# Bionano Solve Theory of Operation: Structural Variant Calling

DOCUMENT NUMBER:

CG-30110

DOCUMENT REVISION:

O

Effective Date:

01/23/2024

# Revision History

| REVISION | NOTES |
|---|---|
| **L** | Added updated graphs for low allele fraction performance. |
| **M** | Updated for Solve 3.8<br>• Update performance evaluation of SV and CNV calling with latest release<br>• Move Integer CNV and performance evaluation from previous versions to Appendices |
| **N** | Updated for Solve 3.8.1<br>• Added new Guided Assembly Pipeline |
| **O** | Updated to incorporate changes for the Stratys system |

# Introduction

Structural variation (SV) is a common source of genetic variation, involving deletion, insertion, and rearrangement of genomic material. It has the potential to impact large stretches of DNA sequence, disrupting genes and regulatory elements. Structural variants are associated with genetic disorders and are used as disease markers in clinical diagnosis of diseases such as DiGeorge syndrome and cancer.

Bionano Solve contains a suite of tools that analyze raw single-molecule data including a haplotype-aware assembler specifically designed to detect homozygous and heterozygous SVs and low variant allele fraction (VAF) variants. Bionano Solve 3.8.1 introduces the new Guided Assembly pipeline with optimized workflows for low-allele fraction and constitutional applications. Guided Assembly is offered as an alternative to de novo assembly and Rare Variant Analysis (RVA). All major SV types are supported, and extensive validation based on simulated and experimental data showed high detection performance.

Large copy number variants are detected based on coverage depth information using a copy number analysis pipeline embedded in all genome analysis pipelines. The copy number analysis pipeline can detect fractional copy number changes and chromosomal aneuploidy events.

Variant Annotation Pipeline (VAP) annotates the SV calls and provides information key to understanding relevance to a biological question or phenotype of interest (see Bionano Solve Theory of Operation: Variant Annotation, CG-30190). It supports analysis of paired datasets (tumor-control, for example) and trio datasets (proband and two parents, for example).

Bionano Solve is fully integrated with Bionano Access™, which provides a user-friendly interface for streamlining analysis. Bionano Access is useful for managing projects, analyzing run results, and visualizing data.

# Structural Variant Calling with Bionano Solve

### *De novo* Assembly

Bionano's de novo assembly algorithm is built on the overlap-layout-consensus strategy with a maximum likelihood model for scoring alignments. For more information, please refer to Appendix D. Following pairwise alignment of the input single-molecule maps, an overlap graph is constructed. Spurious edges are removed, and redundant edges collapsed. The assembler outputs the longest paths in the graph and constructs a set of draft consensus maps. The consensus maps are further refined, extended, and merged.

To optimize assembly of non-homozygous variants, during the extension stages of the assembly, the pipeline analyzes molecule-to-genome map alignments, identifies clusters of molecules with coordinated disrupted alignment, and assembles these clusters separately. This is critical for assembling haplotype maps with significant differences and for detecting a wide range of variants. Additionally, we implemented haplotype-aware components to optimize assembly of heterozygous variants. In the last refinement stage, molecules are aligned to a given genome map and clustered into two alleles. The allele-specific molecules are used to generate the final set of allele-differentiated consensus genome maps.

During assembly, large non-unique regions (which we call complex multi-path regions, or CMPRs) in the genome maps are recognized and marked. These regions, often associated with large segmental duplications in the genome, create ambiguity in the assembly graph and are prone to mis-assembly. They are detected in a de novo fashion - assembled maps are aligned with each other, and maps that share significant stretches of sequence but are otherwise divergent are identified. If the CMPR, or the shared sequence, is at least 140 kbp, and if the option to split CMPRs is enabled, the maps are split (**Figure 1**) to avoid mis-assembly. The labels encompassing the CMPR would be marked in the Mask column in the CMAP output (see OGM File Format Specification Sheet, CG-00008) and highlighted in Bionano Access. Users are also provided the option to not split these maps; the options are presented in Access when an assembly run is set up.

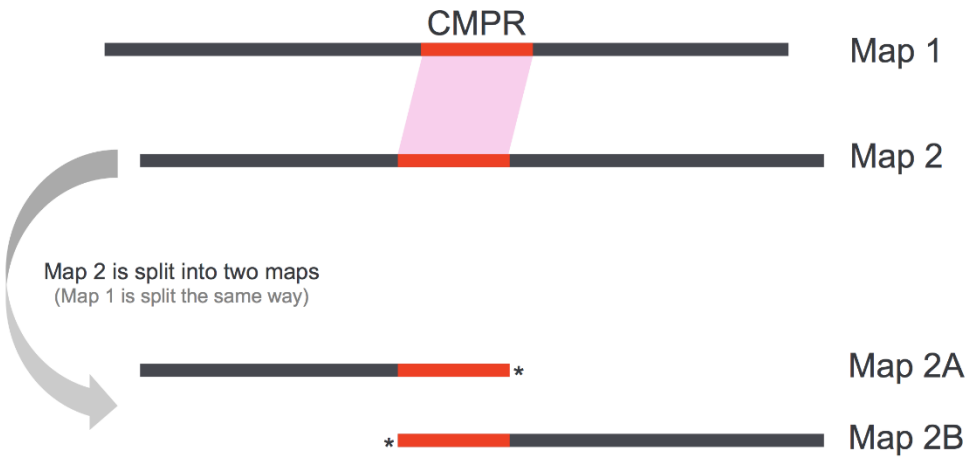



**Figure 1**. Schematic of how maps that share a CMPR (highlighted in red) are split. The sequences flanking the CMPR are divergent between Map 1 and Map 2. The resulting Map 2A and Map 2B each contain a copy of the CMPR. CMPR ends are marked (* in figure) such that the maps would not be merged in subsequent steps.

## SV Calling

SV calls are obtained by aligning consensus genome maps to a reference using a Multiple Local Alignment algorithm and analyzing the alignments for SV signatures. Pairs of alignments within a map are analyzed and inconsistencies representing possible SV events between the genome maps and the reference are identified.

### INSERTIONS AND DELETIONS

An alignment outlier is defined by two well-aligned regions that flank a poorly aligned or unaligned region. An outlier is identified as a deletion if the reference range in the outlier region is larger than the corresponding range on the map, and an insertion if the converse is true. Sequence substitutions may also create alignment outliers without significant net gain or loss of sequence. In those cases, insertion, and deletion calls (often small) may be called, even though they may not be simple insertions or deletions.

### TRANSLOCATION BREAKPOINTS

A fusion point between distant regions of the genome is identified as a translocation breakpoint. Intrachromosomal fusion breakpoints involve regions typically (but not always) at least 5 Mbp away from each other *on the same chromosome*. Interchromosomal translocation breakpoints involve regions *on different chromosomes*. Translocation breakpoint orientation is denoted in the “orientation” column in the SMAP output (**Figure 2**). Reciprocal translocations are expected to be detected as separate translocation breakpoints with opposite orientations. The pipelines do not explicitly pair translocation breakpoints. Also, transpositions of small pieces of sequence (typically less than 50 kbp) may be detected as insertions, if the transposed sequence cannot be confidently aligned to the reference.

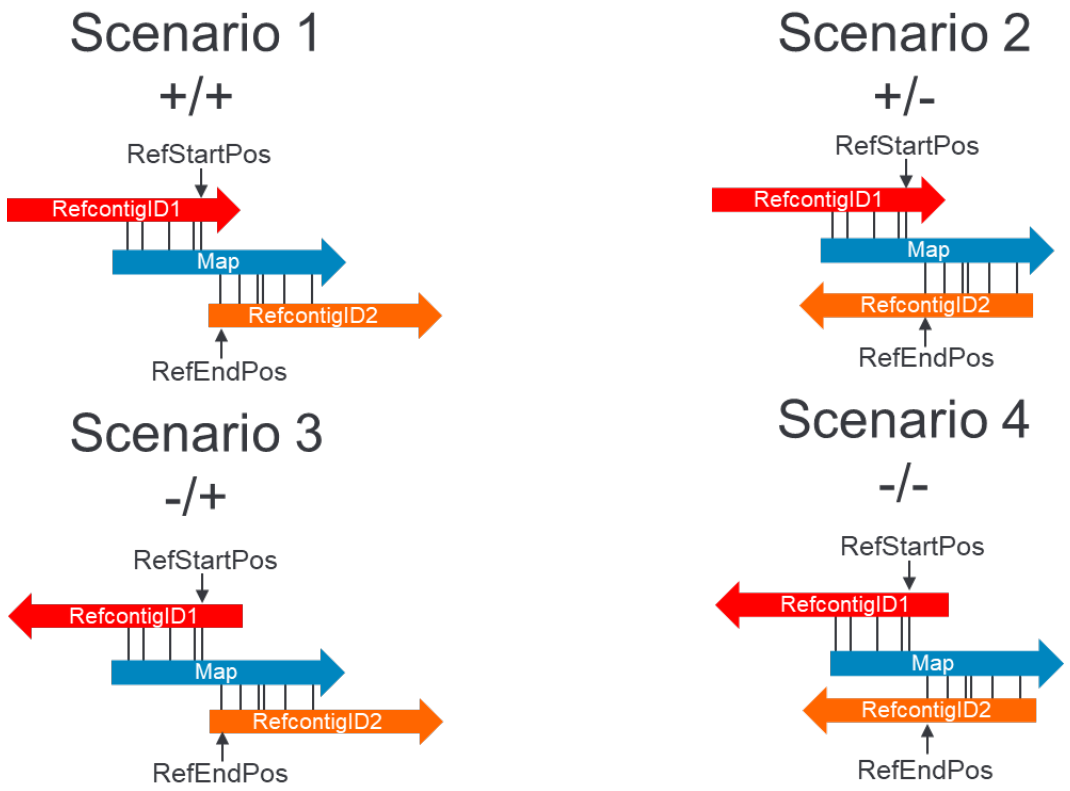



**Figure 2**. Schematic of how translocation breakpoint orientation is defined in SMAP. There are four possible scenarios. Each translocation breakpoint is annotated in one of the four ways: "+/+," "+/-", "-/+," and "-/-". The first sign corresponds to the orientation of the alignment of map to the (RefcontigID1, RefStartPos) reference genome coordinate tuple. Given this definition, a '+' for the *first* sign entails that the reference alignment occurs in the *upstream* direction from the reference breakpoint. The second sign corresponds to the orientation of the alignment of the map to the (RefcontigID2, RefEndPos) reference genome coordinate tuple. A '+' for the *second* sign entails that the reference alignment occurs in the *downstream* direction from the reference breakpoint. The opposite reference alignment direction relative to the breakpoint is also entailed by a '-' in both positions of the SMAP orientation field.

### INVERSION BREAKPOINTS

Inversion breakpoint calls involve neighboring alignments with opposite orientations. Please refer to Appendix E for more information interpreting inversion breakpoint calls. Small inversions (whose inverted regions contained fewer than five labels) are identified by searching in a limited space for potential inverted alignments. They may be spanned by single genome maps and represented in sets of two linked, paired inversion breakpoint entries ("`inversion_paired`"), specifying eight coordinates of interest. For other inversion breakpoint calls, there are two linked SMAP entries ("inversion" and "`inversion_partial`"), specifying six coordinates of interest (**Figure 3).** If the inverted alignment overlaps partially with the neighboring non-inverted alignment, the inverted alignment is trimmed. Inversions with a reference gap larger than 5 Mbp are called as intra-chromosomal fusion breakpoints, provided there was no query alignment overlap of 140kb or more. Insertions or deletions located inside the inversion breakpoint are located at the correction breakpoint interval: in previous release the deletion or insertion would appear to span th0e entire inversion region including both breakpoints.

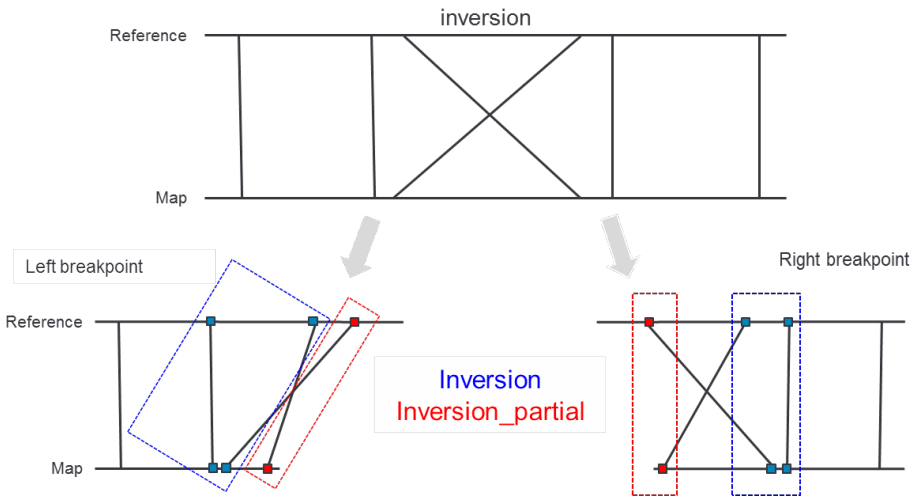



**Figure 3**. Schematic of how inversion breakpoints calls are output in SMAP. "`Inversion`" (highlighted in blue) and "`inversion_partial`" (highlighted in red) entries are linked, and together, they encode six coordinates of interest.

## DUPLICATIONS

Duplications are detected based on direct or indirect evidence of duplication within single maps. There is direct evidence when two stretches of sequence on the (consensus) map align to the same stretch of sequence on the reference, implying that there are (at least) two copies of the same reference sequence on the map (**Figure 4**). They are output as "duplication" events. However, sufficiently large duplications need to be detected differently. In those cases, the map may not span whole copies of the duplicated sequence. Based on the alignment signature, one could still infer duplication events, even though two full duplicated copies may not be observed on the map, and the alignments may not overlap on the reference (**Figure 5**). They are output as "`duplication_split`" events. The duplication detection algorithm supports tandem duplications and non-tandem duplications whose copies are sufficiently close to each other on a given map OR both copies are fully spanned by the same map (NOT a `duplication_split`). The previous descriptions are relevant for duplications where both copies of the duplicated sequence have the same orientation; they are named differently to highlight the distinction between the detection methods. Inverted duplications are called if the two alignments on the map are in opposite orientations. Size cannot be determined for inverted duplications unless the map fully spans both copies.

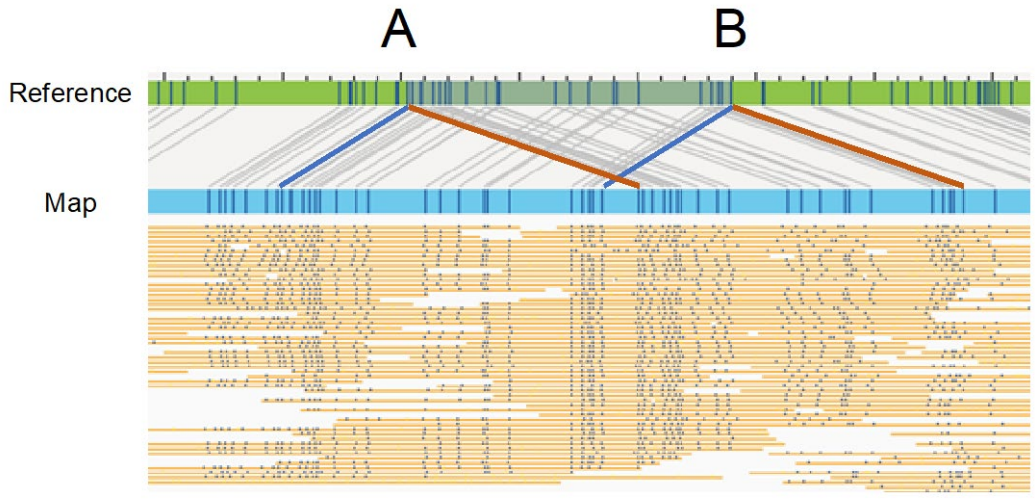



**Figure 4**. Example of a “duplication” event. Two regions in the map (one shaded in blue, and one in red) align to the same region on the reference (the region between points A and B). This suggests that there are two copies of the sequence in the map, and that they were next to each other in the same orientation.

