** Regions.
- Gene Panel Evaluation: **Break end** Regions.
- Variant Details Gene Table: **Break end Regions** plus padding.

INTRACHR FUSION

This event is considered a Gene Disruption event.

- Visualization: **Break end** Regions.
- Column Region: **Break end** Regions.
- Gene Panel Evaluation: **Break end** Regions.
- Variant Details Gene Table: **Break end** Regions plus padding.

INTERCHR TRANSLOCATION

This event is considered a Gene Disruption event.

- Visualization: **Break end** Regions.
- Column Region: **Break end** Regions.
- Gene Panel Evaluation: **Break end** Regions.
- Variant Details Gene Table: **Break end** Regions plus padding.

INVERTED DUPLICATION

This event is considered both Dosage Effect and Gene Disruption event.

- Visualization**: Full Extent** Region.
- Column Region: **Full Extent** Region.
- Gene Panel Evaluation: **Full Extent** Region.
- Variant Details Gene Table: **Full Extent** Region plus padding.

SV LENGTH

For all SV types other than Inversion, the value displayed in the length column is VIA is the absolute value of the SV length reported in the OGM.VCF file. In the case of Inversion, it is the size of the inversion (SVLEN in the OGM VCF would be 0 because there was no change in actual size), which is calculated using the POS and End specified in the OGM VCF file.

ISCN FORMATTING FOR SV IN VIA

Tables 2 and **3** summarize the positional placement of coordinates used with the nomenclature according to the ISCN for OGM data types.

Table 2. ISCN intended coordinates

Event Name	BNGTYPE	Description of Start and End
Deletion	deletion	First and last base of deleted sequence
Tandem Duplication	duplication	First and last base of duplicated sequence
Inverted Duplication	duplication_inverted	First and last base of duplicated sequence
Insertion	insertion	Base before and base after inserted sequence
Inversion	inversion_paired	First and last base of inverted sequence
Inversion Breakpoint	inversion_partial	First and last base of inverted sequence
Interchr Translocation	translocation_interchr	Position of each break-end
Intrachr Fusion	intrachr_fusion	Position of each break-end

Table 3. ISCN coordinates relative to the VCF

Event Name	BNGTYPE	ALT	Start	End
Deletion	deletion	<DEL>	POS + 1	END
Tandem Duplication	duplication	<DUP:TANDEM>	POS + 1	END
Inverted Duplication	duplication_inverted	<INV>	POS + 1 (see note)	END (see note)
Insertion	insertion	<INS>	POS	END + 1
Inversion	inversion_paired	<INV>	POS + 1	END
Inversion Breakpoint	inversion_partial	N].]	L1.POS + 1	L2.POS
Inversion Breakpoint	inversion_partial	[.[N	L1.POS	L2POS - 1
Interchr Translocation	translocation_interchr	N].] or N[.[or].]N or].]N	L1.POS	L2.POS
Intrachr Fusion	intrachr_fusion	N].] or N[.[or].]N or].]N	L1.POS	L2.POS

TABLE KEY

- L1.POS = Position of first VCF line of event
- L2.POS = Position of second VCF line of event

NOTE

- **NOTE**: The current convention is to list the end before the start for inverted duplications.
- The VCF orders SVs by position
- The ALT field for Inversion Breakpoint, Interchr Translocation and IntraChr Fusion indicates the orientation of the breakends as specified in the VCF v4.2 format specifications. More information on VCF formatting is available in the OGM File Format Specification Sheet.

ISCN EXAMPLES

Table 4 provides example nomenclature for different types of SVs.

Table 4. Example nomenclature for different types of SVs

Event Name	BNGTYPE	ISCN
Deletion	deletion	ogm[GRCh38] 1p36.33(710374_711817)x1
Tandem Duplication	duplication	ogm[GRCh38] dup(1)(p36.13p36.13)(16592650_16616818)
Inverted Duplication	duplication_inverted	ogm[GRCh38]:dup(Y)(q11.23q11.23)(24894362_24882439)
Insertion	insertion	ogm[GRCh38] ins(4;?)(q28.3;?)(130100000_138500000;?)
Inversion	inversion_paired	ogm[GRCh38]:inv(1)(p36.21p36.21)(13040509_13326694)
Inversion Breakpoint	inversion_partial	ogm[GRCh38]:inv(1)(p36.21p36.21)(13040509_13326694)
Interchr Translocation	translocation_interchr	ogm[GRCh38] t(2;11)(p25.1;p15.2)(12000000;13800000)
Intrachr Fusion	intrachr_fusion	ogm[GRCh38] fus(4;4)(q28.3;p14)(138500000;35800000)
Loss	loss	ogm[GRCh38] 1p36.33(710374_711817)x0~1
Gain	gain	ogm[GRCh38] 1p36.33(710374_711817)x2~3

UPD Detection

VIA software allows for the automated detection and analysis of uniparental disomy (UPD) events from either SNP arrays or NGS data. The identification of both isoUPD and heteroUPD are important for genetic

interrogations. From SNP arrays and NGS data with corresponding SNP data, VIA software enables laboratories to detect the presence of both isoUPD and heteroUPD when a parental sample is available.
VIA software compares the SNPs of the proband against its parents using a Hidden Markov model (HMM) to identify regions where uniparental inheritance is more likely than usual biparental inheritance. For example, SNP positions where the proband is homozygous opposite from one of the parents makes uniparental inheritance, from the other parent, more likely. Regions of isoUPD and heteroUPD are flagged on the proband sample to enable the reviewer to make an interpretation of genes impacted by the aberrant parental inheritance pattern.

HMM-based approach to detect UPD events

Emission Probabilities:
We must first model the emission probabilities of the set of genotypes *G* we have observed given the various states *S* (hUPD-mom, hUPD-dad, isoUPD-mom, isoUPD-dad, normal). The likelihood of observing a set of genotypes given a particular state *S* can be defined as the following product over all *n* positions:

$$P(G|S) = \prod_{i=0}^{n} P(g_i|S)$$

Note that in the log space, we can take the above sum instead. The brunt of the approach is in modeling the individual likelihoods $P(g_i \mid S)$. Note that we use the general term *g* includes all three genotypes at the position - mom (m), dad (d), and proband (p). Remembering that $P(g_p, g_m, g_d) = P(g_p \mid g_m, g_d)\ P(g_m, g_d)$ using the chain rule, we can first define $P(g_p \mid g_m, g_d)$:

Take, for example, P(AA | AA, BB). This quantity will be different depending on what inheritance model we assume. For example, under an assumption of normal Mendelian inheritance, this genotype is impossible. It is also impossible under either of the two paternal UPD models (hUPD-dad and isoUPD-dad). However, under both maternal UPD models (hUPD-mom and isoUPD-mom_, P(AA | AA, BB) = 1.0. This quantity can be similarly estimated for all possible combinations of genotypes given our theoretical understanding of the five possible inheritance models.

Next we must define $P(g_m, g_d)$. In this case, we empirically estimate from data the probability of all possible mother-father pairs of genotypes in a platform-specific manner.

Transition Probabilities:
We compute transition probabilities based on a theoretical empirical dataset generated from the following four chromosomal archetypes:

Unaffected - no events
Affected - single whole-chromosome UPD event
Affected - segmental UPD alternating between same-parent types, with 10 segments per chromosome
Affected - lone partial UPD event

The total proportion of affected chromosomes (a) and unaffected chromosomes (u) is computed based on the user-defined significance threshold according to the following formula:

$$a = 2^{\log_{10} t/S}$$
$$u = 1 - a$$

where *t* is the user-defined significance threshold, and S is a constant currently hardcoded to 1e-5. It represents the threshold corresponding to 100% affected chromosomes. However, we cap the affected chromosome result at 0.5, to ensure some minimum representation of unaffected chromosomes even with a very lax significance threshold. From this theoretical dataset of chromosomes, we can compute the probability of transitioning at a given position from one of the five inheritance states to any of the others.

Parent of Origin Calculations

The concept of an "informative" probe, which is a SNP probe that is only consistent if inherited from one parent and not the other (For example, when the proband is AA and the mother is AA and father is BB) can be used to effectively determine parent of origin for trio analyses. This means that without any mendelian error, the probe must have been inherited from the mother. The parent of origin call was made based on which parent a significantly higher number of informative probes favoring inheritance from them. While this works well for trios, for duos, if the Parent of Origin is the present parent, no probes will be informative without knowing the genotype of the missing parent. Moreover, if there is even a single erroneous probe indicating the missing parent, the event may be called (in error) for the missing parent.

Therefore, VIA software implemented a statistically more complete solution to calculate the likelihood ratio for the parent of origin of an event using all the probes in the event. To do this, we take each probe in the event and compare the genotype of the proband against the genotypes of each parent and calculate probability that the proband genotype was inherited from that parent. This happens for all the probes in the event and a summary statistic is calculated that indicated how many times more likely it is that the event was inherited from one parent vs. the other. The likelihood ratio threshold for calling an event from one parent or another is 10x. This means that for the software to assign a parent of origin, the probability of the event having been inherited from that parent must be at least 10 times more than inheritance from the other parent. Hence, the likelihood ratio using all the probes in the event provides a better determination of the parent of origin, especially for events that may not have many “informative” probes, as in the case for duos.

Computing the likelihood of an event origin:

The posterior probability of the event state S (whether it was inherited from the mother or father) given the set of trio genotypes G we are observing can be expressed using Bayes law:

$$P(S \mid G) = \frac{P(G \mid S) * P(S)}{P(G)}$$

To avoid evaluating the general probability of observing all trio genotypes in the event, P(G), the Bayes factor is used to determine the marginal likelihood of our two competing hypotheses (mom origin vs dad origin). In this case, we must only compute the likelihood ratio K:

$$K = \frac{P(G \mid S_m)}{P(G \mid S_d)}$$

If K is large (over 100, for example), then we can confidently assume material origin. If K is small (under 0.01), we can confidently assume paternal origin.

To compute K, only the probability of a set of observed genotypes *G* given the two possible origin states *S* (mom, dad) is used. The likelihood of observing a set of genotypes given a particular state *S* can be defined as the following product over all *n* positions:

$$P(G \mid S) = \prod_{i=0}^{n} P(g_i \mid S)$$

Note that in the log space, we can take the above sum instead. The brunt of the approach is in modeling the individual likelihoods $P(g_i \mid S)$. Note that we use the general term *g* includes all three genotypes at the position - mom (m), dad (d), and proband (p). Remembering that $P(g_p, g_m, g_d) = P(g_p \mid g_m, g_d)\, P(g_m, g_d)$ using the chain rule, we can first define $P(g_p \mid g_m, g_d)$. This quantity is straightforward to compute given an assumption of normal Mendelian inheritance. For example, we know that P(AB | AB, AA) = 0.5. Similar theoreticaly values can be obtained for all other possible combinations of genotypes. $P(g_m, g_d)$, on the other hand, must be estimated empirically over all possible mother-father pairs of genotypes, in a platform-specific manner.

Error Tolerance:

After obtaining the joint probability $P(g_p, g_m, g_d)$ for each possible state, error tolerance is added to each probability due to measurement error. Assuming an error weight of E = 0.03, with the uniform probability over all 27 possible genotypes defined as $p_{uniform} = 1 / 27$. The error-adding function is as follows:

$$p_{new} = (1 - E) * p_{old} + E * p_{uniform}$$

Missing Parent Case:

For duo samples with a missing parent, rather than taking the simple product to obtain the joint probability: $P(g_p, g_m, g_d) = P(g_p \mid g_m, g_d)\, P(g_m, g_d)$, the sum over all possible genotypes is taken for the missing parent, where *a* is the available parent, like so:

$$p(g_p, g_m, g_d) = \sum_{g=AA,AB,BB} p(g_p \mid g_a, g)\, p(g_a, g)$$

HRD Genomic Scar Analysis Overview

Genomic Instability Scoring for HRD

Homologous recombination deficiency (HRD) is the inability to repair double-stranded DNA breaks using the HRR (Homologous Recombination Repair) cellular pathway, which consequentially results in an acquired chromosomal breakage. Clinical research has shown that cells with HRD are more sensitive to certain therapies and a measurement of HRD can be an effective pharmacogenetic biomarker across various tumor types. To provide a functional evaluation of HR status, HRD genomic scarring is an analysis approach to assess three specific quantifiable signatures of HRD genomic instability. VIA includes a measurement of these three genomic scars to aid with HRD status assessment in cancer samples across technology types.

HRD Genomic Scar Processing and Definitions

Within the Admin section for Sample Types that are set to an Oncology Test Type, selecting the automated Perform Genomic Scar Calculation checkbox will activate the analysis during the processing for all associated samples. The VIA Theory of Operations document can be referenced for a detailed description of the genomic scar measurement process in VIA. In brief, genomic CNV and AOH profiles generated across data types are analyzed for scar characteristics through the implementation of three processing steps:

- **Merging**
  - During this step, copy number and allelic events are converted into a single genome representation. The two tracks are merged, preserving all existing breakpoints, and the result is stored with no information loss. Each unique combination of CN and allelic calls gets its own merged event state in the resulting single genome representation.
- **Smoothing**
  - The resulting merged track to combine similar event types and across small gaps as well as the centromere. Each of the three scores has a custom smoothing procedure, which considers each individual score's preferences for minimum event size to consider, maximum gap size across which to merge events, and event types to merge. The gap consideration passes if either the events are close enough together, as specified by the maximum gap size parameter for each of the three individual scores or the two events span the centromere, with no probes in between them. Adjacent events are smoothed only if they are of identical types, or if they map to the same canonical event according to the characteristics of each scar.
- **Selection**
  - The resulting breakpoints (LST) and calls (TAI & LOH) that comply with each scar's specifications are selected according to each scar's criteria. Scar selection is dependent on whether an event touches the telomere, centromere, or the following event. An event is considered touching the telomere if it is overlapping the region listed in the telomere.txt for its respective chromosome or if it contains the last probe on the arm. An event is considered touching the centromere if it is overlapping the region listed in the centromere.txt file for its respective chromosome or if it contains the last probe on the arm. An event touches the next event if there are no probes in the gap between the two events.

The applied definition of each scar is:

- **Loss of heterozygosity (LOH)** - number of regions representing one parental allele resulting from a copy number neutral, or a loss, event that is longer than a specified minimum LOH event size, but shorter than the whole chromosome.
- **Telomeric Allelic Imbalance (TAI)** - number of regions with CNV or allelic imbalance longer than the specified minimum TAI event that extends to one of the telomeres but does not cross the centromere.
- **Large-Scale State Transitions (LST)** - number of chromosomal break points between adjacent regions of change in copy number or allelic content longer than a specified minimum LST event size. Adjacent events with a gap less than the maximum LST gap size are merged. State changes at centromeres and telomeres are excluded.

The characteristic event size and gap size for each genomic scar is configurable. A config file HRD Parameters is retained as a TXT file within the VIA server (../VIA Server/Storage/Resources) that can be modified to adjust the default parameters and refine the scarring performance accordingly. The specific parameters used in calculation of the genomic scars are the minimum event size and the maximum gap size for all three scar types.

HRD Genomic Scar Performance

HRD Genomic Scar Processing was performed on 529 ovarian cancer samples from the Nexus Copy Number TCGA Premier dataset processed in VIA, which had been previously curated to correct for over-segmentation and incorrect ploidy. The combined tally of the genomic scars was compared for 497 samples that had previously reported analyses in Takaya et al 2020 (**Table 5**). Genomic scar calculations were compared for 96 samples that had either a mutation and/or methylation in either *BRCA1* and/or *BRCA2* and 191 samples that had neither a mutation or methylation in *BRCA1* and *BRCA2*. (**Table 6**, **Table 7**). As described by Takaya et al 2020, samples were defined as being HR deficient if the tally of genomic scars was greater than 63. Status for Nexus Copy Number TCGA Premier samples processed in VIA was defined as being deficient if the tally of genomic scars was greater than 42. Default HRD parameters were used to calculate genomic scar tallies, except TAI gap size, which was set to 0 MB.

1. Takaya, H., Nakai, H., Takamatsu, S. et al. Homologous recombination deficiency status-based classification of high-grade serous ovarian carcinoma. Sci Rep 10, 2757 (2020).

Table 5. Takaya et al. vs VIA

All Samples (Takaya et al. vs VIA)	
Concordant	414 (83.30%)
Discordant	83 (16.70%)

Table 6. A Mutation/Methylation in at least either BRCA1/2

A Mutation/Methylation in at least either *BRCA1/2*		
HRD Status	Takaya et al. 2020	VIA
Deficient	74 (77.08%)	84 (87.50%)
Proficient	22 (22.92%)	12 (12.50%)

Table 7. No Mutation/Methylation in BRCA1/2

No Mutation/Methylation in *BRCA1/2*		
HRD Status	Takaya et al. 2020	VIA
Deficient	54 (28.27%)	72 (37.70%)
Proficient	137 (71.73%)	119 (62.30%)

The result of the analysis demonstrated high concordance for the automated assessment of genomic scars associated with HRD in VIA as a robust means to determine HR status from technology types producing copy number and B-allele frequency data for application in clinical research oncology, shown in **Figure 13**.

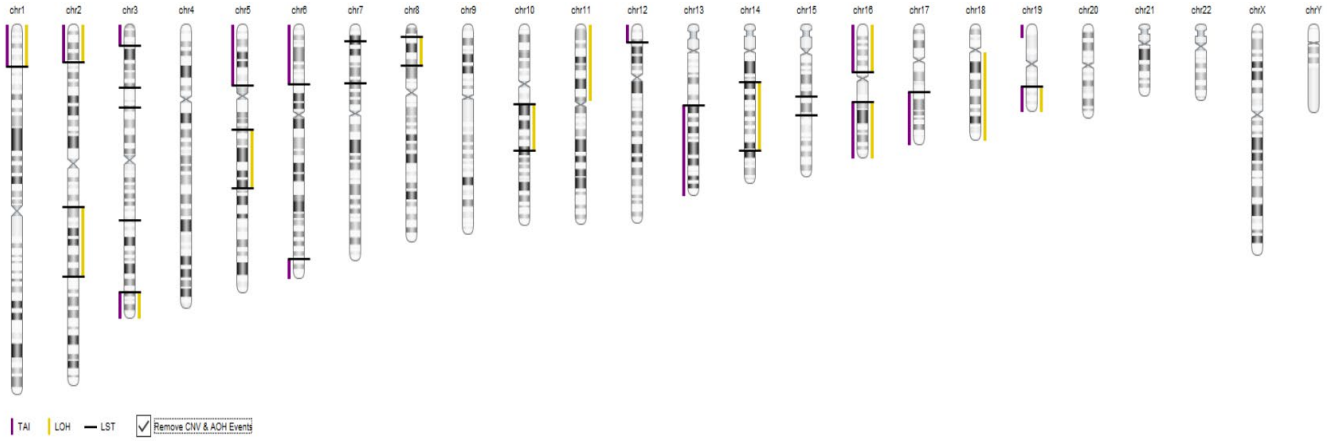


Figure 103. Example visual of HRD genomic scar analysis in VIA

Automatic Pre-Classification Decision Trees

To specify rules for automatic pre-classification, one must create the rules using a specified syntax. Failure to follow the syntax carefully will result in errors and the automated classification may not work. Care also must be taken to make sure all parentheses and curly braces match. The functions used for the decision tree rules are case-sensitive so attention must be given to this as well. Specific keywords and syntax used to create the decision tree rules are detailed in this section.