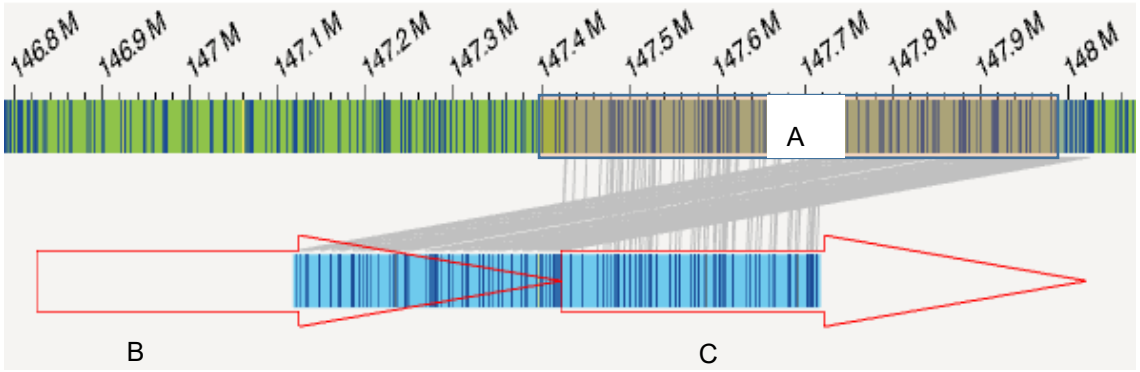



**Figure 5**. Example of a “`duplication_split`” event. In this case there is no overlap in the alignments on the map. However, based on the alignment, segment B on the map, represented by the red arrow on the left, is fused with segment C represented by the arrow on the right. The parsimonious interpretation would be that there is a direct tandem duplication of the sequence in region A. The direction of the arrows shows the orientation of the segments. The outline areas of the arrows show the portion of segments B and C that are inferred and do not appear in the assembled map. While the evidence is indirect and the event inferred, one may confirm such signal using the copy number analysis data (for large duplications).

## LARGE SV TYPES

Solve uses sizing criteria to assign types to large SVs > 200 kbp. Please see **Table 1** for details of the size conditions and the alternative assigned types.

**Table 1**. Summary of type assignment for large SVs

| Candidate Type | Criteria | SV Type Called |
|---|---|---|
| **Duplication** | Reference distance between fusion points plus the size of duplication > 5Mbp | Intra-chromosomal fusion |
| **Duplication** | Reference distance between fusion points is greater than 300 Kbp and less than 5Mbp | Duplication |
| **Deletion** | Deletion size > 5Mbp | Intra-chromosomal fusion |
| **Inversion** | Reference distance between fusion points > 5Mb | Inversion |
| **Intra-chromosomal fusion** | Reference distance between fusion points < 1Mb | Intra-chromosomal fusion |

## Zygosity Classification

Zygosity is a classification of an SV call as homozygous, heterozygous, or unknown. It is currently assigned to insertion, deletion, translocation breakpoint, and inversion breakpoint calls. Conceptually, we try to determine whether there is presence of a reference allele or additional SV alleles. To do that, we look at two criteria: 1) whether the call-in question overlaps another call (which might be from a different SV allele), and 2) whether the call in question overlaps alignment of another genome map that shows no SV (which might be from the reference allele).

In the case of overlap with another call, there is a determination of whether the two calls are likely to represent the same allele. An SV call is categorized as homozygous if there is no overlapping alignment or the same SV is called on another genome map. An SV call is categorized as heterozygous if there is overlapping alignment or a different SV is called on another genome map. If both another alignment and a different SV or multiple different SVs are present at the same location, zygosity is set as heterozygous. For determining if SV calls are the same, different sets of criteria are applied for different SV types. For insertion and deletion calls, they must have overlapping ranges and at least 80% overlap and size similarity. For translocation breakpoint calls, the breakpoints must be within 50 kbp and on the same strand and chromosomes. For inversions breakpoints, the breakpoints must be within 50 kbp and on the same strand and chromosomes. The pipeline does not require exact breakpoint and size matches, to account for slight differences.

Zygosity is not currently assigned to duplication calls. Also, zygosity is not an output for calls from the Rare Variant Analysis pipeline.

## AOH/LOH Detection

### CNV CALLING IN VIA WITH SNP-FASST3

Analysis of OGM data in VIA software includes the capability to apply a new algorithm, SNP-FASST3, for the detection of CNVs and AOH. Detailed description of the algorithm concept and performance data leveraging the simulated data is provided in the document *VIA software Theory of Operations* (CG-00042). Following is a detailed description of the AOH detection algorithm which runs as part of Solve whole genome analysis pipelines. Results from this analysis are presented in Access.

### INTRODUCTION

The *de novo* assembly and guided assembly (constitutional) pipelines contain a module for detecting regions of absence/loss of heterozygosity (AOH/LOH) in human samples. Absence of heterozygosity refers to a specific type of genetic mutation that can be a result of uniparental disomy or consanguinity, in which there is an absence of one normal copy of a gene, and which may result in increased susceptibility to recessive disease. Regions of homozygosity are identified by a consistent decrease in heterozygous SV calls across a genomic region in the case sample compared to the level observed genome-wide in controls. SV zygosity is determined as part of the constitutional analysis pipelines; therefore, AOH detection is only available for the *de novo* and guided assembly pipelines.

As part of the standard assembly pipeline output, the results from the AOH detection pipeline are stored in `contigs/exp_refineFinal1_sv/merged_smaps/loh/`. There are two expected output files in the directory. Selected results files are imported by Bionano Access; the data are plotted in the AOH/LOH track below the CN track in the map-to-reference alignment view and in the whole-genome CN view. In these views and in the Circos view, AOH/LOH calls are denoted as yellow highlighted regions in the SV track. Details about the calls are listed in the AOH/LOH Regions tab, where users can sort and filter the calls.

***loh_calls.txt*** contains information about the AOH calls, specifically the start and end positions and a confidence score. The confidence score is based on the size of the call, and it represents the model's precision for calling AOH events in the given size range bin. The confidence score is directly correlated with the size of the call. **NOTE:** AOH/LOH calls with a size under 25 Mbp are not output, as the model has low precision in this size range, unless the call is near another call (within 25 Mbp), since a true AOH/LOH region may be called as multiple smaller regions with a small gap between. Further size filtering can be applied in Bionano Access.

***loh_per_sv_info.txt*** contains details about the subset of SVs from the input SMAP file that were used in AOH/LOH detection, with a few additional columns containing information about the SVs that were obtained from the HMM, such as the probability that the SV is in an AOH/LOH region, as calculated by a Hidden Markov Model (details in the following section). The SVs listed in this file are those that remained after variant filtering and were the SVs used in AOH/LOH calling. Only these SVs will have a corresponding orange dot in the Bionano Access AOH/LOH track denoting the probability that the SV is in an AOH/LOH region.

Please refer to *OGM File Format Specification Sheet* (CG-00045) for complete details on AOH/LOH output files.

### THEORY

AOH regions are determined using a Hidden Markov Model (HMM), in which the hidden state is whether an SV belongs to an AOH or background region, and the observable states are whether the SV was called as homozygous or heterozygous. Model parameters were estimated by fitting the model to a simulated dataset that

was generated by splicing together SMAP files from Bionano control samples and two haploid samples. Haploid samples are expected to have only homozygous SVs, except for false positive heterozygous SV calls and are the closest approximation to how an AOH region would appear in a real sample. The fact that males have AOH regions across chromosome X, except for the pseudo autosomal region, was accounted for and used in simulating AOH events and training the HMM.

Additional variant filtering was applied to enrich the data for informative sites. The following SVs were filtered out: confidence score below 0.95; unknown zygosity; intrachromosomal SVs; duplicate SVs where the start and end position were identical. Finally, SVs were filtered out if they were known commonly homozygous sites, where at least 95% of Bionano controls had homozygous SV calls in the same 10 kbp region. If a bin contained a known common homozygous SV, all SVs in the same bin were filtered out in the query sample.

Performance was evaluated using samples with known AOH/LOH events and using additional simulated data. It was observed that large AOH regions may be called as multiple smaller AOH calls with small gaps between, if there are false positive heterozygous SV calls in the region. On the other hand, many small AOH calls appear to be false positive, and these tend to stand alone and are often the only AOH call on a chromosome. Therefore, the following size-based filtering is applied to calls that are output: AOH calls under 25 Mbps are filtered out if they are at least 25 Mbps away from another AOH call. This was found to filter out approximately 90% of false positive calls while retaining true AOH calls.

**Known issues:**

- In the Bionano Access whole genome plot, we have observed cases where the AOH algorithm results and the segments appearing in the VAF plot do not agree. If this occurs, the AOH call should be prioritized. Please see the Variant Allele Fraction (VAF) Calculation section for details.
- Homozygous SVs that were filtered are displayed in the AOH/LOH track in Bionano Access as green dots without a corresponding orange dot representing the confidence score. AOH/LOH calls that have a large number of homozygous calls without accompanying confidence scores may be indicative of a false positive.

### Confidence Modeling

The raw SV calls are made based solely on interpretation of the map alignments; however, other factors also impact whether the SVs are truly present in a sample. For example, SV calls may be less reliable in complex regions, where there may be conflicting alignments. The SV confidence scores are intended to provide a measure of “confidence” of the SV calls and a way for users to sort, filter, or prioritize the calls.

The SV confidence scores are computed based on models that were trained using simulated, isolated, non-complex SVs and detected SVs from real samples where orthogonal data were available. Real SVs in a genome tend to cluster and are found in certain structurally complex regions. They are difficult to simulate, and the simulation data do not capture the full range of complexities observed in complex regions and in real samples. Thus, the confidence scores may not be as informative and reliable for SV calls in those regions. As discussed in the following sections, the default confidence cutoffs in Bionano Access serve to provide a starting point. When selecting the recommended thresholds, we tried to balance sensitivity and the number of false positive calls. We looked at each SV type using simulated data and real data (for which we had orthogonal SV data) for this process, and we also considered feedback from internal teams and the field. Users may experiment with other cutoffs and vary the stringency based on project needs. For example, adjusting the cutoff upward should increase the fraction of true calls, but other lower confidence true calls may be filtered out as a result.

Users are encouraged to use confidence scores as one of the possible tools for filtering SV calls. The confidence scores may be used in conjunction with the SV masks and variant annotations.

#### CONFIDENCE FOR INSERTIONS AND DELETIONS

The confidence of an insertion or deletion call, which ranges from 0 to 1, reflects an estimate of the probability of the call being a true positive, or positive predictive value (PPV). These estimates are derived based on simulation studies, where we simulated insertions and deletions in the genome and assessed the resulting SV calls. It considers the SV size, the non-normalized p-value (log10) of the two well-aligned regions, and the non-normalized log-likelihood ratio of the poorly aligned or unaligned region. The genomic context is not explicitly considered; confidence scores of calls around complex regions may be less reliable. The confidence cutoff in Bionano Access is defaulted to 0 for insertions and deletion calls. Calls with lower confidence may be of interest in a discovery setting; however, users may increase the stringency based on project needs.

Confidence for insertion and deletion calls <500 bp in size is undefined and set to -1.

#### CONFIDENCE FOR INVERSION AND TRANSLOCATION BREAKPOINTS

Inversion and translocation breakpoints are scored using a machine learning model trained on a combination of simulated and real human data. A distinct set of models is applied for non-human datasets. For Bionano Solve 3.8, a new minimum confidence threshold has been set for translocations. Users are encouraged to develop custom score cutoffs for the non-human models.

The recommended score cutoffs for the human models are:

- Intrachromosomal fusion breakpoints: 0.02
- Interchromosomal translocations breakpoints: 0.02

Further discussion on the scoring model is included in Appendix F.

#### CONFIDENCE FOR DUPLICATIONS

Currently, confidence for duplications called by the SV calling pipeline is undefined and set to -1. Inspection of duplication calls is encouraged.

### Variant Allele Fraction (VAF) Calculation

Bionano Solve estimates the VAF for each SV detected by the *de novo,* guided assembly and RVA pipelines. Its value ranges from greater than 0 to 1 and records the percentage of molecules calling the structural variant out of the total coverage in the region. In a normal diploid sample, a homozygous SV will have a VAF approaching 1 while a heterozygous variant will have a VAF of approximately 0.5. A value of -1 is provided when the algorithm could not find a value or there was a processing error. The values are present by default in the output SMAP under the VAF column. A parameter is available in the command line tool to disable the computation.

The VAF calculation uses the coverage of the consensus map segments and consists of two stages. The first stage clusters the SV calls in the SMAP file to find equivalent ones (those that are the same SV event detected by two or more different consensus maps). In the second stage the coverages for the maps calling the same SV are aggregated using the alignments of the consensus maps to the reference as a guide. The aggregated coverage is used as input to a Bayesian inference that determines the probability of having a VAF value $\propto_k$ for an allele *k*

given the coverage for a set of labels *D* in the SV region, and the genotype *G*: $P(\propto_k |D, G)$. The value that maximizes the probability is chosen as the VAF. For duplications, it is not possible to apply Bayesian inference and the VAF is simply the quotient between the SV coverage and the total coverage at the genomic locus.

We also provide a segmentation algorithm to help distinguish changes in the VAF pattern across the genome, which is useful to detect aneuploidies. Further details are provided in Appendix H.

**Known issues:**

- In the whole genome plot, we have observed cases where the AOH algorithm results and the segments appearing in the VAF plot do not agree (e.g., AOH is called and VAF show variants with allele fractions <1). This may occur because the analyses are performed independently using different rules for which variants to include in the calculation. For AOH detection only variants with extremely high confidence scores are considered, while for the VAF segmentation all variants are employed. This is done as a trade-off to improve genome-wide segmentation accuracy in samples at the potential cost of discordant results in AOH regions. When observing a possible conflict between an AOH region and a VAF segment line created by some heterozygous variants, we recommend guiding your analysis based on the AOH results.

## Copy Number Variant (CNV) Calling

### CNV CALLING IN VIA WITH SNP-FASST3

Analysis of OGM data in VIA software includes the capability to apply a new algorithm, SNP-FASST3, for the detection of CNVs and AOH. Detailed description of the algorithm concept and performance data leveraging the simulated data is provided in the document *VIA software Theory of Operations* (CG-00042). Following is a detailed description of the FractCNV algorithm which runs as part of Solve whole genome analysis pipelines. Results from FractCNV are presented in Access.

### INTRODUCTION

Copy number (CN) analysis is performed as part of all whole genome analysis pipelines. The CN analysis tool analyzes an input molecule-to-reference alignment, normalizes this raw molecule coverage profile using control data, and segments the genome (inferred local CN states across the genome) based on detected changes in the underlying copy number state. CNV calls are output and annotated with confidence scores. Chromosomal aneuploidy events are detected after post-processing the initial CNV calls; however, whole-genome aneuploidy events cannot be detected.

The CN analysis tool has two main components – a fractional CN analysis module and an integer CN analysis module. The fractional CN module is optimized for detecting events in genomes with multiple CN state changes and events at lower allelic frequencies (AF)[1]. These events are frequently observed in highly heterogeneous genomes such as cancer. The fractional CN pipeline is intended for DLE-1 datasets. The integer CN module is intended for human Nt.BspQI and Nb.BssSI datasets and for homogenous genomes. The fractional and integer modules are integrated into the pipeline; there is not an explicit switch to use one or the other. The pipeline automatically determines which one to use based on the input data and available auxiliary data. DLE-1 datasets

[1] Allele frequency (AF) is proportional to the fraction of chromosomes in a sample that carry given events.

are expected to go through the fractional module which is discussed in detail in the following. For theory and performance data on the integer CN modules, see Appendix J.

Necessary auxiliary input for the standard human reference builds (hg19, hg38, and T2T-CHM13v2.0) and mouse genomes (mm10 and mm39) is packaged with the tool. With user-provided data, the pipelines can also support additional reference and enzyme combinations.

The tool was implemented in the R programming language. It can be run on the command line as a standalone tool, or as part of a *de novo*, guided assembly or RVA run. The resulting compressed output is compatible with and can be visualized in Bionano Access after import[2]. The normalized coverage profile is plotted in Bionano Access, and the CNV calls can be sorted and filtered in the Copy Number tab.

### INPUT

The pipelines automatically generate and use the necessary input for the CN analysis for supported input datasets. A reference needs to be specified at the start of the pipelines such that the molecule-to-reference alignment is generated.

Instead of using default data packaged with the tool, users could provide custom control data. For command-line usage, the minimum required input in the default mode includes the r.cmap and .xmap output from the molecule-to-reference alignment (`alignmolvref`) stage of the pipeline, a name for the sample, and an output path for the results.

### OUTPUT

As part of the standard pipeline output, the results from the CN analysis tool are stored in `alignmolvref/copynumber/`. There are several expected output files in the directory. Selected results files are imported by Bionano Access; the CN data are plotted in the CN track above the reference map in the map-to-reference alignment view, in the Circos view, and in the whole-genome CN view. Details about the CNV calls are listed in the Copy Number tab, where users can sort and filter the calls. Aneuploidy calls are in the Aneuploidy tab.

`cnv_rcmap_exp.txt` contains per-label coverage information. The format of this file is like the standard CMAP format, but with several additional columns, the definitions of which depend on which pipeline is run (see **Table 2** below).

**Table 2**. Per-label Coverage Information

| Column Name | Fractional CN Pipeline |
|---|---|
| **ScaledCoverage** | Sample coverage divided by average control coverage |
| **NormalizedCoverage** | Same as above (to maintain consistent format) |

[2] Results from a standalone, command-line CN analysis run may not be directly imported into Access at this time.

| Column Name | Fractional CN Pipeline |
|---|---|
| **CopyNumber** | Rounded copy number states |
| **fractionalCopyNumber** | Fractional copy number states |
| **MeanCov** | Average coverage |

The fractional **CopyNumber** and **CopyNumber** columns are the same as those in `cnv_calls_exp.txt`. **ScaledCoverage** and **NormalizedCoverage** are considered intermediate results (see "Theory" section below for detail). The CN-related columns reflect local changes in the CN state. For human, the normal diploid CN state is 2. CN states of 0 and 1 typically correspond to homozygous and heterozygous sequence loss, respectively. CN states of 3 and higher correspond to sequence gain.

**`cnv_calls_exp.txt`** contains the start and end positions of CNV calls (those whose CN states differ from baseline). The **fractionalCopyNumber** column contains the scaled and smoothed CN state for a given CNV call, and the CopyNumber column is rounded from **fractionalCopyNumber**. For the fractional CN pipeline, confidence is 1 minus the probability of observing the mean coverage of a segment if the segment had a baseline CN state.

**`cnv_calls_exp_full.txt`** is like `cnv_calls_exp.txt`. This file contains the combined results from the integer and fractional CN modules. An additional column named "Algorithm" denotes which module a call is from. Calls from the fractional CN module are denoted as "Region-based."

`cnv_chrAneuploidy.txt` contains per-chromosome aneuploidy calls. The **fractChrLen** column indicates the fraction of a given chromosome that was consistent with an aneuploidy event. The score column is the weighted probability of observing the segments in the aneuploidy event assuming the segments had baseline CN states. The fractCN column indicates the likely aneuploidy state of the chromosome.

`cnv_chr_stats.txt` contains more detailed per-chromosome statistics. The noise statistics for a sample and the inferred sex of the sample are noted in the header section.

**`cnv.log`** contains information about the CN analysis run. Warning and error messages may be useful for troubleshooting purposes.

### ENVIRONMENT SETUP

The pipeline was implemented with R version 3.6.1. and requires publicly available R packages and two Bionano-specific packages (`BionanoR` **and** `FractCNV`). For installation help, see *Bionano Solve Installation Guide* (CG-30182).

## USAGE

**Table 3**. The *de novo* assembly pipeline and RVA call the main R wrapper script "CNV.R" to perform CN analysis. The CN analysis is typically run using default Bionano-supplied control data. Additional parameters to the *de novo* assembly and RVA may be added, as shown below. The parameters for the fractional CN analysis module (cr, cm, and ce) can be supplied in the Bionano Access interface.

| Module | Parameter | Notes (corresponding parameters in CNV.R in the first line of each entry) |
|---|---|---|
| **Fractional** | **cr** | --controlRef [file]<br>User-provided control data for the fractional copy number pipeline. This should be the output file generated from running the script generate_controlFractCNV.R (see "Pre-process custom control data"). |
| | **cm** | --cnvMask [file]<br>User-provided CNV mask BED file for the fractional copy number pipeline. This should be the output file generated from running the script generate_cnv_mask_FractCNV.R (see "Pre-process custom control data"). |
| | **ce** | **--chrExpectedCopyNumbersFile [file]**<br>Tab-delimited text file describing the expected copy numbers for each chromosome, for different sexes. The required column headers are cmapid, female, and male. Any chromosome ID missing from the table is assumed to be diploid for all sexes. See examples of this file in /Pipeline/Analysis/FractCNV/data/. An example of a sufficient chrExpectedCopyNumbersFile for human is shown below:<br>cmapid female male<br>23 2 1<br>24 0 1<br>**NOTE:** The parameters sexChrType, *sexChr1* and *sexChr2* have been deprecated and are no longer used. The parameter chrExpectedCopyNumbersFile replaces these. |

**Table 4**. To run CNV.R as a standalone operation, please refer to the run_default.sh as an example in the main CN analysis code directory in the Bionano Solve build. The following input parameters may be used for additional flexibility when calling the CNV.R wrapper script.

| Module | Parameter | Notes |
|---|---|---|
| **General** | --resultsDir [dir] | Path to output from the CN analysis pipeline. The default path is alignmolvref/copynumber in the pipeline output directory. |
| | --sampleRcmap [rcmap] | Path to input r.cmap from the molecule-to-reference alignment. The default path is alignmolvref/merge/alignmolvref_merge_r.cmap. |
| | --sampleXmap [xmap or dir] | Path to input .xmap file(s) from the molecule-to-reference alignment. This can be the path to single xmap file that contains all the molecules-to-reference alignments, or a directory path to the “alignmolvref/merge” folder in pipeline output directory. The pipeline would retrieve the relevant xmap files from the directory path. |
| | --sampleName [string] | Sample name. The default is “exp.” |
| | --chrRemove [string] | Chromosome ID in the CMapId column of r.cmap to exclude from the analysis. The format is the same as componentsRemoved. |
| **Integer** | --testingDir [dir] | Path to simulated data used for recalculating the confidence tables for the integer CN analysis module. The default path is /testing_data in the CN analysis pipeline package. |
| | --componentsRemoved [string] | Principal components derived from control coverage profiles to be removed during normalization. The format is a:b where principal components a to b are removed, or a,b,c where principal components a, b, and c are removed. |
| | --controlSdCutoff [float] | The upper threshold of scaled coverage standard deviation for selecting control datasets. |

## THEORY

The main steps of the fractional CN pipeline are:

- Normalization
- Segmentation
- Baseline estimation
- CNV detection and confidence score computation
- Chromosomal aneuploidy detection

### Normalization

This pipeline takes advantage of control data to model and reduce variation unrelated to true CNV events in raw coverage data. The set of control samples that are used are assumed to have very few or no true large CNV events. First, the coverage data is pooled from the control samples by summing the raw coverage at each label across the samples. The summed coverage at each label is then divided by the median coverage of all labels. This relative coverage accounts for the systematic biases in raw coverage data due to factors such as local label density patterns and repeat regions that may impact coverage but are not directly related to true CNV events. For each query sample of interest, the raw coverage is normalized at each label by dividing it by the relative coverage estimated from the control samples. The normalized coverage data are used as input for segmentation in **Step 2**. An example illustrating the normalization step is shown in **Figure 6**.

### Segmentation

Conceptually, the CN state of a particular genomic region is proportional to the mean coverage of labels in that region. To detect complex CNV events with possibly multiple CN changes in the genome, a segmentation algorithm called Wild Binary Segmentation[3] (WBS) is used to partition the genome into distinct segments. Each segment is inferred to have a statistically different mean coverage, and hence, a CN state that is different from its immediate neighboring segments in the genome. WBS is an extension to the Binary Segmentation and Circular Binary Segmentation algorithms that have been used extensively to detect CNV events in array-CGH or next generation sequencing (NGS) data. WBS has been shown to have better sensitivity in detecting smaller changes when compared to conventional methods. After segmentation, a set of genomic intervals is returned, where each interval is inferred to have a different CN state compared to its immediate neighboring regions. An example of the segmentation procedure is illustrated in **Figure 7**.

[3] Piotr Fryzlewicz. Wild binary segmentation for multiple change-point detection. Ann. Statist. Volume 42, Number 6 (2014), 2243-2281.

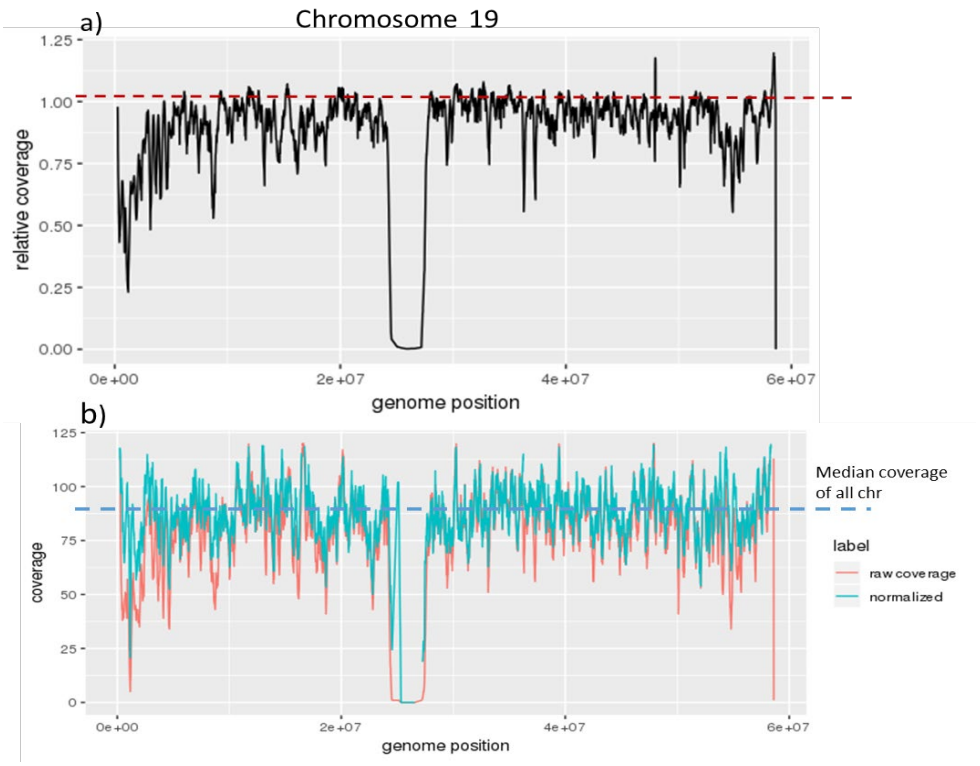



**Figure 6**. Normalization of raw coverage data. a) The relative coverage of chr19 from the pooled control data is shown. The relative coverage in chr19 was consistently below one. This means that on average, fewer molecules mapped to chr19 compared to other chromosomes. There is also local fluctuation (e.g., beginning of the chromosome) in the coverage due to factors such as label density. The normalization procedure is designed to correct for such systematic biases in raw coverage data. b) Comparison of coverage data from an example query sample before (red) and after (cyan) normalization. After normalization, the overall coverage became closer to the median coverage of the whole genome and local variation in the data was reduced.

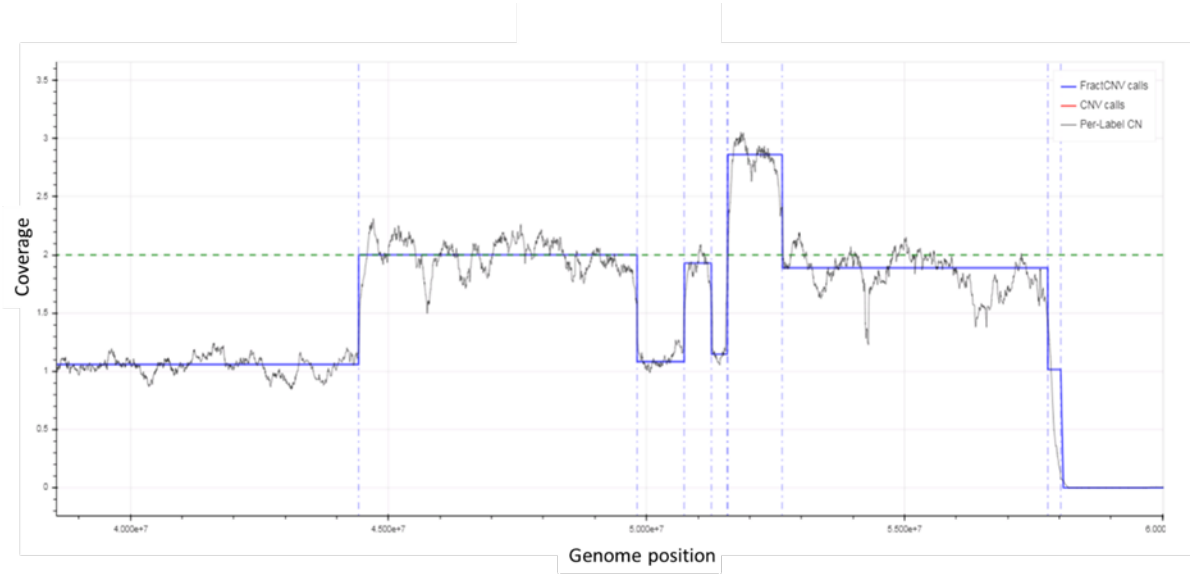



**Figure 7**. Segmentation of coverage profile. Genomic regions with the same CN states are assumed to have the same mean coverage with random local fluctuation around the mean. “Jumps” are expected when neighboring regions have different CN states. The segmentation procedure looks for such signature jumps and partitions the genome into non-overlapping intervals that are deemed to have different CN states. The figure shows example results from segmentation. The segment boundaries are marked by the vertical dashed lines. The solid blue line indicates the average coverage for a segment, and the black line shows the normalized coverage at each label along the genome.

### Baseline estimation

One is interested in detecting regions of the genome where the CN states deviate from the normal or baseline CN state. In human, the normal CN state is diploid (copy number of 2). However, unless external data such as those from karyotyping are available, the part of the genome that is diploid is not known *a priori* and could only be inferred from the data. Also, there is not enough information to infer *absolute* copy number states of a genome solely from coverage data. Only *relative* copy number states can be inferred from the data. If the true ploidy is known based on external sources and is not diploid, users could scale the results accordingly. If the true ploidy is 3, one would multiply the inferred copy number states by 3/2 (or 1.5) for all chromosomes.

The pipeline makes a simplifying assumption that for a genome of interest, the fraction of the genome with a normal CN state is larger than the fraction of the genome with abnormal CN states. It then estimates the mean coverage of the normal diploid state by computing the mode of the coverage of all labels. This is defined to be the baseline coverage for a genome, and it is used as the basis for calling genomic regions with abnormal CN states. However, baseline estimation can be challenging if there are multiple overlapping underlying distributions of similar proportions, resulting in close peaks in the overall coverage distribution.

Two examples of baseline estimation are illustrated in **Figure 8**. The sex chromosomes are treated differently than autosomes. For human datasets, the pipeline first checks whether chrY is presented by determining if there is non-trivial molecule coverage along chrY. If chrY is deemed present, the baseline coverage for the sex chromosomes is adjusted to be half of that of the autosomes. Otherwise, the baseline coverage for chrX is assigned to be the same as autosomes. The expected sex chromosome coverage values can be specified in the input for other species as discussed in the Usage section.