NOTE: Case, spacing, and use of quotes are important so please make note of that in the examples. Any quotes must be ASCII quotes. Type in quotation marks directly into the query field rather than copying and pasting from other documents as often these use smart quotes rather than ASCII quotes. If quotes are not ASCII quotes, the query will not work.

Rules typically follow the IF, THEN, ELSE statement syntax with use of logical operators **AND** and **OR**. E.g.,

```
IF {A > 10} THEN {X = "yes"}
IF {(A < 10) AND (B > 10)} THEN {X = "no"}
```

There are also other functions used to test various attributes of the event in question.

An example of a rule:

To classify as **Likely Benign**, any copy number loss event that does not overlap any gene in **OMIM Morbid** track, the following rule can be specified:

```
IF {CN_EVENT_KIND IS CN_LOSS OR CN_EVENT_KIND IS HOMOZYGOUS_COPY_LOSS}
THEN {
    IF {!(OVERLAP(OMIM Morbid) > 0 )}
    THEN {CLASSIFY("Likely Benign")}
     }
```

Or it can be written more concisely using CASE statements:

```
CASE {CN_EVENT_KIND IS CN_LOSS OR CN_EVENT_KIND IS HOMOZYGOUS_COPY_LOSS}
{ CASE{!(OVERLAP(OMIM Morbid) > 0 )}  {CLASSIFY("Likely Benign")} }
```

NOTE: there are examples of decision tree scripts in the last section of this document.

Keywords and Syntax For the Decision Tree Rules

TEXT STYLE

All functions must be in uppercase, or the function will not work.

```
IF {A > 10} THEN {X = "yes"}  works
if {A > 10} Then {X = "yes"}  will not work
```

COMMENTING IN THE SCRIPT

Two forward slashes (//)can be used to add comments to the scripts and text after this is not interpreted as part of the code. E.g.,

```
CASE {CN_EVENT_KIND IS CN_LOSS OR CN_EVENT_KIND IS HOMOZYGOUS_COPY_LOSS}
{ CASE{!(OVERLAP(OMIM Morbid Phenotypes) > 0 )}  {CLASSIFY("Likely Benign")} }  // calls copy losses that do not
   overlap with the OMIM Morbid Phenotypes track as "Likely Benign"
```

AUTOCOMPLETE

So that users do not have to remember all various functions and possible values, the decision tree editor has a predictive text/autocomplete feature where as one starts typing, possible values are displayed. Use the mouse to select a value by double clicking on it, using the **TAB** key to select the first item or use the **arrow** keys to highlight the term and hit **Enter**. **NOTE**: Autocomplete only works if the user is typing using ALL CAPS.

Typing an uppercase **C** brings up the list seen in **Figure 14**. If a lowercase **c** is entered, no options are displayed for completing the statement. **Figure 115** is an example of looking for CN events. A **Ctrl-Space** command can also be keyed in to see a list of all possible keywords.

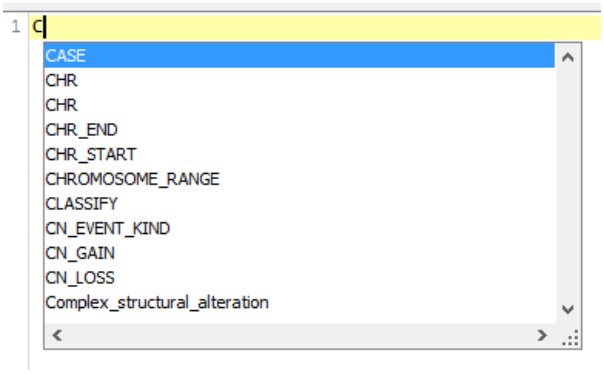


Figure 14. Autocomplete

Figure 125. Looking for CN events

OPERATOR BASICS

When the value being evaluated is numerical, the following operators are used:

> Greater than

< Less than

>= Greater than or equal to

!= Not equal to

<= Less than or equal to

== Equal to (**NOTE**: must use two equal signs in succession)

When the value being evaluated is a text string, the following operators are used:

== Equals

!= Does not equal

Basic Functions

VAL (USING VARIABLES)

It is possible to define and set a variable at the beginning of the script to be used throughout the script. This makes it easy to change various thresholds that may be defined by changing the value of the variable rather than searching for and changing the constant in many different locations.

The keyword **val** is used to define a variable. E.g.,

val percent_overlap = 0.50

val similarity_threshold = 0.85

val `listSampleTypes = List("CytoScan HD," "Affymetrix OncoScan")`

Then within the script the variables can be used instead of the actual values:

```
CASE {OVERLAP("OMIM Morbid Phenotypes") > percent_overlap} {CLASSIFY("Likely Pathogenic")}

CASE {SIMILARITY("DECIPHER Syndromes") > similarity_threshold}  {CLASSIFY("Pathogenic")}

CASE {SIMILARITY ("DECIPHER Syndromes Gains")> similarity_threshold } {CLASSIFY("Pathogenic")}      // Classify
   pathogenic based on similarity to cases in DECIPHER Syndromes

CASE {SIMILARITY ("ClinGen Dosage Sensitive Map Triplosensitivity Pathologic Regions")> similarity_threshold }
   {CLASSIFY("Pathogenic")}

CASE {(SIMILAR_CASES("ClinGen Postnatal Gains Uncertain Significance", similarity_threshold) >= 3) OR (SIMILAR_CASES
   ("ClinGen Postnatal Gains Pathogenic", similarity_threshold) >=3 ) OR (SIMILAR_CASES ("ClinGen Postnatal Gains
   Likely Pathogenic", similarity_threshold) >=3 )} {CLASSIFY("VUS")}
```

If the numbers 0.5 and 0.85 are used in many places in the script and you want to adjust that, it is easy to do so as one only needs to change it in one spot. If you want to use 0.89 instead of 0.85 as the similarity threshold, you can just change the value of the variable once at the beginning of the script.

IF {} THEN {} ELSE {}

If the condition being evaluated is true do one thing and if it is false do something else.

```
IF { condition is true} THEN {do this} ELSE {do that}

IF {OVERLAP("OMIM Morbid Phenotypes") > 0} THEN {CLASSIFY("Possible reportable event")} ELSE {CLASSIFY("VUS")
```

If the event overlaps with a region in the OMIM Morbid Phenotypes track, then classify the event as “Possible reportable event” or else classify it as “VUS.”

The following uses of IF/THEN statements are NOT supported:

- Combining IF/THEN statements in a decision tree script without ELSE clause:

  ```
  IF {ANY_SNP_EVENT_KIND} THEN { CLASSIFY("VUS") }

  IF {EVENT_SIZE > (1 Mb)} THEN { CLASSIFY("Pathogenic") }
  ```

- Nest IF statements:

```
IF {ANY_SNP_EVENT_KIND} THEN {CLASSIFY("VUS")} ELSE {IF {EVENT_SIZE > (1 Mb)} THEN {CLASSIFY("Pathogenic")}}
```

AND/OR

AND or **OR** can be used to check that either multiple conditions are true or that one of multiple conditions is true. E.g.,

```
IF {(grade > 1) AND (grade < 6)} THEN {school = "elementary"} ELSE {school = "not elementary"}
IF {(grade == 6) OR (grade == 7) OR (grade == 8)} THEN {school = "middle"} ELSE {school = "not middle"}
```

Example usage:

IF { (OVERLAP("OMIM Morbid Phenotypes") < 0) AND (OVERLAP("RefSeq Exons")==0) } THEN {CLASSIFY("Likely Benign")} ELSE {CLASSIFY("Unclassified")}

CASE {} {}

This replaces multiple **IF/THEN** statements. This is the preferred way to write statements as it is more concise. The first set of parentheses has a condition being tested, if the condition is true, the second set contains the action. Additional CASE statements can be added to the line. Conditions are evaluated in sequence until a stopping/terminating action is encountered (e.g., CLASSIFY). Additional CASE statements can be nested within the second set of curly braces (the action item). If a true condition is not met, evaluations move on to the next rule.

```
CASE {if grade equals "kindergarten"} {then in elementary school}
CASE {if grade <= 5} {then in Elementary school}
CASE {if grade <= 8} {then in Middle School}
CASE {if grade <= 12} {then in High School}
```

If in the example above, the grade happens to be 13, then none of the cases will be true and the next rule will be evaluated. If the grade happens to be 7, then the first three cases will be evaluated and since the 3rd case will result in *true*. The next case will not be evaluated and the next rule will be evaluated.