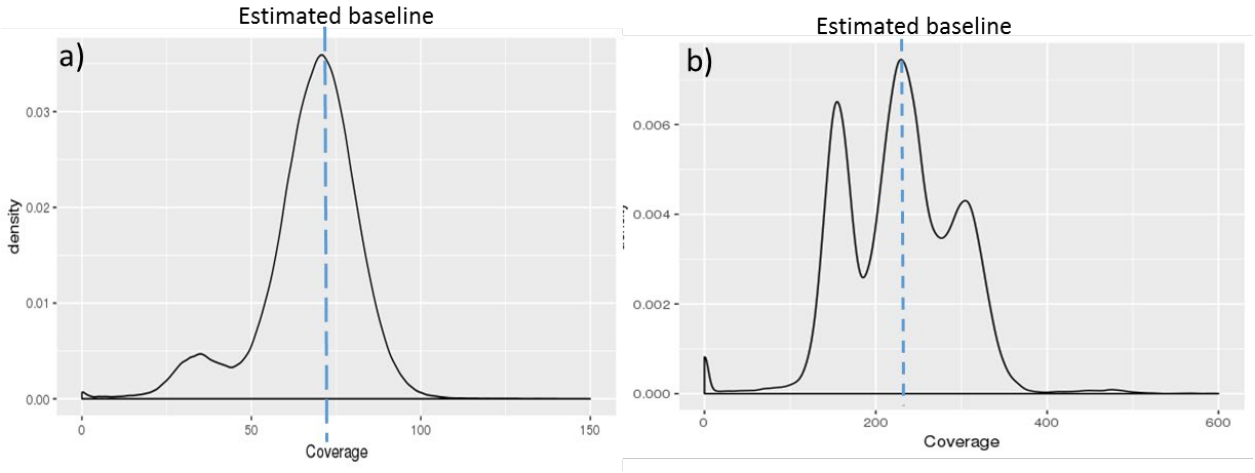



**Figure 8**. Baseline estimation. To estimate the baseline coverage, the distribution of the coverage data is analyzed, and the mode of the distribution is taken as the baseline coverage. a) The coverage distribution of a normal sample is shown. This sample is expected to have very few or no true CNV events; thus, the coverage data shows one single dominant distribution. The smaller distribution with the mode approximately half of that of the dominant distribution is the coverage distribution for sex chromosomes, since this is a male sample. b) The coverage distribution for a cancer sample with complex CNV events is shown. The various peaks in the distribution correspond to known chromosome aneuploidy events in the sample. The mode of the overall distribution (i.e., the tallest peak) is defined to be the baseline coverage of the genome.

### CNV detection and confidence score computation

Once the baseline is established, the pipeline determines which genomic intervals from the segmentation step have abnormal CN states by checking if a particular genomic region has a mean coverage that is (statistically) significantly higher or lower than the baseline. It expects that genomic regions with same CN states have the same mean coverage equal to the baseline coverage plus some small variation due to random and systematic fluctuation in the data. Conceptually, the expected coverage (and hence, a CN state) under the null hypothesis should be consistent with the baseline. Intuitively, the smaller this probability is (i.e., the chances of observing the empirical mean coverage of a region's being much lower or higher than the baseline coverage or CN state), there is higher confidence that this region has a true CNV event or multiple events whose aggregated coverage deviates from the baseline. The probability is adjusted by accounting for multiple hypothesis testing. The confidence scores are computed as one minus the adjusted probability. Like **Step 3**, for human, the pipeline first determines if chrY is present. If it is, the confidence scores for sex chromosomes are computed by assuming the baseline coverage is half of those of autosomes. For non-human datasets, the pipeline handles the sex chromosomes as specified in the chromosome configuration input. If the chromosome configuration is absent, the pipeline would assume that all chromosomes have the same baseline coverage.

Whole chromosome aneuploidy detection is described in detail in the Whole-chromosome Aneuploidy Detection section below.

### Region-specific Scoring

Because many regions in the genome behave differently than a “typical” region, using a single statistical model to model the coverage profile throughout the whole genome may have incorrect assumptions. These are often complex genomic regions such as centromeres or telomeres, or regions that contain segmental duplications or repetitive sequences. The coverage in these regions tends to be systematically higher or lower than the baseline and displays unusually high variability. Therefore, they are easily mistaken to contain true CNV events, based on the whole-genome model. With the increasing availability of Bionano data from control samples (i.e., samples from healthy individuals where very few or no CNV events are known), Solve uses a separate statistical model for each genomic region. Data from ninety-nine control samples were used for building the model. Rather than using a single whole-genome statistical model to evaluate whether the coverage profile of a query genomic region is significantly different, now the coverage profile of the query region is evaluated against the background distribution of coverage profiles from a set of control samples in a region-specific manner (See **Figure 9**). A kernel density method is used to model the empirical distribution and compute region-specific p-values and confidence scores.

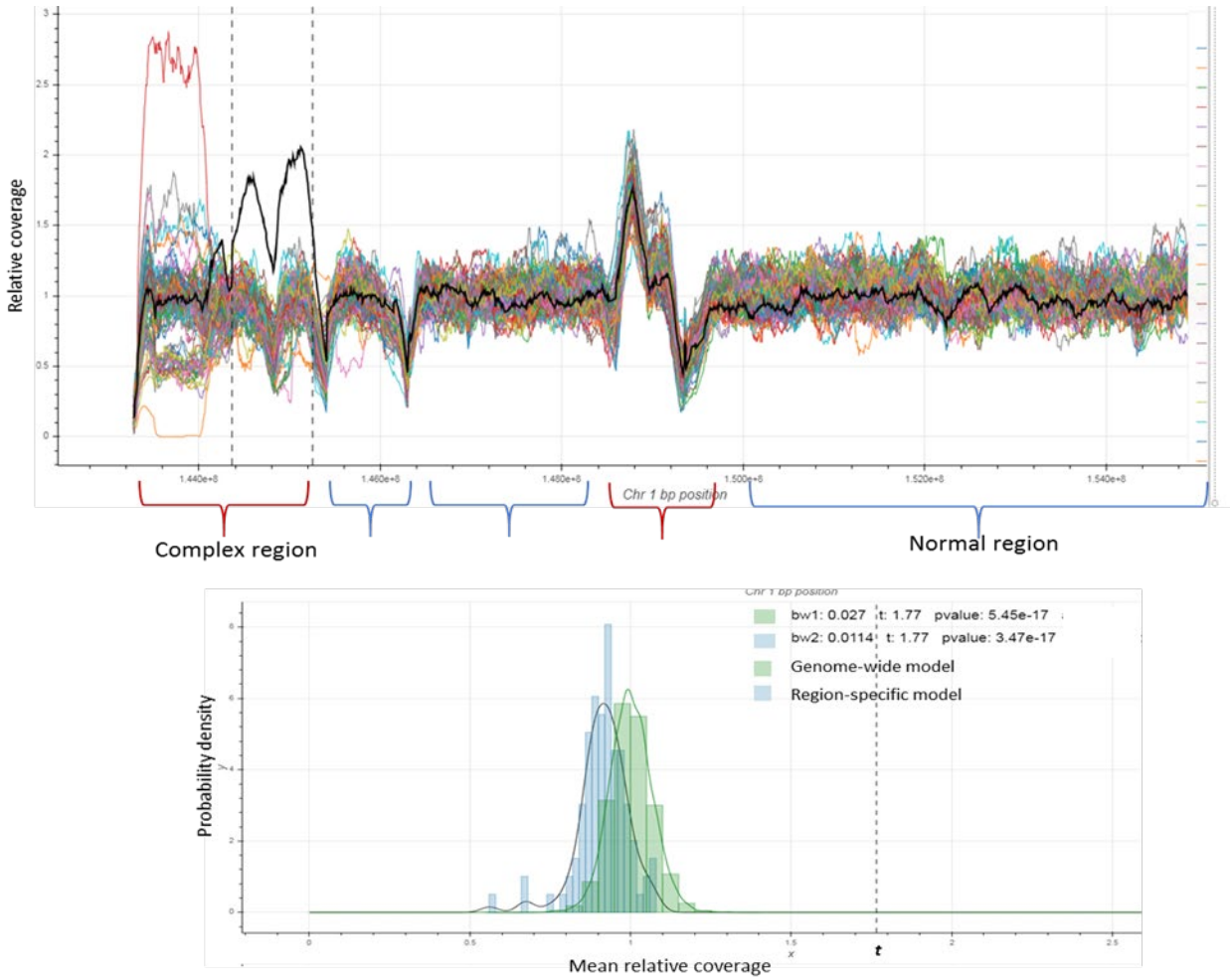



**Figure 9**. Top panel: Coverage profiles of an example region from approximately one hundred control samples are shown as colored lines. The coverage profile of a query sample is shown as bold black line. Genomic regions can be broadly divided into normal and complex categories. Normal regions are regions where the coverage profile follows an expected pattern – the coverage fluctuates randomly and uniformly around the baseline coverage (regions indicated by blue curly brackets). Complex regions are regions where the coverage profile has atypical patterns in a region/location-specific manner (indicated by red curly brackets). The CNV pipeline utilizes data from a large set of control samples to build a region-specific statistical model for each of the complex regions and determine whether the coverage profile of a query sample is statistically different from control samples. To illustrate this, a region of interest is highlighted between two vertical dashed line in the top panel. Bottom panel: to determine if the region has a CNV event, the mean coverage of all control samples is used to build an empirical distribution of coverage of the specific region (blue histogram). This is contrasted with the old whole-genome model (green histogram). The coverage profile of the query sample (vertical dash line in bottom figure and solid black line in top panel) is evaluated against the background distribution of the control samples (blue histogram), and the statistical significance is evaluated using a kernel density estimator.

## MULTI-ALLELIC REGIONS

Some regions of the genome are multi-allelic with respect to their copy number states in the control population. The empirical distribution of coverage profile in these regions is multi-modal. In the current pipeline, we treated this multi-modal distribution as a single background distribution. As a result, some common CNV events in the control population will not be considered statistically significant events. However, if a rare or pathogenic variant

that occupies a CN state that is substantially different than those in the control population, it would still be considered significant (see **Figure 10**, bottom panel).

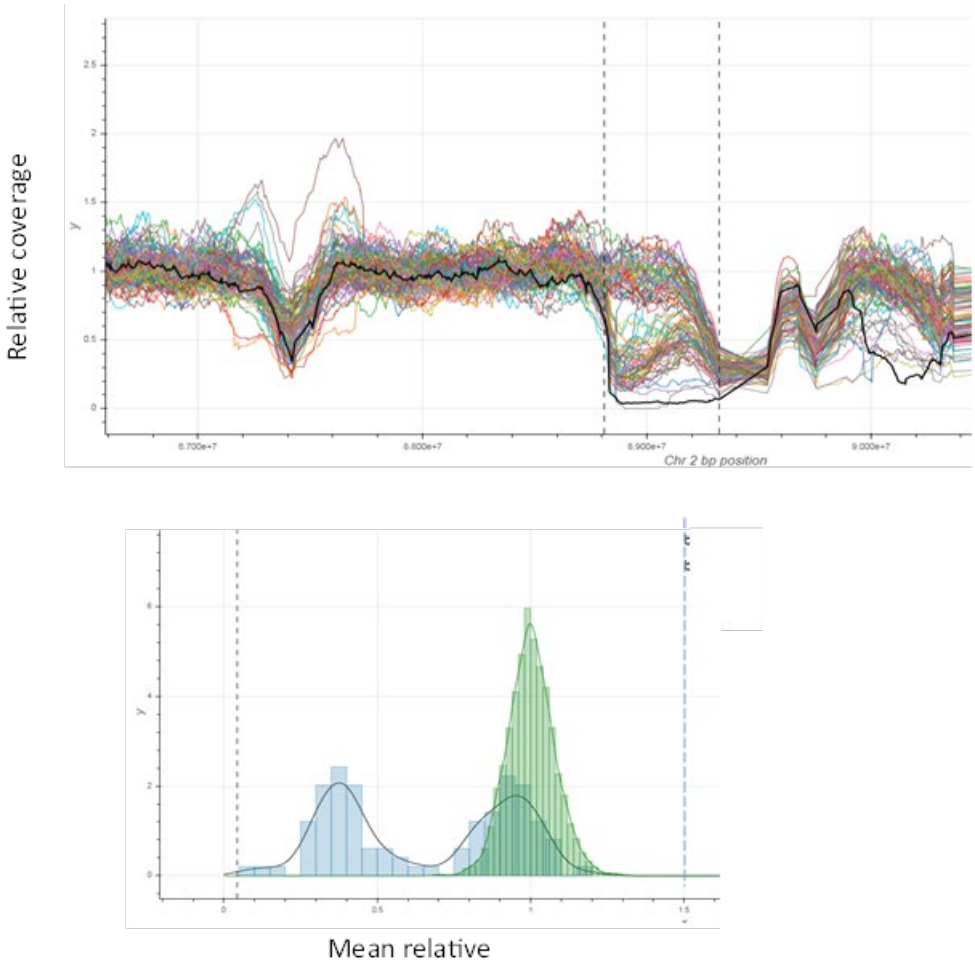



**Figure 10**. Example of a multi-allelic region. Top panel: Color lines are coverage profile from control samples. The region between two vertical dashed line is a region of interest with a multi-allelic CNV. Bottom panel: Blue histogram shows the distribution of mean relative coverage of all labels in the region of interest and green histogram shows the distribution of mean relative coverage across the genome with same size as the region of interest. The coverage of region of interest clearly has two distinct states around relative coverage of 1 and 0.5. As a result, samples with coverage near 1 or 0.5 in this region would not have significant CNV calls, but if a sample with relative coverage around 0, or 1.5 in this region (vertical dash line) would have statistically significant CNV calls.

### CNV events in the Y chromosome

When analyzing the data from control samples, we observed that the variability of coverage profile in the Y chromosome is substantially higher compared to other chromosomes in the genome (see **Figure 11**). This means that the sensitivity of calling fractional CNV events will be relatively lower. In fact, a substantial portion of Y chromosome is in masked regions (see “Masking of high variance regions” for definition of mask regions). With the regions-specific scoring model, CNV events could still be called in these variable regions, but users will have to ignore the mask and use the confidence score as a guide to look for true CNV events.

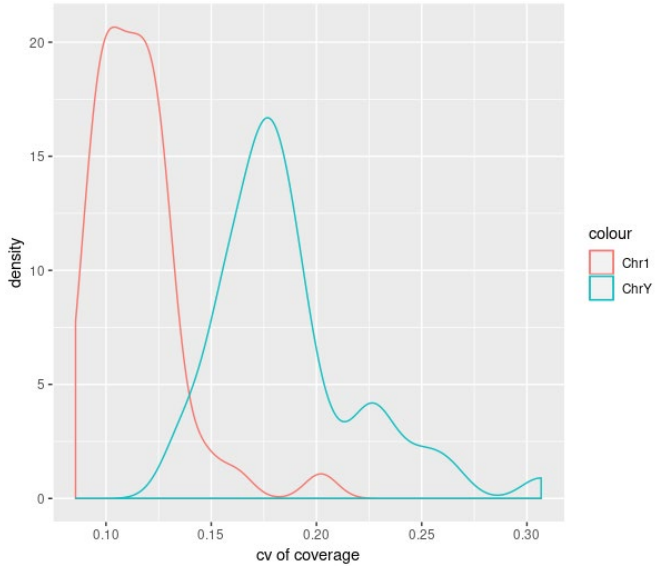



**Figure 11**. Coefficient of variation of coverage values across control samples for Chromosome 1 and Y.

### CNV events near SV breakpoints

Traditionally, CNV pipelines detect gains and losses events by analyzing the coverage profile and detect any statistically significant increase and decrease in coverage depth. However, structural variation near the breakpoint of other types of SV such as inversion or translocation can also cause drops in coverage since molecules near the SV breakpoints cannot be fully aligned to the reference. In other technologies such as microarray or next generation sequencing, because the probes or the reads are noticeably short compared to the size of typical CNV events, the impact on coverage around SV breakpoints tends to be minimal. However, due to the use of ultra-long molecules, the coverage drop can be significant and can lead to deletion calls. An example illustrating the coverage drop near the breakpoints of a large inversion is shown below **(Figure 12**). While there is likely not a genuine deletion, this drop in coverage is indicative of some rearrangement happening at the location. This needs to be considered when interpreting deletion events from the CNV pipeline.

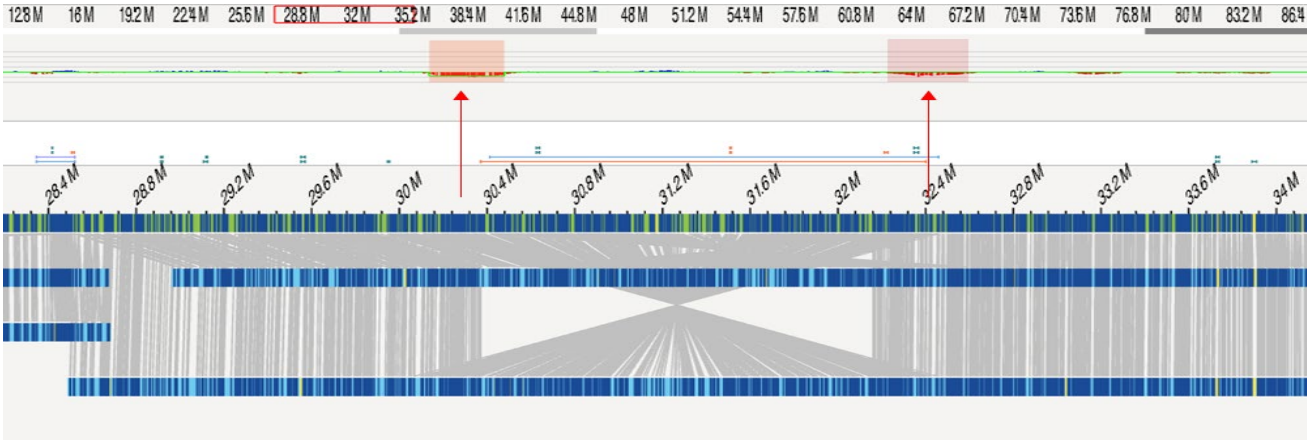



**Figure 12**. Example of deletion calls near SV breakpoints. The CNV pipeline detects statistically significant increase or decrease in the coverage profile data. In addition to detecting the classical duplication and deletion events, it sometimes detects drop in coverage near breakpoint around other SV type such as duplication. This is because molecules can only partially align to the reference near SV breakpoints and thus contribute to the lower coverage depth in such regions.

### Systematic bias in coverage profile

In a small fraction of samples, it has been observed that certain systematic biases in the coverage profile exist. These biases cause systematic increase or decrease in coverage in a location-specific manner and do not correlate with true CNV events. The magnitude of these biases may differ across samples. **Figure 13** shows examples of coverage profile with severe systematic bias (a), mild bias (b), and no bias (c). As illustrated in (c), the coverage profile looks relatively flat and uniform when there is no systematic bias. However, with bias (in a and b), the coverage profile looks like there are many CNV events. To assess the amount of bias in the input sample, the pipeline reports several statistics. The key observation is that when a sample contains significant systematic bias, the variation measure of the coverage profile would be elevated across the whole genome. This is often measured by the global coefficient of variation (cv) in the coverage profile. This value should usually be less than 15%.

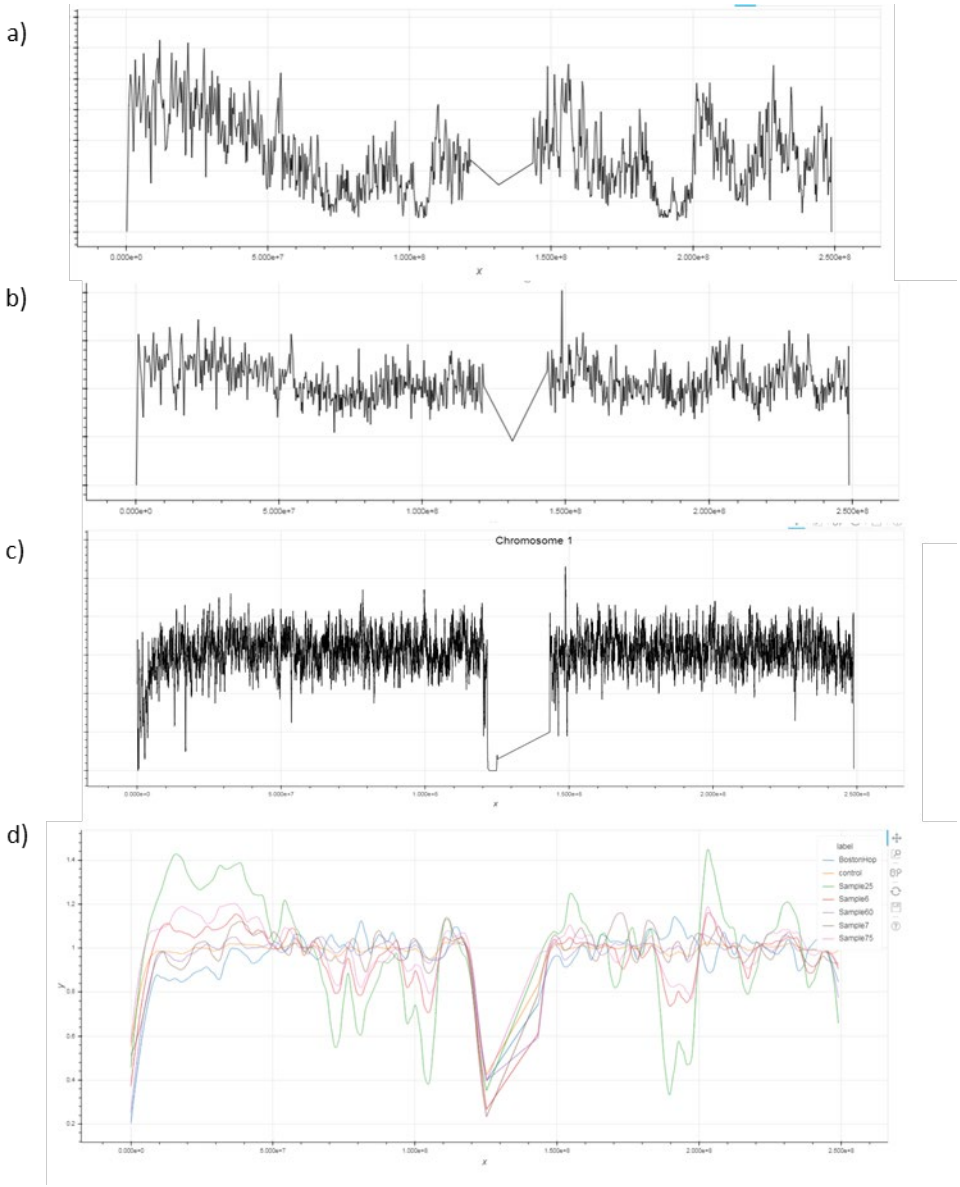



**Figure 13**. Example coverage profiles of samples with severe systematic bias (a), mild systematic bias (b), and no significant bias (c). The systematic bias shares a similar pattern across different samples. The colored lines in d) are the smoothed trend lines of the coverage profiles from multiple samples that are detected to contain different levels of bias.

However, true CNV events can also lead to an increase in variation in the coverage profile. To address this, the pipeline also computes the variation of the coverage in local windows. The idea is that nearby genomic locations are likely to be in the same CN states, so their variation should reflect the background variation of the data, despite the presence of true CNV events. We measure local variation in two window size: 2-Mbp windows and 6-Mbp windows. It provides variation estimates in distinct size scale. As mentioned in *4. CNV detection and confidence score computation* above, the variation of coverage profile depends on the total coverage depth and size of the region being considered. The higher the coverage or region size, we expect the variation to be smaller. Thus, based on empirical data, we built a model of what the variability of coverage data should be if a sample has no systematic bias. Then, we computed and reported the percent difference between the expected cv and the observed cv in the sample. The higher this difference is, the more severe the bias is in the sample. As a general

guideline, if the observed percent difference is greater than 20%, the sample should be considered to contain systematic bias. The noise estimates are in cnv_chr_stats.txt.

Two metrics are included in the 'cnv_chr_stats.txt' file to quantify systemic coverage biases (**Figure 14**). These are 1) wave template correlation and 2) correlation with label density. The first metric measures the correlation between the coverage of the query sample and the coverage profiles from known samples with large systematic bias. The idea is to recognize known patterns of systematic bias in new samples. The second metric measures the correlation between coverage of a particular genomic region and the label density in that region. It was found that this can be another source of systematic bias in coverage data. In both cases the higher the correlation the more serious the systemic bias. By analyzing the noise metrics in a set of control samples with or without known systematic bias, it was found for a sample to have reasonable low level of bias, the wave template correlation should be smaller than 0.4 and the absolute value of the label-density correlation should be less than 0.25.

There may be increased false positive CNV calls in datasets with high systematic biases. The coverage profiles tend to fluctuate significantly even in the absence of true CNV events. Users noticing more than expected CNV calls are encouraged to contact Bionano Support (support@bionano.com) for any assistance with such datasets.

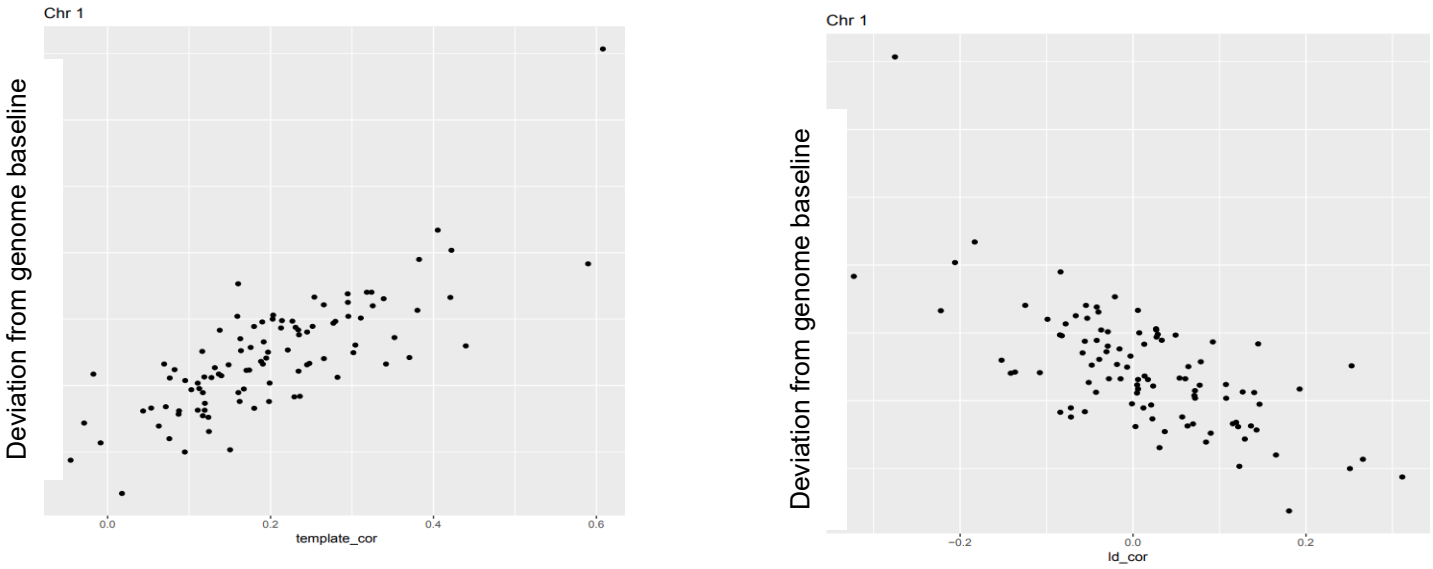



**Figure 14**. Quantifying systematic bias in coverage profile in control samples. To measure systematic bias in a set of control samples we computed the deviation of the coverage of a particular chromosome from that of the whole genome. Control samples are NOT expected to have chromosome aneuploidy events thus we expected the chromosome coverage to be equal to or like the genome-wide baseline. Any deviation from it can be a potential systematic bias in the coverage profile. Two possible sources contribute to this systematic bias. In the panel on the left we computed wave template correlation which is the correlation between the coverage of a sample versus samples with known systematic bias. In the right panel we computed the correlation between label density and coverage profiles. As shown, the higher this correlation the higher the deviation of coverage of a chromosome from the genome-wide baseline, thus indicating that these two correlations can be used as a metric to quantify the systematic bias in new samples.

### Whole-chromosome Aneuploidy Detection

After calling potential CNV events, the pipeline performs an additional summary operation by grouping all the deletion or duplication events for each chromosome. If more than 80% of a chromosome by length has either lower or higher CN states than baseline, a whole-chromosome aneuploidy event would be reported in the output file with prefix “cnv_chrAneuploidy”.

Bionano Solve includes an algorithm to detect whole chromosome aneuploidy events with extremely low allele fraction (i.e., down to 5%). This module considers the coverage data of each chromosome as a whole and compares that against the rest of the genome. It then detects whether the chromosome coverage data deviates from the whole genome baseline coverage. By considering the coverage data from the whole chromosome, the algorithm takes advantage of the substantial number of data points available and can detect exceedingly small changes in coverage. However, this also means the algorithm is sensitive to local copy number changes in the chromosome. It can potentially mistake a significant local CNV change as a low allelic fraction whole genome event. Therefore, genomic regions with significant local CNVs that were detected in the previous steps were removed from consideration before analyzing each chromosome for aneuploidy event. This means that such analysis is not suitable to analyze samples with extensive local rearrangement or CNV events such as cancer samples. In such cases, we fall back to the default aneuploidy detection algorithm as described above.

To improve the sensitivity of aneuploidy detection, we have identified two types of systematic bias that can cause fluctuations in the coverage data that does not correspond to biological events (see section Systematic bias in coverage profile above). These biases are relatively small and were ignored in previous version of the CNV pipeline. However, they become significant when one is trying to detect exceedingly small changes in coverage data as in low allelic fraction cases. A bias correction/normalization procedure was implemented to remove such systematic bias from coverage data. As shown in **Figure 15** below, by removing such systematic bias we can reduce the variation of chromosome coverage significantly and which allow us to detect small fractional changes in coverage data.

After removing systematic bias from the coverage data, the empirical distribution of the relative coverage for each chromosome (relative to genome-wide baseline) was constructed based on a set of control samples. When a new sample was being analyzed, the statistical significance of the deviation of the chromosome coverage from its genome-wide baseline was evaluated based on this empirical model and any significant deviation are considered as chromosome aneuploidy events. The confidence score reported is 1-pvalue based on the empirical statistical model.