E.g., if the DGV score is greater than .98 then classify as “Benign.”

```
CASE {SCORE(DGV) > .98} {CLASSIFY("Benign") }
```

If the DGV score is greater than .98 then classify as *Benign*. If not, then evaluate the second case to see if the score is greater than .85 and, if so, then classify as “Likely Benign.”

```
CASE {SCORE(DGV) > .98} {CLASSIFY("Benign") }
CASE {SCORE(DGV) > .85} {CLASSIFY("Likely Benign") }
```

NOTE: In the nested CASE statement below, if the event size is less than 100 then the next CASE is evaluated. If the event is a copy number event, then it is classified as “Likely Benign;” if it is not a copy number event or not less than 100, the next rule would be evaluated.

```
CASE {EVENT_SIZE <100} { CASE {CN_EVENT_KIND IS COPY_GAIN} {CLASSIFY("Likely Benign"}}
```

Classifying an Event

CLASSIFY()

Allows classification of an event and terminates the script. The value that goes inside the parentheses must be a text string in quotes and it must match one of the pre-defined classification values set up via the VIA Administrator.

For example, the VIA Administrator has defined the following classification values: “Pathogenic” “Likely Pathogenic,” “Benign,” “Likely Benign,” “VUS.”

Example usage:

```
CASE {OVERLAP("OMIM morbid")} {CLASSIFY("Likely Pathogenic")}
```

NOTE: If the event overlaps with an event in the OMIM morbid track, then the event is probably pathogenic so classify this as “Likely Pathogenic.”

The syntax above is valid (the value passed into “CLASSIFY” is a defined classification term and is enclosed in quotes. The following are examples of invalid syntax usage:

```
CASE {OVERLAP("OMIM morbid")} {CLASSIFY("LIKELY PATHOGENIC")}
```

NOTE: The value passed into CLASSIFY is case-sensitive and must exactly match the pre-defined classification values. Here the value passed in is in all uppercase whereas the defined pre-classification values are not (defined value: Likely Pathogenic).

```
CASE {OVERLAP("OMIM morbid")} {CLASSIFY(Maybe Pathogenic)}
```

NOTE: The value passed in to CLASSIFY must match exactly the pre-defined classification values. “Maybe Pathogenic” is not one of the pre-defined classification values. Also, note the quote marks are required.

```
CASE {OVERLAP("OMIM morbid")} {CLASSIFY(Likely Pathogenic)}
```

NOTE: Value passed into CLASSIFY is a pre-defined classification value, but it is not enclosed in quotes, which is required.

Evaluating the type of Event (e.g., copy number gain, AOH, SNV, etc.)

These functions assess the type for the event. They evaluate whether the event is a copy number event and what type of copy number event (single copy gain, high copy gain, …) or whether the event is an allelic event (AOH, total allelic loss, …).

ANY_CN_EVENT_KIND, ANY_SNP_EVENT_KIND, ANY_SEQVAR_EVENT_KIND

Looks for any copy number (ANY_CN_EVENT_KIND), allelic (ANY_SNP_EVENT_KIND), or sequence variant (ANY_SeqVAR_EVENT_KIND) event.

NOTE: Prior to version 7.0 ANY_SV_EVENT_KIND is used for sequence variants. Version 7.0 onwards use ANY_SEQVAR_EVENT_KIND for sequence variants.

Example usage:

```
CASE {ANY_CN_EVENT_KIND}
{ CASE{SCORE(DGV) > .95} {CLASSIFY("Likely Benign")} }
```

If the event is any copy number event and the DGV score is greater than 0.95, then classify as “Likely Benign.”

CN_EVENT_KIND IS, SNP_EVENT_KIND IS, SEQVAR_EVENT_KIND IS

NOTE: Prior to version 7.0 SV_EVENT_KIND IS was used for sequence variants. Version 7.0 onwards use SEQVAR_EVENT_KIND IS for sequence variants.

Checks for the type of copy number event (e.g., high copy gain), SNP event (e.g., AOH), or sequence variant event (e.g., SNV, Insertion, Deletion).

Example usage:

```
CASE {CN_EVENT_KIND IS HIGH_COPY_GAIN} {…}
CASE {SV_EVENT_KIND IS STRUCTURAL_ALTERATION} {…}
```

Valid values for **CN_EVENT_KIND**:

HIGH_COPY_GAIN
CN_GAIN
CN_LOSS
HOMOZYGOUS_COPY_LOSS

Valid values for **SNP_EVENT_KIND**:

AOH
ALLELIC_IMBALANCE
HETERO_UPD_FATHER
HETERO_UPD_MOTHER
ISO_UPD_FATHER
ISO_UPD_MOTHER

Valid values for **SEQVAR_EVENT_KIND**:

SNV, **Indel**, …

There are numerous possible values for SEQVAR_EVENT_KIND. To see the complete list of possible values, go to the **Admin->Sample Types->Sample Review Preferences->Filter** tab. Any values listed in the “Seq. Var. Events” “Show Events of Type” section are permitted, shown in **Figure 16**.

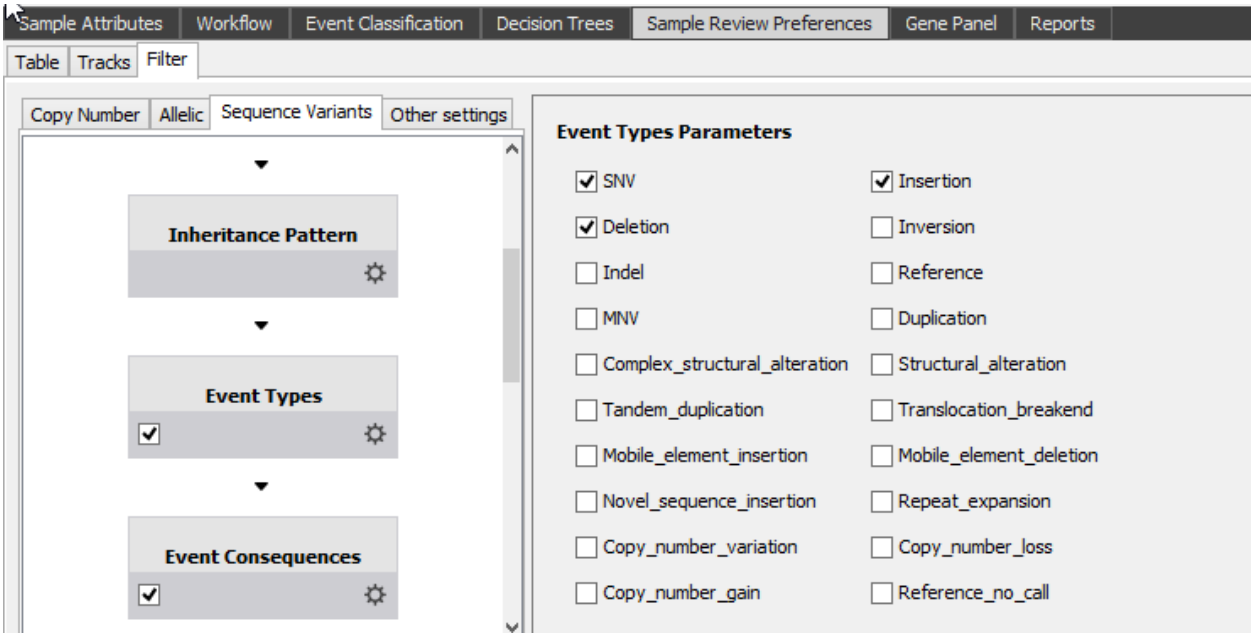


Figure 136. Screenshot of partial list of values in the **Event Types** filter.

The negation can be used with this function to look for any event other than a single event type. For example, if one wants to check to see if a CN event is a high copy gain and nothing else, one can use the negation (!):

```
CASE {(ANY_CN_EVENT_KIND) AND !(CN_EVENT_KIND IS HIGH_COPY_GAIN)}  {…}
```

If the event is a copy number event other than high copy gain, then…

NOTE: In the statement above, users must also first check to see if the event is a copy number/allelic/sequence variant event using the functions beginning with “ANY_.” For example, for copy number, only checking to see that it is not a high copy gain will return any other event (including allelic events) as true.

Evaluating Event Size and Location

EVENT_SIZE

Specifies the length of an event (gain, loss, etc.). Units can be in bp, Kb, Mb, and Gb. E.g., the following all equal 1,000,000,000 bp:

1000000000

1000000.Kb

1000.Mb

1.Gb

NOTE: To use Gb, Mb, Kb, a period must be placed between the number and the unit. Specifying as 1000 Mb is incorrect. It must be specified as 1000.Mb. When specifying in bp, no unit is needed.

Allowed operators:

- > Greater than
- < Less than
- >= Greater than or equal to
- != Not equal to
- <= Less than or equal to
- == Equal to (**NOTE**: must use two equal signs in succession)

Example usage:

```
CASE {EVENT_SIZE >== 5000} {…}
CASE {EVENT_SIZE >== 5.Kb} {…}
```

CHR IS

Determines on which chromosome the event is located. The function returns a “true” or “false.”

Example usage:

```
CASE {CHR IS Chr4} {do this}
```

Syntax for specifying chromosomes: Can use uppercase for first letter or all lowercase with chromosome numbering ranging from 1 to 24 and can also use X and Y.

E.g., the following are all valid:

```
Chr3, chr3, ChrX, chrX, ChrY, chrY, Chr23, chr23
```

POSITION()

Refers to the location of an event and returns a “true” or “false” Arguments (locations on one or more chromosomes, either a base pair location range or a chromosomal location range) are passed in to the POSITION() function separated by commas. An optional minimum overlap value can be passed in as the last item in the argument list and if so, the function will then test to see if at least the specified percentage of the event is overlapping with one of the specified regions. If no minimum overlap value is explicitly specified, it is assumed to be 1.0 (i.e., 100% of the event must lie within one of the specified ranges). The minimum overlap argument must be a number between 0 and 1.0.

If the minimum overlap value is not specified, an unlimited number of chromosome ranges can be passed in. If minimum overlap is specified, up to 12 chromosome ranges can be passed in. **NOTE**: The ranges specified are all inclusive (i.e., if the range specified is 10000->20000, then the function will return true if the event is from 10000 to 11000 and if the event is from 17000 to 20000).

Example usage without specifying a minimum overlap:

```
CASE {POSITION(10000 -> 20000, Chr1 -> Chr3, Chr5::CHR_START -> 500000)} {do this }
```

NOTE: If the entire event (100%) falls in any one of the following ranges, then it is true:

- between and including 10,000 bp and 20,000 bp on any chromosome or
- anywhere on chromosomes 1, 2, or 3, or
- on chromosome 5 between and including the start of the chromosome and 500,000 bp

Example usage specifying a minimum overlap:

```
CASE { POSITION (CHR4::CHR_START -> 10.Mb, Chr10::10000000 -> 15000000, 0.6)} { do this }
```

NOTE: If at least 60% of the event falls in any one of the specified ranges, then the function will evaluate to true.

NOTE: For events that are much larger than the region, even if the specified region is completely covered/overlapping with the event, the function may not return as true since the specified percentage of the event must overlap the region. E.g., if the region is 200 bp and the event is 600 bp and the event completely covers the region, the following function will not return as true since 60% of the event (360 bp) cannot overlap the region as the region is too small (200 bp).

```
CASE { POSITION (Chr10::10000000 -> 10000200, 0.6)} { do this }
```

Location units can be in bp, Kb, Mb, and Gb. E.g., the following all equal 1,000,000,000 bp:

1000000000

1000000.Kb

1000.Mb

1.Gb

NOTE: To use Gb, Mb, Kb, *a dot must be placed* between the number and the unit. Specifying as 1000 Mb is incorrect. It must be specified as 1000.Mb. When specifying in bp, no unit is needed. E.g.,

```
CASE {POSITION(10.Kb -> 2.Mb, Chr1 -> Chr3, Chr5::CHR_START -> 500.Kb, Chr10::CHR_START->200000)} {do this}
```

CHR_START, CHR_END

Refers to the start and end positions of a chromosome. **CHR_START** refers to the bp location of the start of the chromosome and **CHR_END** refers to the bp location of the end of the chromosome. These can only be used in conjunction with the **POSITION** function.

Example usage:

```
CASE { POSITION(10000 -> CHR_END) } {do this }
```

NOTE: If the event falls within 10,000bp and the end of the chromosome on any chromosome, then do this.

```
CASE { POSITION(CHR_START -> 200000) } {do this}
```

NOTE: If the event falls within the start of the chromosome and 200,000bp on any chromosome, then do this.

```
CASE { POSITION(CHR_START -> 200000, 300000 -> CHR_END) } {do this}
```

NOTE: If the event falls within the start of the chromosome and 200,000bp, or it falls within 300000bp and the end of the chromosome, then do this.

```
CASE { POSITION(Chr3::100000 -> CHR_END) } { ... }
```

NOTE: If the event falls on chromosome 3 within 100000bp and the end of the chromosome, then do this.

```
CASE { POSITION(Chr3::CHR_START -> 200000)} { ... }
```

NOTE: If the event is on chromosome 3 and falls within the start of the chromosome and 200000bp, then do this.

```
CASE { POSITION(Chr3::CHR_START -> 200, Chr4::300 -> CHR_END) } { ... }
```

NOTE: If the event is on chromosome 3 and falls within the start of chromosome 3 and 200,000bp or if the event is on chromosome 4 and falls within 300,000bp and the end of chromosome 4, then do this.

Comparing Events to Region lists and Evaluating Similarity scores

EVENT_OVERLAP()

***Replaces the function* OVERLAP().**

Tests to see how much of the event overlaps with regions in a specified region list and how the overlap percentage compares to a specified threshold. The input to EVENT_OVERLAP() is a text string that is the name of an annotation list/track. The EVENT_OVERLAP function will then determine if the event in question overlaps with (is covered by) one or more regions in the list and will return a decimal number indicating the percent of the event overlapping the region. For example, if the event overlaps with a region in the specified region list by 31% (i.e., 31% of the event overlaps with the specified region), the EVENT_OVERLAP function will return a value of ".31". Then the percent overlap can be compared to a given percent using operators to return either TRUE or FALSE. See examples in **Figures 17-19**.

Figure 147. Example 1: EVENT_OVERLAP (OMIM Morbid Phenotypes) will evaluate to 1 (100%)

Figure 158. Example 2: EVENT_OVERLAP (OMIM Morbid Phenotypes) will evaluate to 0.8 (80%)

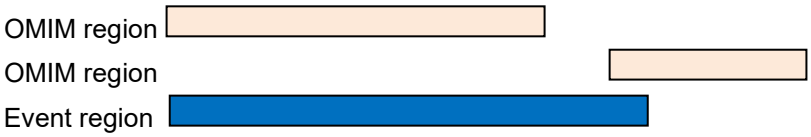


Figure 169. Example 3: EVENT_OVERLAP (OMIM Morbid Phenotypes) will evaluate to 0.9 (90%) since 90% of the event is covered by regions overlapping the event.

When multiple regions are overlapping the event, the coverage will be added or summed. It is possible for this to be greater than 1 if many regions stack up over the sample event.

Allowed operators:

- **>** Greater than
- **<** Less than
- **>=** Greater than or equal to
- != Not equal to
- **<=** Less than or equal to
- **==** Equal to (**NOTE**: Must use two equal signs in succession)

A new optional, padding/flanking region parameter has been introduced in version 7.0. This parameter is used to specify a base pair padding to the event region in both directions when calculating the overlap. The padding region can be specified without units to indicate base pairs or with units, Kb, Mb, and Gb.

Example usage:

```
CASE {EVENT_OVERLAP("OMIM Morbid Phenotypes") > .51} {CLASSIFY("Likely Pathogenic")}

CASE {EVENT_OVERLAP("OMIM Morbid Phenotypes", 2000000) > .51} {CLASSIFY("Likely Pathogenic")}

CASE {EVENT_OVERLAP("OMIM Morbid Phenotypes", 2 Mb) > .51} {CLASSIFY("Likely Pathogenic")}
```

If more than 51% of the event overlaps a region in the **OMIM Morbid Phenotypes** track, then classify the event as “Likely Pathogenic”.

REGION_OVERLAP()

Tests to see how much of a region in a specified region list overlaps with the event in question and how the overlap percentage compares to a specified threshold. The input to REGION_OVERLAP() is a text string that is the name of an annotation list/track. The **REGION_OVERLAP** function will then determine if a region overlaps with (is covered by) the event in question and will return a decimal number indicating the percent of the region overlapping the event. For example, if a region in the specified region list overlaps with the event by 31% (i.e., 31% of the region is covered by the event), the **REGION_OVERLAP** function will return a value of “.31”. Then the percent overlap can be compared to a given percent using operators to return either TRUE or FALSE. See examples in **Figures 20-22**.

Figure20. Example 1: REGION_OVERLAP(OMIM Morbid Phenotypes) will evaluate to 0.8 (80%).

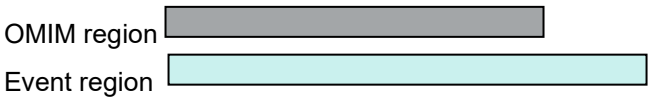


Figure 171. Example 2: REGION_OVERLAP(OMIM Morbid Phenotypes) will evaluate to 1 (100%).

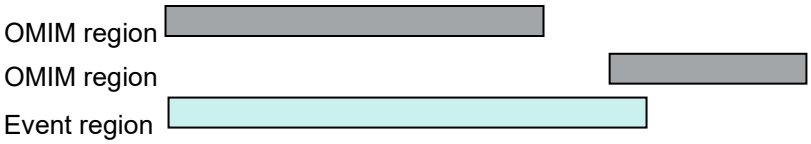


Figure 182. Example 3: REGION_OVERLAP (OMIM Morbid Phenotypes) will evaluate to 1 (100%).

In cases where multiple regions are overlapping the event, each region's overlap with the event is calculated separately and then the largest overlap value is returned. Since one region is completely covered by the event (100%) and the other is only covered 20% by the event (value is .2), the larger value, 1 (100%) is returned. This value cannot be greater than 1.

Allowed operators:

> Greater than

< Less than

>= Greater than or equal to

!= Not equal to

<= Less than or equal to

== Equal to (**NOTE**: Must use two equal signs in succession)

Example usage:

```
CASE {REGION_OVERLAP(OMIM Morbid Phenotypes) > .80} {CLASSIFY("Likely Pathogenic")}
```

NOTE: If more than 80% of a region in the OMIM Morbid Phenotypes track overlaps the event, then classify the event as "Likely Pathogenic".

SIMILARITY()

Tests to see if the event is like regions (of the same class) in a specified region list. The input to SIMILARITY() is a text string that is the name of an annotation list/track. The **SIMILARITY** function will then determine if the event in question is like a region in the list and will return a decimal number indicating the percent similarity. It only looks at those events that are of the same class/type (e.g., if the event in question is a gain, then the function only looks at gain events in the region list). For example, if the event is like a region in the specified region list by 70%, the SIMILARITY function will return a value of "0.7". Then the similarity can be compared to a given percent using operators to return either TRUE or FALSE.