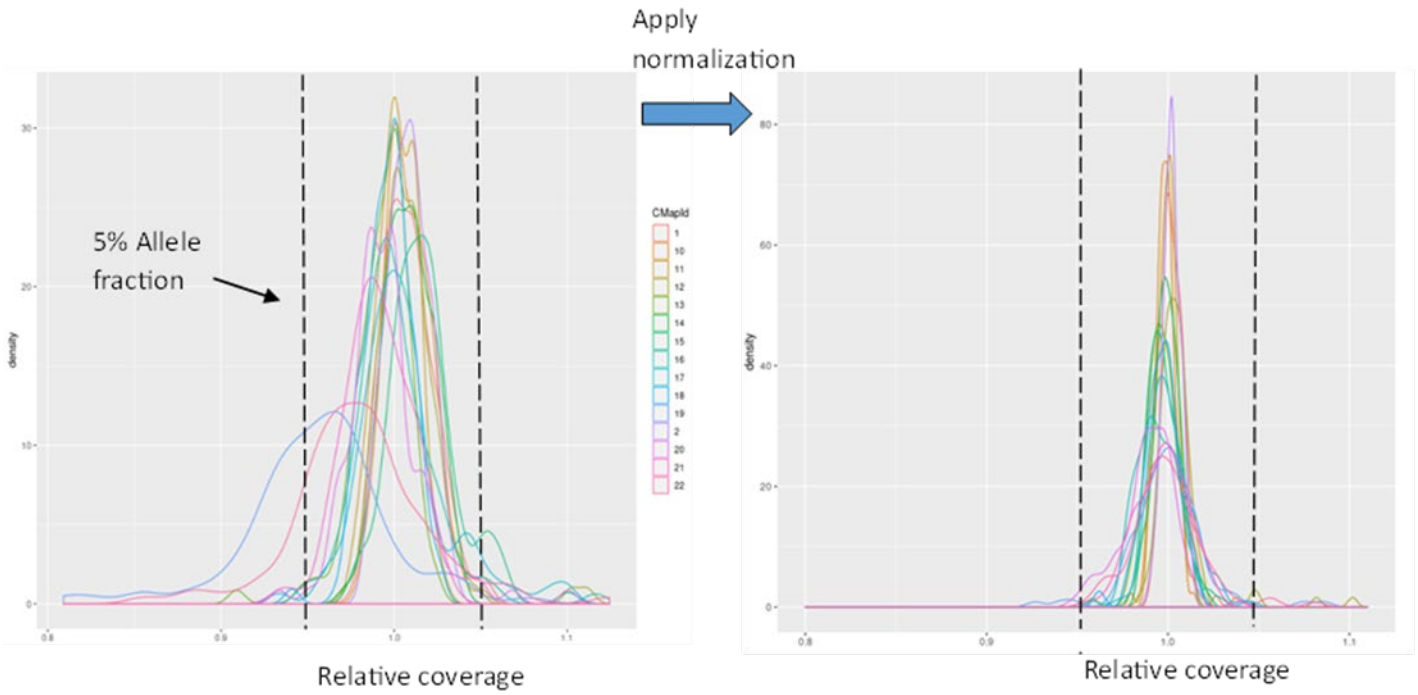



**Figure 15**. Systematic bias normalization reduces variation in chromosome coverage. The distribution of the relative coverage of each chromosome in a set of controls samples is shown. Relative coverage is the coverage of a particular chromosome relative to the whole genome coverage baseline. Different chromosomes have different variations in chromosome relative coverage. After the application of bias correction/normalization procedure the variation across all chromosomes is significantly reduced.

### MASKING OF HIGH VARIANCE REGIONS

Based on analysis of DLE-1 control datasets, we defined regions of the genome that had unusually high variance in their relative coverage compared to typical loci across control datasets. Relative coverage for each sample is defined as the raw coverage divided by the median coverage across the genome. We compute an average and standard deviation for each locus. A position is deemed variable if its variance is above three standard deviations away from the average.

These high variance regions are often concentrated around centromere and telomere regions, where molecule alignment can be unreliable. CNV calls overlapping with these regions (by at least 45% in length) are annotated as "_masked"; they are more likely to be false positive calls. The mask does not influence the initial CNV calling step. Calls are made with no knowledge of the mask. The mask is applied only after the calls have been made.

The masks are available for hg19, hg38, T2T-CHM13v2.0, mm10, and mm39 and are automatically applied during the pipeline run. They are also available as BED files for visualization in Bionano Access. It is possible that common, highly polymorphic regions are included in the masks. The masked calls are shown in the CN table in Access and can be manually inspected.

**NOTES:**

- Performance for sex chromosomes is not as well-understood; less control data was available for sex chromosomes. Also, while supported, analysis of non-human samples has not been validated. Manual curation is encouraged.
- For female samples, no CN calls on chromosome Y are output.
- If there are no CNV calls, the CN table (cnv_calls_exp.txt) and the aneuploidy table (cnv_chrAneuploidy.txt) may be empty.
- The pipeline assumes that the query dataset is of sufficient quality relative to the control datasets that were used for normalization.

**PRE-PROCESS CUSTOM CONTROL DATA**

Scripts for pre-processing custom control data are for advanced users to prepare their own control data for the CN pipeline. See Appendix I for detailed instructions.

### SV Masking

The genome analysis pipelines include BED files for annotating insertion and deletion calls overlapping N-base gaps in the reference and putative FP translocation breakpoint calls so that they could be filtered in Access. The former would be annotated with a suffix "_nbase" in the SV type. Insertion and deletion calls in N-base regions may simply be due to mis-sizing of the N-base gaps in the reference and not genuine SVs. The latter would be annotated with a suffix "_common" or **"_**segdupe**"** in the SV type, depending on whether they overlap with common FP calls in control samples or annotated segmental duplication regions >50kb in size, respectively. For example, it includes selected sub-centromeric and sub-telomeric regions that are prone to generating putative FP translocation breakpoint calls. We provide BED files for each analysis pipeline separately. Details on custom BED generation are described in the "Appendix B" and "FAQ" sections.

The putative FP translocation breakpoint calls are derived based on translocation breakpoint calls in control samples, and Bionano has continued to increase the number of control samples.

Below are the unique base pairs contained in each of the categories in the ***SV mask*** BED files for Solve 3.8 (the same segmental duplication and gap regions are shared between BED files). T2T-CHM13v2.0 is a gapless reference and includes no gap regions. Total bp reported are non-redundant, unique bp – because there may be overlap between gap, segmental duplication and "common" regions, the total is different than the sum of the three categories.

**Table 5**. Segmental duplication and gap bases in Solve 3.8 SV Masks

| Mask type | hg38 | hg19 | T2T-CHM13v2.0 |
|---|---|---|---|
| segmental duplication | 88,294,774 | 85,579,029 | 185,123,523 |
| **gap** | **85,960,550** | **234,344,783** | **n/a** |

**Table 6**. Unique bases in Solve 3.8 SV Masks

| Pipeline | hg38 | | hg19 | | T2T-CHM13v2.0 | |
|---|---|---|---|---|---|---|
| | common | total | common | total | common | total |
| **De novo assembly** | 23,654,061 | 179,068,217 | 23,886,354 | 325,413,043 | 19,861,718 | 186,706,326 |
| **RVA** | 15,651,384 | 177,825,620 | 17,726,294 | 323,839,338 | 16,963,401 | 186,205,892 |
| **Guided assembly** | 26,158,881 | 179,075,814 | 27,494,224 | 325,124,437 | 24,582,867 | 186,717,312 |
| **Guided assembly - LAF** | 23,813,948 | 180,537,572 | 25,405,639 | 326,903,380 | 20,580,174 | 186,601,258 |

Solve uses the same CNV mask for all pipelines. Below is the amount of sequence in the file (bp) in Solve 3.8:

**Table 7**. Unique bases in Solve 3.8 CNV Mask

| hg38 | hg19 | T2T-CHM13v2.0 |
|---|---|---|
| **314,237,741** | 310,187,299 | 357,041,676 |

### Rare Variant Analysis Pipeline (RVA)

The ability to detect constitutional and somatic SVs is important for studies of cancer genomes. Bionano's *de novo* assembly-based SV detection approach – first constructing a *de novo* assembly and then comparing the resulting assembly with a reference – is extremely sensitive in detecting homozygous and heterozygous insertions, deletions, duplications, inversions, and translocations in a diploid genome as small as 500 bp. However, this approach may miss SVs at low variant allele frequencies. Introduced in Bionano Solve 3.4, the RVA pipeline is designed specifically to identify variants at low variant allele frequencies in the single-molecule data, while still finding homozygous and heterozygous variants. Low variant allele frequency variants are prevalent in heterogeneous samples such as cancers or samples with allelic mosaicism.

RVA consists of two major components: 1) the "split-read" analysis, and 2) the copy number analysis. Both analyses use information from a sample's molecule alignment against a reference assembly, bypassing the *de novo* assembly, and calling SVs based on these alignments. Neither analysis makes assumptions about the ploidy of the sample when detecting low allele-frequency variation. The "split-read" analysis calls SVs by examining the molecule alignments and searches for clusters of molecules with internal alignment gaps, and multiple alignments. The copy number analysis detects copy number variation (CNV) by identifying regions of the genome with significant coverage elevation or depression.

There are three major steps in detecting SVs by the "split-read" analysis: initial molecule alignment and clustering of SVs, consensus generation by molecule extension refinement, and final SV calling.

### Initial molecule alignment and clustering by SV

Molecules in an input BNX file are aligned directly to the reference, for example, human reference build GRCh38. Significant internal alignment gaps (termed outliers) and end alignment gaps (end outliers) may indicate the presence of SVs in the sample. The pipeline requires only a default minimum of three molecules calling the same SV, for SVs over 30kb in size and up to 10 molecules calling the same SV for smaller deletions down to 5kb. Molecules are determined to confirm the same insertion, deletion, duplication, or inversion if the inferred variant positions overlap and their inferred variant sizes are similar (default of 20% size similarity). To confirm the same translocation, the inferred translocations must be in the same orientation and their breakpoints in proximity (within a default distance of 35 kbp).

#### CONSENSUS GENERATION BY MOLECULE EXTENSION REFINEMENT

A consensus is built using clusters of molecules that identified the same SV. The purpose of the consensus step is to verify that those SV-calling molecules agree and can form a consensus that represents the variant allele. In this step, the loci on the reference assembly flanking the inferred SV are extracted, and the molecules that called the SVs are aligned to each of the two SV-flanking reference fragments. The pipeline attempts to reconstruct the SV allele by using the molecules to extend into the SV region. If the molecules come from the same variant, they would have similar label patterns and form a consensus map that represents the variant allele. **NOTE:** For each SV, the same extension procedure is performed twice: one extension from the fragment left of the SV and one extension from the right.

#### FINAL SV CALLING

Finally, the new local consensus maps are realigned to the reference to check if the same initial SV calls are made. The pipeline will only report SVs that are confirmed in the final SV calling step. **NOTE:** Potentially two consensus maps could form for each SV, but only one is kept in the end.

The SVs identified by RVA are listed in Bionano SMAP format. The SMAP file can then be used as input into downstream analysis workflows such as Variant Annotation Pipeline.

### Variant Allele Fraction (VAF) Calculation

VAF values are provided by default in the output SMAP file of RVA with the same interpretation as for the *de novo* assembly pipeline. The calculation uses consensus map coverage and the same Bayesian inference as for *de novo* but requires some extra steps to obtain the coverage for the reference allele as it is not one of the RVA outputs. The extra steps are as follows: 1) Extract the fragments of the reference map spanning the SVs detected by the pipeline. 2) Competitively realign the input molecules to both the consensus maps and the extracted reference fragments; this way the input molecules aligned to the reference fragments provide the coverage for the reference allele 3) Run SVs calling on the consensus maps newly created during the competitive realignment. These maps and the newly called SVs are expected to minimally differ from those obtained after RVA but they will contain more accurate coverage values, and 4) Once the VAF is calculate for these new SVs, the value is transferred to their equivalent in the original RVA output. In the case that an original variant cannot receive a transferred VAF, the value will appear as -1. Refer to “Appendix H” for further information.

## Guided Assembly

Solve 3.8.1 introduces a new analysis pipeline called Guided Assembly (or reference guided assembly). Guided Assembly is a variant of the de novo assembly pipeline that uses a reference genome instead of draft consensus maps as the seed for extension and refinement. The Guided Assembly pipeline provides two different operation types, one optimized for constitutional applications (Guided Assembly or Guided Assembly – Constitutional) and one optimized for cancer or other applications aimed at detecting low-allele fraction variants (Guided Assembly – LAF). Together, these two analysis modes are offered as alternatives to the *de novo* assembly pipeline and the Rare Variant Analysis pipeline, respectively.

### THEORY

The Guided Assembly process closely follows that of *de novo* assembly (described in detail in Appendix D). The initial steps consisting of molecule filtering and estimation of error parameters (autoNoise) are identical. However, Guided Assembly skips pairwise alignment of molecules and the creation of an initial consensus draft assembly and instead uses the reference genome (represented in an input CMAP) as the starting point for refinement and haplotype splitting. Consensus maps are refined through extend and merge stages as in *de novo* and then structural variants are detected by aligning refined maps to the reference genome. Copy number variants are detected using the same process as is used in both *de novo* and RVA.

The constitutional and low-allele fraction operations of Guided Assembly are performed by the same software pipeline with different parameters (optargs). The constitutional assembly operation is parameterized identically to the *de novo* assembly pipeline, where the expectation is of a diploid genome with each allele being represented evenly. Constitutional analysis can detect variants down to 20% mosaicism. Guided Assembly – LAF is tuned specifically to detect variants at allele fractions down to 5% with high sensitivity and precision.

### COMPARISON TO *DE NOVO* ASSEMBLY AND RARE VARIANT ANALYSIS

Guided Assembly – LAF improves the variant detection performance of RVA at low-allele fractions. The pipeline enables detection of smaller insertions, deletions, and duplications at 5% allele fraction while increasing the size of insertions that can be detected. An additional advantage of Guided Assembly is that it performs a full assembly of the entire genome rather than the targeted assembly around putative structural variants that is performed by RVA. This means that Guided Assembly can be more accurate in its variant allele frequency calculations for low frequency variants and can give additional confidence in the *absence* of variant calls by providing a full assembly of the reference allele. Guided assembly for constitutional applications shows equivalent performance to *de novo* assembly, with the primary benefit being the consolidation of both operations into a unified pipeline.

# Structural Variant Calling Performance: Guided Assembly

Variant calling performance for all pipelines has been evaluated using simulated data at varying coverage levels and variant allele frequencies. Detailed methods are included in “Appendix A.” Briefly, DLE-1 molecules were simulated from unedited and edited versions of the hg19 reference genome (with 1600 insertions and 1600 deletions from 500 bp to 1 Mbp), mixed at different proportions to represent relevant allele fractions and coverage depths, and used for SV calling.

## Structural Variant Calling Performance Using Simulated Data

### SUMMARY – GUIDED ASSEMBLY (LAF)

We observed at least 90% sensitivity at 300X effective coverage at 5% variant allele frequency:

- Insertions between 3 – 140 kbp
- Deletions > 7 kbp
- Translocations (or transpositions where the sizes of the translocated fragments are > 70 kbp)
- Inversions > 50 kbp
- Duplications > 50 kbp

### PERFORMANCE FOR SIMULATED INSERTIONS AND DELETIONS

Performance data are shown for insertions deletions in **Figure 16** below.

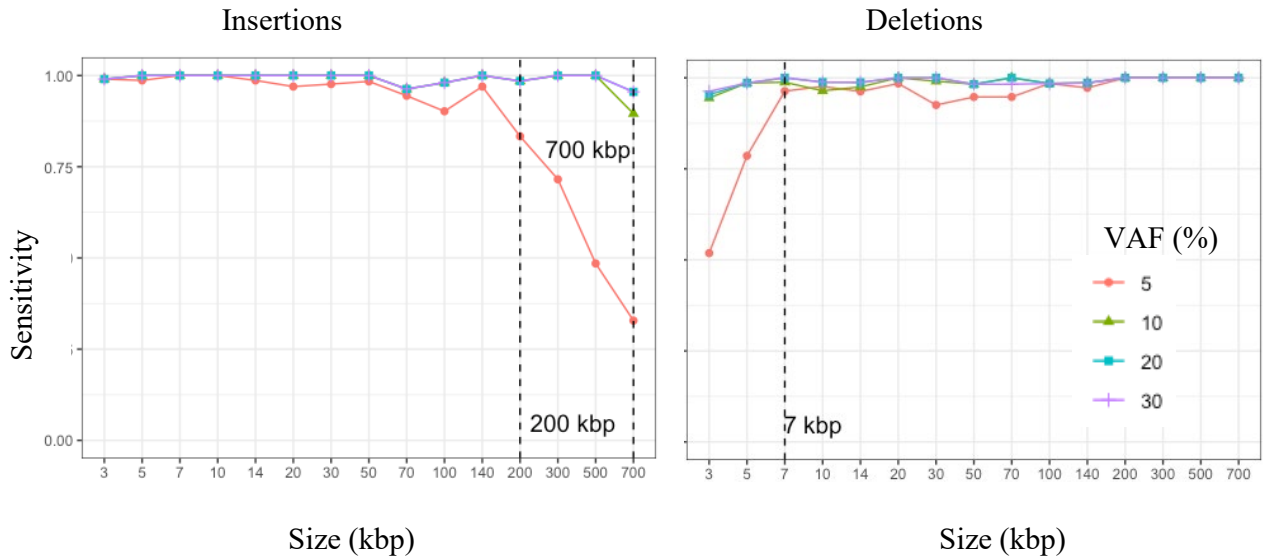



**Figure 16**. Insertion and Deletion Calling Performance With Simulated 300x Dle-1 Data at Different Variant Allele Frequencies. Performance For Simulated Duplications, Translocation Breakpoints, and Inversion Breakpoints

Performance data are shown in **Figure 17**. Duplications, translocations, and inversions need to be at least a certain size for the pipeline to recognize that a piece of sequence has been duplicated, transposed, or inverted. Small duplications and translocations will be classified as insertions, if the involved sequence is not large enough to be uniquely aligned. Small inversions with gain or loss of sequence relative to the reference may be classified as insertions or deletions.