Similarity to another event is based on a ratio of overlap of the common region between the two events and the entire length encompassed by the two events. The percent similarity is defined as a/b where a is the event in question and b is the region covered by both the region being compared as well as the event. **Figure 23** demonstrates this concept.

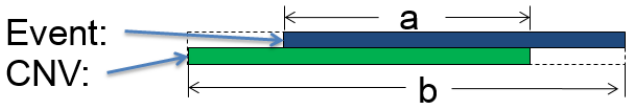


Figure 193. Similarity

Allowed operators:

> Greater than

< Less than

>= Greater than or equal to

!= Not equal to

<= Less than or equal to

== Equal to (**NOTE**: Must use two equal signs in succession)

Example usage:

```
CASE {SIMILARITY("DECIPHER Syndromes") > 0.5}  {CLASSIFY("Pathogenic")}
```

NOTE: If the event is like an event in the DECIPHER Syndromes list by more than 50%, then classify the event as "Likely Pathogenic".

SIMILAR_CASES()

Tests to see if the event in question is like events in a region file in the system and if the events meet or exceed a similarity threshold also passed in as an argument. The input to SIMILAR_CASES() is a text string that is the name of a region file and a number indicating the similarity threshold. The number of similar events that pass that criteria are then compared to a given number using operators to return either TRUE or FALSE.

SIMILAR_CASES (region file, similarity threshold): This looks at events in a region file like the event in question at or exceeding a similarity threshold also passed in as an argument. The number of similar events that pass that criteria are then compared to a given number using operators to return either TRUE or FALSE. E.g., **SIMILAR_CASES**("ClinGen Postnatal Gains Pathogenic", 0.9) looks at events in the "ClinGen Postnatal Gains Pathogenic" region file that are similar to the event in question by at least 90% (see section on *SIMILARITY*(), page 38, for details on calculation of similarity). Next, the function makes sure that the number of cases meeting the similarity requirement meets the user defined minimum number of cases. E.g.,

```
CASE {(SIMILAR_CASES("ClinGen Postnatal Gains Pathogenic", 0.9)) >= 6 } { … }
```

NOTE: If the "ClinGen Postnatal Gains Pathogenic" region file contains events that are like the event in question by at least 90% and if there are at least six such events in the region file, then move to the next step.

SCORE()

A comparison of specific properties/functions to the event in question is made to generate a score. It is used in conjunction with other functions to provide the number of previous cases with similar events, the DGV score, and the evidence score.

Keywords/functions that can be passed to the Score() function: **DGV, PREVIOUS_SIMILAR_CASES, EVIDENCE**

PREVIOUS_SIMILAR_CASES()

This function looks at previous cases to find those that are like the case under review based on criteria passed into the function. It only works for copy number events. This function must be used in conjunction with the SCORE function in the following manner:

SCORE(PREVIOUS_SIMILAR_CASES (classification, similarity threshold, list of Sample Types, earliest processing date)**)**

PREVIOUS_SIMILAR_CASES looks at previous cases to return the number that matches the classification that is passed into the function and that meet or exceed a similarity threshold also passed in as an argument. Additional parameters that can be passed in include a list of Sample Types to limit the search to only those Sample Types and the earliest processing date to limit the search to only those samples processed on or since the specified date. Only the classification parameter is a required and others are optional but **parameters must be specified in this order**: classification, similarity threshold, list of Sample Types, earliest processing date. If a similarity threshold is not specified in the function, a default of 0.9 is used.

The number of similar events that pass that criteria are then compared to a given number using operators to return either TRUE or FALSE via the SCORE function. E.g.,

SCORE(PREVIOUS_SIMILAR_CASES(“Benign,” 0.95) > 2**)** looks at past cases in the database that have been classified as “Benign”. It then identifies those that are similar to the event in question by at least 95% (see section on *SIMILARITY*(), page 38, for details on calculation of similarity). Next, the function makes sure that the number of cases meeting the similarity requirement meets the user defined minimum number of cases.

The list of Sample Types is passed in as a list using the following syntax:

```
List(“Affymetrix CytoScan 750K – Postnatal”, “Affymetrix CytoScan HD – Postnatal”, “Illumina CytoSNP-850K –
    Postnatal”)
```

E.g., `SCORE(PREVIOUS_SIMILAR_CASES("Benign", 0.95, List("Affymetrix CytoScan 750K – Postnatal", "Affymetrix CytoScan HD – Postnatal", "Illumina CytoSNP-850K – Postnatal"))) >= 6 }  { CLASSIFY("Benign")}`

NOTE: A variable can also be specified for the list of Sample Types at the beginning of the decision tree script and then the variable can be used in the script instead. E.g..

```
val limitSampleTypes = List("Affymetrix CytoScan 750K – Postnatal", "Affymetrix CytoScan HD – Postnatal", "Illumina
    CytoSNP-850K – Postnatal")
```

E.g., `SCORE(PREVIOUS_SIMILAR_CASES("Benign", 0.95, limitSampleTypes)) >= 6 }  { CLASSIFY("Benign")}`

NOTE: To also specify which samples to match based on processing date, an earliest processing date can be specified. This means that the function will only look at samples processed on or after the specified date.

E.g., `SCORE(PREVIOUS_SIMILAR_CASES("Benign", 0.95, limitSampleTypes, "2018-01-01")) >= 6 } { CLASSIFY("Benign")}`

If the database contains events that have previously been classified as “Benign” that are similar to the event in question by at least 95%, that are of Sample Types specified by the variable “limitSampleTypes”, where samples were processed on or after January 1, 2018, and if there are at least six such similar past cases, then classify the current event as “Benign”

NOTE: PREVIOUS_SIMILAR_CASES score is only calculated for copy number events so one must first check to make sure that the event is a copy number event before checking for the PREVIOUS_SIMILAR_CASES score or else an error will occur.

```
Correct usage:  CASE {ANY_CN_EVENT_KIND AND SCORE(PREVIOUS_SIMILAR_CASES("Benign", 0.95)) >= 6 }  { CLASSIFY("Benign")}
```

NOTE: If the database contains events that have previously been classified as “Benign” that are like the event in question by at least 95% and if there are at least six such similar past cases, then classify the current event as “Benign”

Incorrect usage: `CASE {SCORE(PREVIOUS_SIMILAR_CASES("Benign", 0.95)) >= 6 } { CLASSIFY("Benign")}`

In the example above, a check is not made for a copy number event and therefore the statement by itself is not valid and will cause an error.

SCORE(DGV)

SCORE(DGV) looks at how similar the event in question is to events in the Database of Genomic Variants (DGV). As there are likely multiple reports of polymorphisms reported in the same region, the function looks at all of these and first identifies the most similar report in terms of direction (e.g., gain or loss) as well as similarity as defined earlier. It then takes the similarity score and scales based on the number of samples reported in the publication. For example, if evaluating the DGV Score for a loss event, the system might identify a publication that has reported 1 case with 100% similarity. In that case, the Similarity score would be approximately 65%. However, if there had been 50 such cases reported, the similarity score would be greater than 98% .

NOTE: The DGV score is only calculated for copy number events so one must first check to make sure that the event is a copy number event before checking for the DGV score or else an error will occur.

Example usage:

Correct usage: `CASE {ANY_CN_EVENT_KIND AND SCORE(DGV) > .85} {CLASSIFY("Likely Benign")}`
Incorrect usage: `CASE {SCORE(DGV) > .85} {CLASSIFY("Likely Benign")}`

NOTE: In the example above, a check is not made for a copy number event and therefore the statement, by itself, is not valid and will return an error.

The check for a copy number event must be performed each time a case statement evaluates the DGV score:

```
CASE {ANY_CN_EVENT_KIND AND SCORE(DGV) > .95} {CLASSIFY("Benign")}
   CASE {ANY_CN_EVENT_KIND AND SCORE(DGV) > .85} {CLASSIFY("Likely Benign")}
```

SCORE(EVIDENCE)

SCORE(EVIDENCE) looks at the Evidence Score for events that have this value. The Evidence Score is calculated if a sample has HPO terms associated with it. The Evidence Score for an event is the number of genes in the region with a matching phenotype.

Example usage:

```
CASE {SCORE(EVIDENCE) >= 5} { … }
```

NOTE: If the event contains five or more genes that match a sample phenotype, then...

```
CASE {(OVERLAP("OMIM Morbid Phenotypes Dominant") > 0) AND (SCORE(EVIDENCE) > 1)} {CLASSIFY("Likely Pathogenic")}
```

NOTE: If the event involves a dominant OMIM morbid gene matching HPO terms associated with the case, then classify as “Likely Pathogenic.”

Syntax for structural variants (SV)

Evaluating the type of structural variant event

The following functions assess the type of structural variant event.

ANY_SV_EVENT_KIND
Looks for any structural variant event.

Example usage:

```
IF { ANY_SV_EVENT_KIND } THEN { CLASSIFY(“Benign”) }
CASE { ANY_SV_EVENT_KIND } { CLASSIFY(“Benign”) }
```

SV_EVENT_KIND IS
Checks for the type of structural variant event.

Valid values for SV_EVENT_KIND

- Insertion
- Deletion
- Inverted_duplication
- Tandem_duplication
- Inversion
- Inversion_breakpoint
- Interchr_translocation
- Intrachr_fusion

Example usage:

```
CASE { SV_EVENT_KIND IS Intrachr_fusion } {CLASSIFY("Benign") }
CASE { SV_EVENT_KIND IS Tandem_duplication } {CLASSIFY("VUS") }
CASE { SV_EVENT_KIND IS Interchr_translocation } {CLASSIFY("Likely Benign")}
```

Comparing Events to Region Lists and Evaluating Similarity Scores

EVENT_OVERLAP

This function is like the EVENT_OVERLAP function described above.

Example usage:

```
CASE { (ANY_SV_EVENT_KIND) AND (OVERLAP("Imprinted Genes") > 0) } { CLASSIFY("Tier 1A") }
```

FUSION_MATCH

Tests to see if any of the fusion breakends (including confidence intervals) overlaps with fusion regions in a specified region list. The input to FUSION_MATCH() is a text string that is the name of an annotation list/track. A fusion is identified by two rows in the annotaion list. For example, the following entries identify a MLLT3 and KMT2A translocation.

```
chr9    20341669 20622499 t(9;11)_MLLT3::KMT2A_translocation
chr11   118436492        118526832        t(9;11)_MLLT3::KMT2A_translocation
```

The FUSION_MATCH() function will then determine if the fusion breakends of an SV event in question overlap with any fusion in the region list. The function returns true indicating that both breakends of the SV event. The function returns false if only one breakend or none of the breakends of the SV event overlap the fusions defined in the region list. This function is applicable to inter chr translocation and intra chr translocation and hence is applied to these functions as shown in the example below.

Example usage:

```
CASE {(SV_EVENT_KIND IS Interchr_translocation)}
        {
        CASE {FUSION_MATCH("AML Translocation Interchr", 3.Kb)} {CLASSIFY("Tier 1A-Review")}
        CASE {FUSION_MATCH("Pan Heme Translocation Interchr", 3.Kb)} {CLASSIFY("Pan-Heme Overlap")}
        }

CASE {(SV_EVENT_KIND IS Intrachr_fusion)}
        {
        CASE {FUSION_MATCH("AML Translocation Intrachr", 3.Kb)} {CLASSIFY("Tier 1A-Review")}
        CASE {FUSION_MATCH("Pan Heme Translocation Intrachr", 3.Kb)} {CLASSIFY("Pan-Heme Overlap")}
        }
```

Evaluating Sample Attributes

ATTRIBUTE()

This function evaluates values of specified factor names (attributes). Each sample can have several attributes, e.g., Gender, Tumor Type, Age, etc.

Allowed operators:

When comparing using numbers,

- **>** Greater than
- **<** Less than
- **>=** Greater than or equal to
- != Not equal to
- **<=** Less than or equal to
- **==** Equal to (**NOTE**: must use two equal signs in succession)

When the value being evaluated is a text string, the following operators are used:

- == Equals (returns a “true” or “false”)
- != Does not equal

Example usage:

```
CASE {ATTRIBUTE("Gender") == "male" } {…}
CASE {ATTRIBUTE("Age") < 30} {…}
```

Example Decision Tree Script

Below is a sample decision tree script using CASE statements for pre-classification of constitutional analysis. This script can be used as a template and further customized with in-house tracks from your own past cases.

```
/*
Using ACMG Guidelines for CNV calling of events not previously classified as Benign, Likely Benign, or Artifact more
than 4 times
Classifies CNV events as:
Benign
Likely Benign
VUS
Likely Pathogenic
Pathogenic
Artifact
Exclude

Classifies AOH events as
   */
   val Sim_threshold = 0.85

CASE {ANY_CN_EVENT_KIND} {

CASE {(SCORE(PREVIOUS_SIMILAR_CASES("Pathogenic", 0.85)) == 0) AND
```

```
(SCORE(PREVIOUS_SIMILAR_CASES("Likely Pathogenic", 0.85)) == 0) AND
(SCORE(PREVIOUS_SIMILAR_CASES("VUS", 0.85)) == 0)} {
CASE {SCORE(PREVIOUS_SIMILAR_CASES("Artifact", 0.95)) >=4} {CLASSIFY("Artifact")}
CASE {SCORE(PREVIOUS_SIMILAR_CASES("Benign", 0.95)) >=4} {CLASSIFY("Benign")}
CASE {SCORE(PREVIOUS_SIMILAR_CASES("Likely Benign", 0.95)) >=4} {CLASSIFY("Likely Benign")}
}
CASE {SCORE(PREVIOUS_SIMILAR_CASES("Pathogenic", 0.95)) >=1} {CLASSIFY("Pathogenic")}
CASE {SCORE(PREVIOUS_SIMILAR_CASES("Likely Pathogenic", 0.95)) >=4} {CLASSIFY("Likely Pathogenic")}
}

CASE {OVERLAP("RefSeq Genes") == 0} {CLASSIFY("Exclude")}

CASE {ANY_CN_EVENT_KIND AND SCORE(DGV)>0.88} {CLASSIFY("Likely Benign")} // Classify likely benign based on DGV
cases/score and absence of COE Paper genes as Likely benign No Reporting

CASE {ANY_CN_EVENT_KIND} {

CASE{SCORE(ACMG_CNV) >= 0.99} {CLASSIFY("Pathogenic")}
CASE{SCORE(ACMG_CNV) >= 0.9} {CLASSIFY("Likely Pathogenic")}
CASE{SCORE(ACMG_CNV) >= -0.89} {CLASSIFY("VUS")}
CASE{SCORE(ACMG_CNV) >= -0.98} {CLASSIFY("Likely Benign")}
CASE{SCORE(ACMG_CNV) < -0.98} {CLASSIFY("Benign")}
}
```

Tiered Variant Decision Tree for Hematological Malignancies

The latest release of VIA includes reference files with the annotation and tracks update for guideline targets, which are region coordinates that are derived from various global societies specific to Acute Myeloid Leukemia (AML), Myelodysplastic Syndrome (MDS), and an aggregation of all hematological malignancy guidelines (pan-heme). These resources, along with custom region files, can be used for a customizable decision tree for CNV, AOH, and SV event pre-classification. New installations of VIA include a standard decision tree for these three disease states that is constructed by Bionano to facilitate the identification of relevant events for researchers.

Schematic for disease-specific decision tree

A preconfigured decision tree is provided for customization by the site administrators, as necessary. Decision trees are constructed to preclassify SV and CNV events based on overlap to Heme-guideline target lists curated by Bionano . Variant pre-classification in VIA with the rules-based decision tree is serial, meaning the order of syntax is important as an event proceeds through the decision tree linearly until it is classified, or ultimately left unclassified. The classification schema outlined in **Figure 24** is an example and fully configurable to leverage differing classification states, resources, or tiering strategy to achieve site-specific objectives.

		SV Event Types							CN Event Types				LOH
	Guideline Target Files	Del	Ins	Inv	Inter	Intra	Tan Dup	Inv Dup	Loss	Null	Gain	Amp	AOH
	CNV Solve Mask								Artifact	Artifact	Artifact	Artifact	
Disease Specific	Deletion- Large								Tier 1A	Tier 1A			
	Deletion- Small	Tier 1A											
	Duplication- Large										Tier 1A	Tier 1A	
	Duplication- Small		Tier 1A				Tier 1A	Tier 1A			Tier 1A	Tier 1A	
	Inversion			Tier 1A									
	Interchrom Trans				Tier 1A - Review								
	Intrachrom Trans					Tier 1A - Review							
	Rearrangements - Large								Tier 1A	Tier 1A	Tier 1A	Tier 1A	
	Rearrangements -Small	Tier 1A	Tier 1A	Tier 1A	Tier 1A	Tier 1A	Tier 1A	Tier 1A					
	Monsomy								Tier 1A				
	Trisomy										Tier 1A		
Pan Heme	Deletion- Large								Pan-Heme Overlap	Pan-Heme Overlap			
	Deletion- Small	Pan-Heme Overlap											
	Duplication- Large										Pan-Heme Overlap	Pan-Heme Overlap	
	Duplication- Small		Pan-Heme Overlap				Pan-Heme Overlap	Pan-Heme Overlap			Pan-Heme Overlap	Pan-Heme Overlap	
	Inversion			Pan-Heme Overlap									
	Interchrom Trans				Pan-Heme Overlap								
	Intrachrom Trans					Pan-Heme Overlap							
	Rearrangements - Large								Pan-Heme Overlap	Pan-Heme Overlap	Pan-Heme Overlap	Pan-Heme Overlap	
	Rearrangements -Small	Pan-Heme Overlap	Pan-Heme Overlap	Pan-Heme Overlap	Pan-Heme Overlap	Pan-Heme Overlap	Pan-Heme Overlap	Pan-Heme Overlap					
	Monsomy								Pan-Heme Overlap				
	Trisomy										Pan-Heme Overlap		
CIViC	CIViC Genes	CIViC Gene	CIViC Gene	CIViC Gene	CIViC Gene	CIViC Gene	CIViC Gene	CIViC Gene	CIViC Gene	CIViC Gene	CIViC Gene	CIViC Gene	
Copy Number	Prev to Sim Tier 1B								Tier 1B	Tier 1B	Tier 1B	Tier 1B	
	Prev to Sim Tier 2								Tier 2	Tier 2	Tier 2	Tier 2	
	Prev to Sim Tier 3								Tier 3	Tier 3	Tier 3	Tier 3	
	Prev to Sim Tier 4								Tier 4	Tier 4	Tier 4	Tier 4	
	Prev to Sim Artifact								Artifact	Artifact	Artifact	Artifact	
	DGV Similar >88%								Tier 4	Tier 4	Tier 4	Tier 4	
	Unclassified												

Figure 24. Legend: Del – Deletion, Ins = Insertion, Inv = Inversion, Inter = Inter-chromosomal translocation, Intra = Intrachromosomal fusion, Tan Dup = Tandem Duplication, Inv Dup = Inverted Duplication, Null = Homozygous Loss, Amp = Amplification, LOH = Copy Neutral Loss of Heterozygosity

Decision tree script for disease specific tiering

Example syntax for decision tree to pre-classify CNV and SV variants obtained with OGM data with Bionano guideline files specific for AML as illustrated in the Decision Tree schematic above.