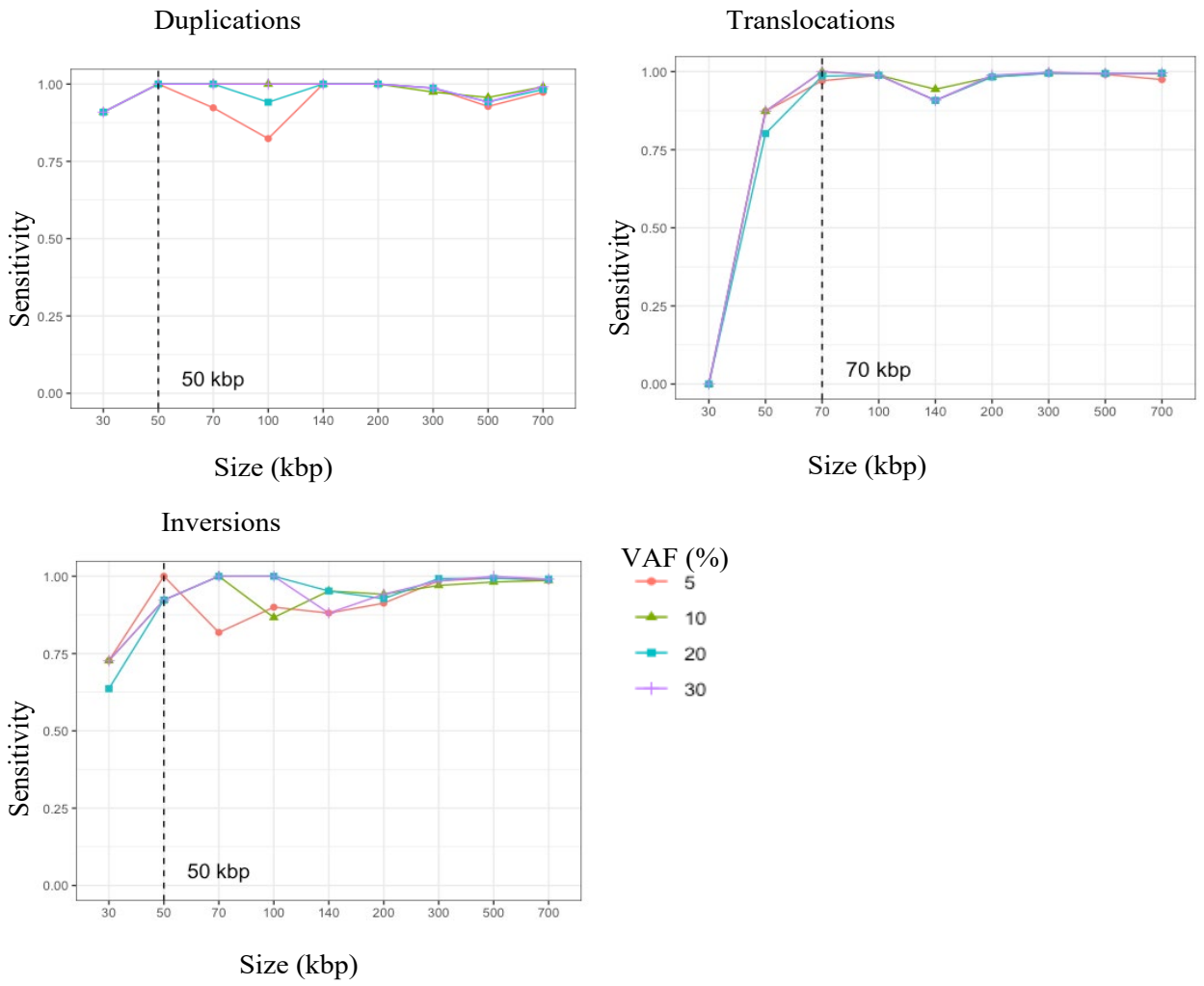



**Figure 17**. Duplication, translocation breakpoint, and inversion breakpoint calling performance with simulated 300X DLE-1 data at different variant allele frequencies. Inversion assessments are reported with a confidence score cutoff of zero.

### SUMMARY – GUIDED ASSEMBLY

We also used simulated data and real data to assess SV detection performance of the guided assembly pipeline for constitutional analysis. DLE-1 molecules were simulated at 80X effective coverage from unedited and edited versions of the hg19 reference genome (with 1600 insertions and 1600 deletions from 200 bp to 1 Mbp) and used for assembly and SV calling with the Bionano Solve pipeline.

### PERFORMANCE FOR SIMULATED INSERTIONS AND DELETIONS

DLE-1 molecules were simulated at 80X effective coverage from unedited and edited versions of the hg19 reference genome (with 1600 insertions and 1600 deletions from 500 bp to 1 Mbp) and used for assembly and SV calling with the Bionano Solve pipeline (**Figure 18**).

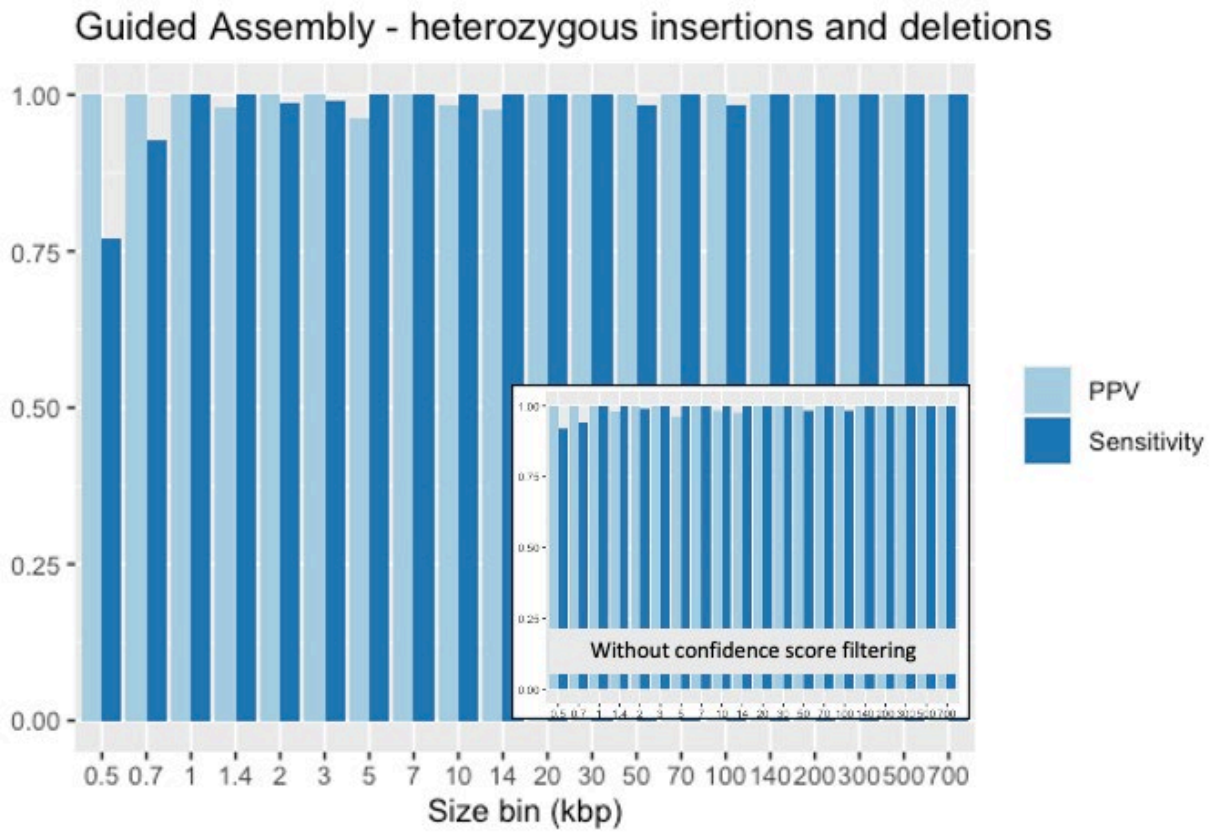



**Figure 18**. Heterozygous insertion and deletion calling performance with simulated 80X DLE-1 data. In/del confidence score is less informative for calls < 1kb; removing confidence score filtering improves sensitivity in this size range without loss of PPV (inset).

### PERFORMANCE FOR OTHER SIMULATED EVENTS

Similarly, DLE-1 molecules were simulated from unedited and edited versions of the hg19 reference genome (with translocated fragments, or transposition events, inversions, and duplications of assorted sizes) and used for assembly and SV calling with the Bionano Solve pipeline. The performance data are shown in **Figure 19.**

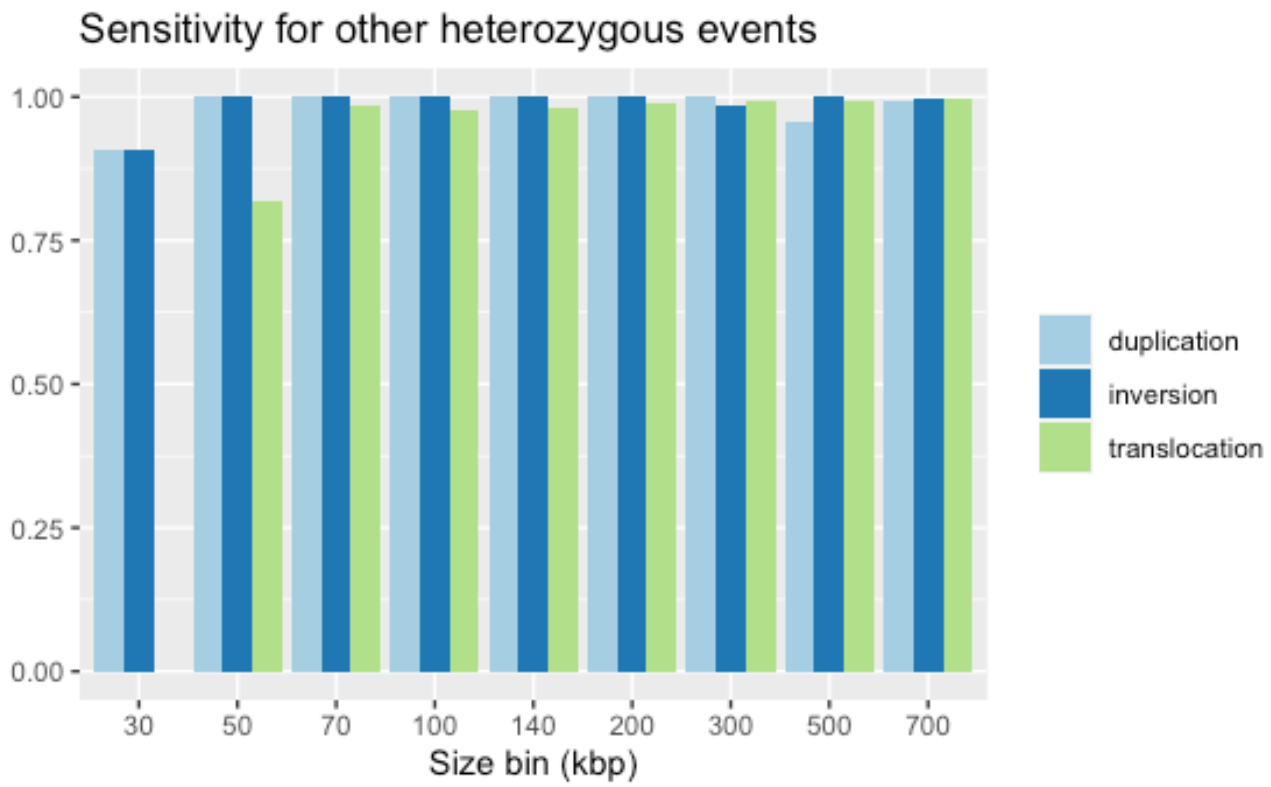



**Figure 19**. Heterozygous sensitivity of various SV types with simulated 80X DLE-1 data

### Concordance with Rare Variant Analysis

A retrospective study using 115 cancer samples that had been previously analyzed with the Rare Variant Analysis pipeline was conducted to evaluate the analytical concordance of Guided Assembly with RVA. Guided Assembly analyses were generated for 55 acute myeloid leukemia (AML) and 60 myelodysplastic syndrome (MDS) samples that had material variants reported in previous analyses. The analysis showed >99% recall of reported variants, with 481/485 SV events reported with similar breakpoints (**Table 8**). Unmatched SVs differed based on refined breakpoints in Guided Assembly.

**Table 8**. Concordance of Guided Assembly LAF with RVA in 115 Cancer Samples

| Structural Variant Type | AML calls (55 samples) | MDS calls (60 samples) | Recall |
|---|---|---|---|
| Deletions | 59/59 | 138/141 | 0.99 |
| Duplications | 32/32 | 73/73 | 1.00 |
| Inversions | 11/11 | 4/4 | 1.00 |

| | | | |
|---|---|---|---|
| Interchromosomal translocations | 47/47 | 104/106 | 0.99 |
| Intrachromosomal fusions | 2/2 | 45/45 | 0.98 |
| Total | 151/151 | 364/370 | 0.99 |

### Concordance with *de novo* Assembly

A similar concordance study was performed using samples that were previously analyzed with *de novo* Assembly and orthogonal methods. Ten samples with 400 Gbp / 80-100x effective coverage and ten samples with 800 Gbp / 160X effective coverage were analyzed with Guided Assembly and SV calls were examined to confirm that previously reported variants were still called. Previous variants were recalled in 100% of samples. Results are presented below in **Table 9** and **Table 10**.