//
Event Classifications:
Tier 1A
Tier 1B
Tier 2
Tier 3
Tier 4
Tier 1A-Review
Pan-Heme Overlap

CIViC Gene
Artifact

*/
// CNV Masking overlap Artifacts

```
CASE {(ANY_CN_EVENT_KIND)}
    {
    CASE {EVENT_OVERLAP("Solve CNV Mask") > 0.45} {CLASSIFY("Artifact")}
    }
```

```
   // Gene Fusions disease-specific
   CASE {(SV_EVENT_KIND IS Interchr_translocation)}
      {
      CASE {FUSION_MATCH("AML Translocation Interchr", 3.Kb)} {CLASSIFY("Tier 1A-Review")}
      }

   CASE {(SV_EVENT_KIND IS Intrachr_fusion)}
      {
      CASE {FUSION_MATCH("AML Translocation Intrachr", 3.Kb)} {CLASSIFY("Tier 1A-Review")}
      }

   // Deletions disease-specific
   CASE {(SV_EVENT_KIND IS Deletion)}
      {
      CASE {EVENT_OVERLAP("AML Deletion Small", 3.Kb) > 0} {CLASSIFY("Tier 1A")}
      }

   CASE {(CN_EVENT_KIND IS CN_LOSS) OR (CN_EVENT_KIND IS HOMOZYGOUS_COPY_LOSS)}
      {
      CASE {SIMILARITY("AML Deletion Small") > 0.9} {CLASSIFY("Tier 1A")}
      CASE {SIMILARITY("AML Deletion Large") > 0.9} {CLASSIFY("Tier 1A")}
      }
```

// **Duplication disease-specific

```
   CASE {(SV_EVENT_KIND IS Tandem_duplication) OR (SV_EVENT_KIND IS Inverted_duplication) OR (SV_EVENT_KIND IS
      Insertion)}
      {
      CASE {EVENT_OVERLAP("AML Duplication Small", 3.Kb) > 0} {CLASSIFY("Tier 1A")}
      }
```

```
CASE {(CN_EVENT_KIND IS CN_GAIN) OR (CN_EVENT_KIND IS HIGH_COPY_GAIN)}
    {
    CASE {SIMILARITY("AML Duplication Small") > 0.9} {CLASSIFY("Tier 1A")}
    }


// **Monosomy disease-specific
CASE {(CN_EVENT_KIND IS CN_LOSS)}
    {
    CASE {SIMILARITY("AML Monosomy") > 0.9} {CLASSIFY("Tier 1A")}
    }


// **Trisomy disease-specific
CASE {(CN_EVENT_KIND IS CN_GAIN)}
    {
    CASE {SIMILARITY("AML Trisomy") > 0.9} {CLASSIFY("Tier 1A")}
    }


// **Rearrangements disease-specific
CASE {(ANY_SV_EVENT_KIND)}
    {
    CASE {EVENT_OVERLAP("AML Rearrangements Small", 3.Kb) > 0} {CLASSIFY("Tier 1A")}
    }

CASE {(ANY_CN_EVENT_KIND)}
    {
    CASE {SIMILARITY("AML Rearrangements Small") > 0.9} {CLASSIFY("Tier 1A")}
    }


// Gene Fusions Pan-Heme
CASE {(SV_EVENT_KIND IS Interchr_translocation)}
    {
    CASE {FUSION_MATCH("Pan Heme Translocation Interchr", 3.Kb)} {CLASSIFY("Pan-Heme Overlap")}
    }
```

```
CASE {(SV_EVENT_KIND IS Intrachr_fusion)}
    {
    CASE {FUSION_MATCH("Pan Heme Translocation Intrachr", 3.Kb)} {CLASSIFY("Pan-Heme Overlap")}
    }


// Deletions Pan-Heme
CASE {(SV_EVENT_KIND IS Deletion)}
    {
    CASE {EVENT_OVERLAP("Pan Heme Deletion Small", 3.Kb) > 0} {CLASSIFY("Pan-Heme Overlap")}
    }


CASE {(CN_EVENT_KIND IS CN_LOSS) OR (CN_EVENT_KIND IS HOMOZYGOUS_COPY_LOSS)}
    {
    CASE {SIMILARITY("Pan Heme Deletion Small") > 0.9} {CLASSIFY("Pan-Heme Overlap")}
    CASE {SIMILARITY("Pan Heme Deletion Large") > 0.9} {CLASSIFY("Pan-Heme Overlap")}
    }


// **Duplication Pan-Heme
CASE {(SV_EVENT_KIND IS Tandem_duplication) OR (SV_EVENT_KIND IS Inverted_duplication) OR (SV_EVENT_KIND IS
    Insertion)}
    {
    CASE {EVENT_OVERLAP("Pan Heme Duplication Small", 3.Kb) > 0} {CLASSIFY("Pan-Heme Overlap")}
    }


CASE {(CN_EVENT_KIND IS CN_GAIN) OR (CN_EVENT_KIND IS HIGH_COPY_GAIN)}
    {
    CASE {EVENT_OVERLAP("Pan Heme Duplication Small") > 0} {CLASSIFY("Pan-Heme Overlap")}
    CASE {EVENT_OVERLAP("Pan Heme Duplication Large") > 0} {CLASSIFY("Pan-Heme Overlap")}
    }


// **Monosomy Pan-Heme
CASE {(CN_EVENT_KIND IS CN_LOSS)}
    {
    CASE {SIMILARITY("Pan Heme Monosomy") > 0.9} {CLASSIFY("Pan-Heme Overlap")}
    }
```

```
// **Trisomy Pan-Heme
CASE {(CN_EVENT_KIND IS CN_GAIN)}
    {
    CASE {SIMILARITY("Pan Heme Trisomy") > 0.9} {CLASSIFY("Pan-Heme Overlap")}
    }


// **Rearrangements Pan-Heme
CASE {(ANY_SV_EVENT_KIND)}
    {
    CASE {EVENT_OVERLAP("Pan Heme Rearrangements Small", 3.Kb) > 0} {CLASSIFY("Pan-Heme Overlap")}
    }


CASE {(ANY_CN_EVENT_KIND)}
    {
    CASE {SIMILARITY("Pan Heme Rearrangements Small") > 0.9} {CLASSIFY("Pan-Heme Overlap")}
    CASE {SIMILARITY("Pan Heme Rearrangements Large") > 0.9} {CLASSIFY("Pan-Heme Overlap")}
    }


//Cancer Genes CNV
CASE {(ANY_CN_EVENT_KIND)}
    {
     CASE {EVENT_OVERLAP("CIViC Genes")>0} {CLASSIFY("CIViC Gene")}
    }


//Cancer Genes SV
CASE {(ANY_SV_EVENT_KIND)}
    {
     CASE {EVENT_OVERLAP("CIViC Genes", 3.Kb)>0} {CLASSIFY("CIViC Gene")}
    }


//Case History
CASE {(ANY_CN_EVENT_KIND)}
    {
    CASE {SCORE(PREVIOUS_SIMILAR_CASES("Tier 1B", 0.95)) >=1} {CLASSIFY("Tier 1B")}
    CASE {SCORE(PREVIOUS_SIMILAR_CASES("Tier 2", 0.95)) >=1} {CLASSIFY("Tier 2")}
    }
```

```
CASE {(ANY_CN_EVENT_KIND) AND
(SCORE(PREVIOUS_SIMILAR_CASES("Tier 1A", 0.85)) == 0) AND
(SCORE(PREVIOUS_SIMILAR_CASES("Tier 1B", 0.85)) == 0) AND
(SCORE(PREVIOUS_SIMILAR_CASES("Tier 2", 0.85)) == 0)}
   {
   CASE {SCORE(PREVIOUS_SIMILAR_CASES("Tier 3", 0.95)) >=4} {CLASSIFY("Tier 3")}
   CASE {SCORE(PREVIOUS_SIMILAR_CASES("Tier 4", 0.95)) >=4} {CLASSIFY("Tier 4")}
   CASE {SCORE(PREVIOUS_SIMILAR_CASES("Artifact", 0.95)) >=4} {CLASSIFY("Artifact")}
   }

// DGV
CASE {ANY_CN_EVENT_KIND AND SCORE(DGV) > 0.88} {CLASSIFY("Tier 4")}
```

Technical Assistance

For technical assistance, contact Bionano Genomics Technical Support.

You can retrieve documentation on Bionano products, SDS's, certificates of analysis, frequently asked questions, and other related documents from the Support website or by request through e-mail and telephone.

TYPE	CONTACT
Email	support@bionano.com
Phone	Hours of Operation: Monday through Friday, 9:00 a.m. to 5:00 p.m., PST US: +1 (858) 888-7663
Website	www.bionano.com/support
Address	Bionano Genomics, Inc. 9540 Towne Centre Drive, Suite 100 San Diego, CA 92121