**Table 9**. Known variants detected by Guided Assembly - 80x Coverage

| Sample | Variant(s) | Variant Type(s) |
|---|---|---|
| Sample 1 | ogm [GRCh38] ins(17;3)(p13.3;q29)(2252518/2372750);(197617365_197768658)) | insertion<br>translocation |
| Sample 2 | arr[GRCh37] Yp11.32q12(246520_59331055)x0~1 | mosaic Y loss |
| Sample 3 | ogm[GRCh38] ins(Xq28;6q14.1)((153198191);(76563749_76768226)) | insertion<br>translocation |
| Sample 4 | ogm[GRCh38] 8p23.3p23.1(61805_7678461)x1<br>8p23.1p11.23(12372838_36990558)x3 | deletion<br>duplication |
| Sample 5 | ogm[GRCh37] Xp13(585079_620146) | male hemizygous Xp13 deletion |
| Sample 6 | arr[GRCh37] 15q11.2q13.1(23717628_28513165)x1 | intrachromosomal fusion |
| Sample 7 | ogm[GRCh37] Xp13(585079_620146) | female heterozygous Xp13 deletion |
| Sample 8 | ogm [GRCh38] 12q21.31q24.33(84677954_132112844)x3 | inversion |

| Sample | Variant(s) | Variant Type(s) |
|---|---|---|
| Sample 9 | arr[GRCh37] 2p25.3p25.2(36400_4801965)x3, 10q26.12q26.3(123027564_135403394)x1 | translocation |
| Sample 10 | ogm[GRCh38] ins(2;2)(p16.1;(p11.2)x3)(57360475_57404670;(84990032_85345853)x3) | insertion<br>duplication |

**Table 10**. Known variants detected by Guided Assembly - 160x coverage

| Sample | Variant(s) | Variant Type(s) |
|---|---|---|
| Sample 1 | ogm[GRCh38] 7q22.1q22.1(96099496_102427110)x3 7q22.1q31.2(102691012_115760349)x3 | duplication |
| Sample 2 | ogm[GRCh38] 12p13.33p13.31(14568_6263013)x3 22q13.33(49499507_50805587)x1 | deletion |
| Sample 3 | ogm[GRCh38] 15q11.2q13.2(22291770_30078870)x4 15q13.2q13.3(30109229_31884144)x3 | duplication |
| Sample 4 | ogm[GRCh38] t(10;12)(q25.2;p13.31)(112786550;6245718) | translocation |
| Sample 5 | ogm[GRCh38] 17p13.3(2616531_2651366)x1 | deletion |
| Sample 6 | ogm[GRCh38] 3p25.2(12312742_12587249)x3 16p13.12p13.11(14667883_16373394)x1 | deletion |
| Sample 7 | ogm[GRCh38] 7p22.3(10487_2490744)x3 7p22.37p21.3(2496413_8913689)x3<br>7p21.37p21.1(8915727_19440093)x3 9p24.3(14566_1535031)x1 | duplication<br>translocation |
| Sample 8 | ogm[GRCh38] (X)x1 Xp22.33p21.1(223951_34168552)inv dup(X)(p11.21p11.21)(55,493,463_55,691,156)<br>Xp22.33p11.23(2644445_49275511)x1~2 Xp11.23p11.21(49790932_55484450)x1~2 Xq11.2q25(64634663_127332751)x1~2<br>Xq26.1q28(129718236_151596337)x1~2 | inversion<br>duplication<br>other |
| Sample 9 | ogm[GRCh38] fus(4;4)(q34.1;q35.2)(174077256_188933796 | intrachromosomal fusion |

| Sample | Variant(s) | Variant Type(s) |
|---|---|---|
| | inv(4;4)(q35.2q35.2)(189560372_189799615) | inversion |
| Sample 10 | ogm[GRCh38] (13)x2~3<br>13q22.1q33.2(74793018_104248134)x2~3 hmz | Mosaic T13 |

# Structural Variant Calling Performance: Rare Variant Analysis

Structural variant calling performance of the Rare Variant Analysis pipeline was evaluated using the same simulated dataset as for Guided Assembly LAF. Detailed methods are included in “Appendix A.” Unless otherwise noted, variant calling performance is reported using the recommended confidence score cutoffs for each variant type.

### Structural Variant Calling Performance Using Simulated Data

#### SUMMARY

We observed at least 90% sensitivity at 300X effective coverage at 5% variant allele frequency:

- Insertions between 5 – 50 kbp
- Deletions > 7 kbp
- Translocations (or transpositions where the sizes of the translocated fragments are > 70 kbp)
- Inversions > 70 kbp
- Duplications > 70 kbp

#### PERFORMANCE FOR SIMULATED INSERTIONS AND DELETIONS

Detection performance for insertions and deletions is shown in **Figure 20**.

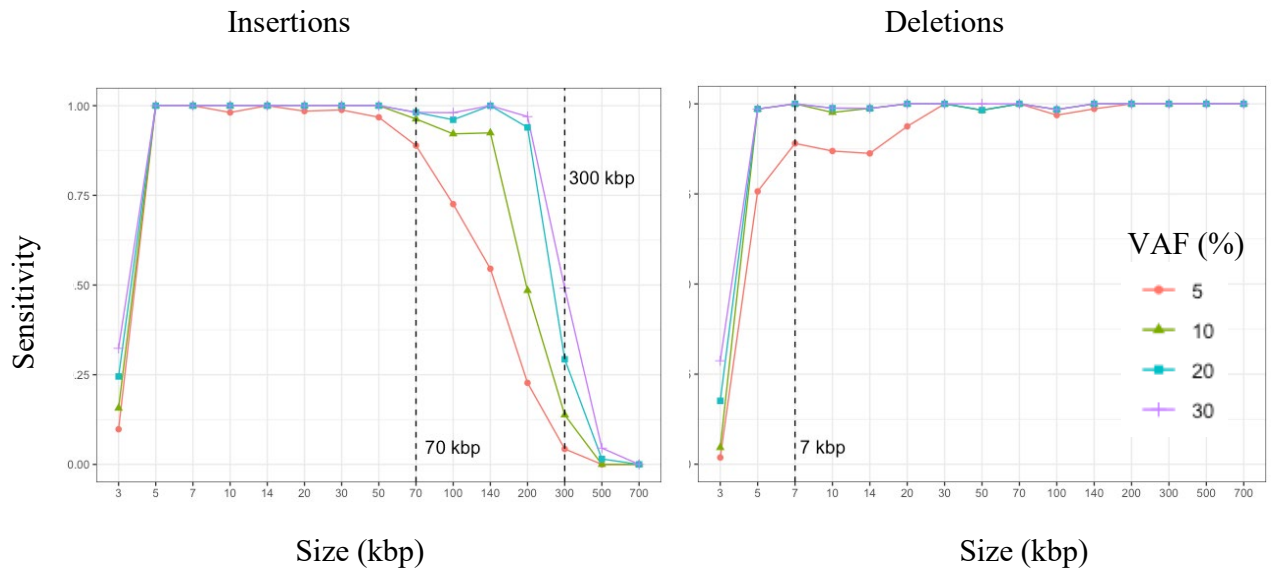



**Figure 20**. Insertion and deletion calling performance of Rare Variant Analysis with simulated 300X DLE-1 data at different variant allele frequencies.

## PERFORMANCE FOR SIMULATED DUPLICATIONS, TRANSLOCATION BREAKPOINTS, AND INVERSION BREAKPOINTS

Performance data are shown in **Figure 21**.

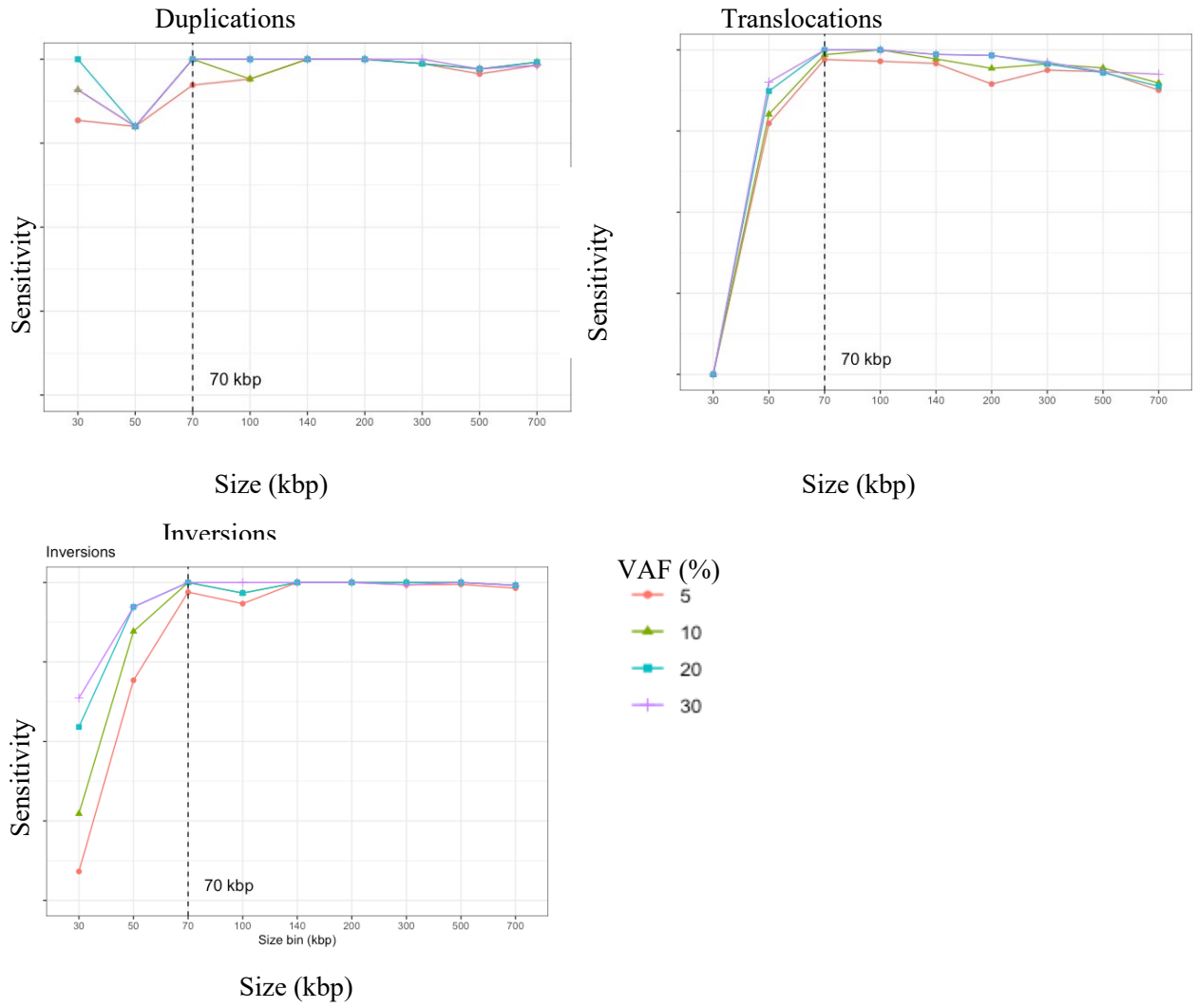



**Figure 21**. Duplication, translocation breakpoint, and inversion breakpoint calling performance with simulated 300X DLE-1 data at different variant allele frequencies. Inversions and translocation assessments are reported with a confidence score cutoff of zero for RVA.

# Structural Variant Calling Performance: *de novo* Assembly Pipeline

SV detection performance of the *de novo* assembly pipeline was assessed using the simulated data used to evaluate Guided Assembly for constitutional applications.

### PERFORMANCE FOR SIMULATED INSERTIONS AND DELETIONS

Detection performance for insertions and deletions using *de* novo Assembly is shown in **Figure 22**.

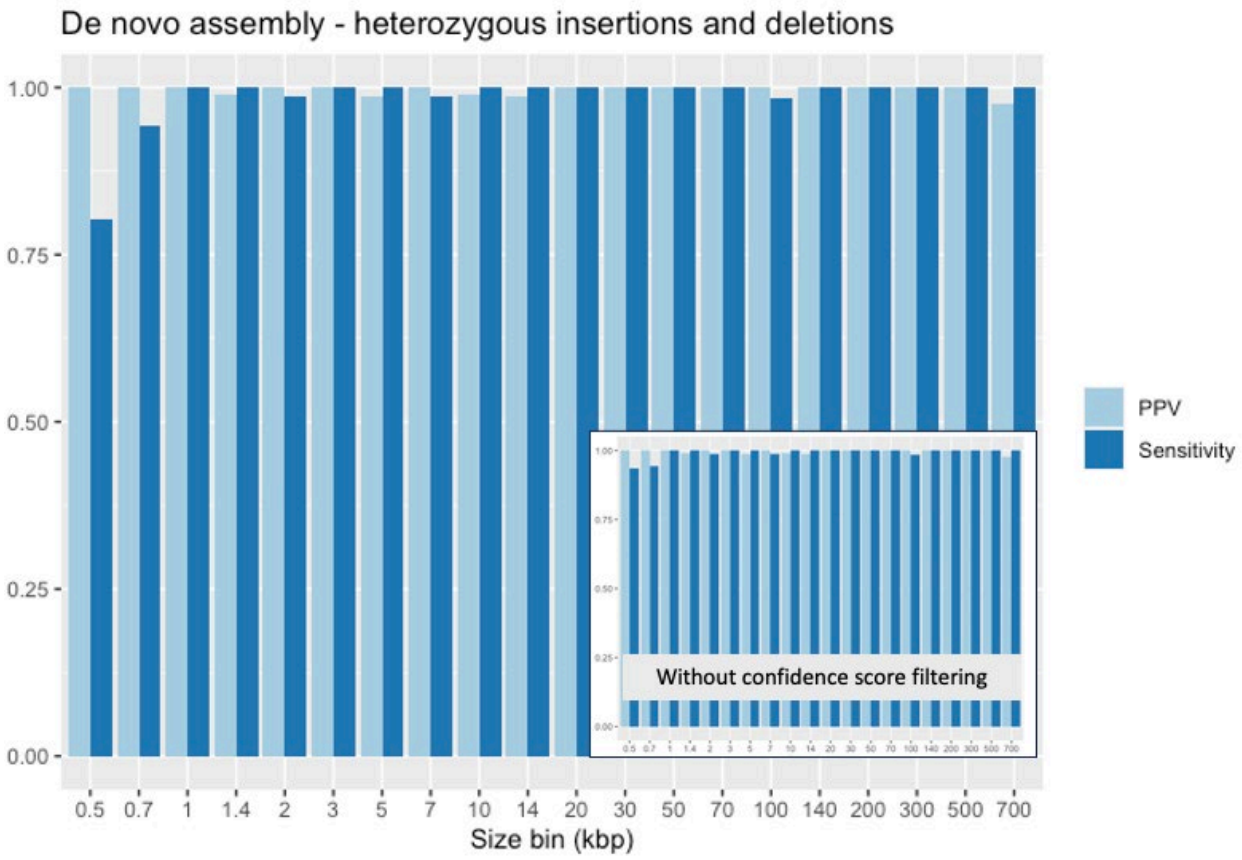



**Figure 22**. Heterozygous insertion and deletion calling performance with simulated 80X DLE-1 data. In/del confidence score is less informative for calls < 1kb; removing confidence score filtering improves sensitivity in this size range without loss of PPV (inset).

### PERFORMANCE FOR OTHER SIMULATED EVENTS

The performance data for inversions, duplications and translocations using *de novo* Assembly are shown in **Figure 23**.

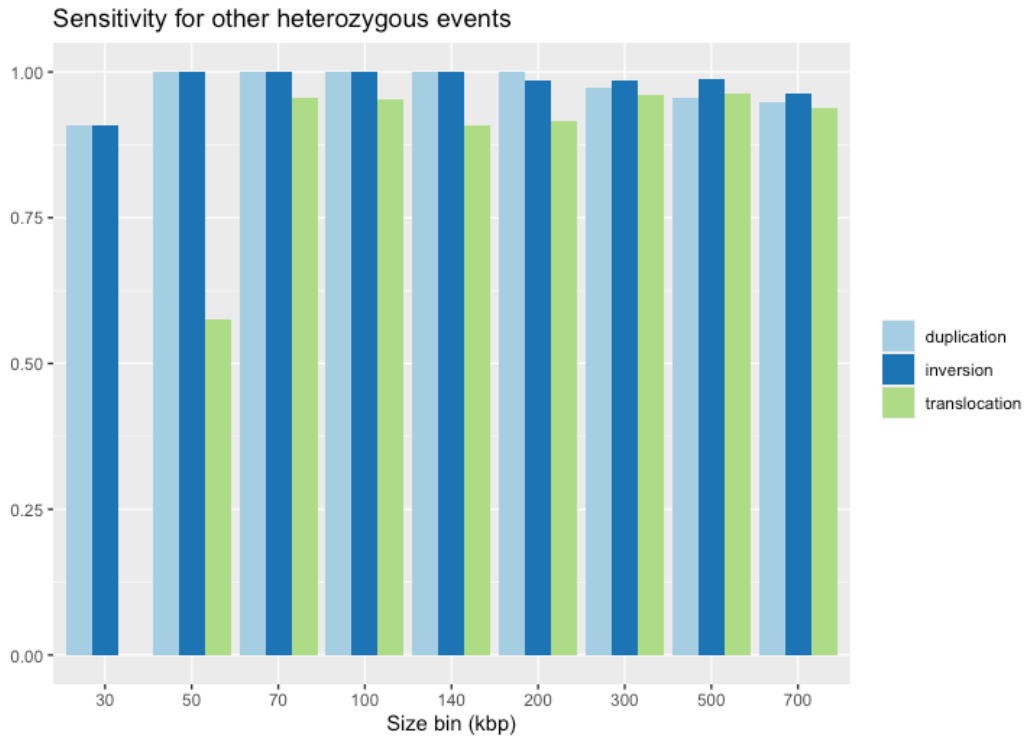


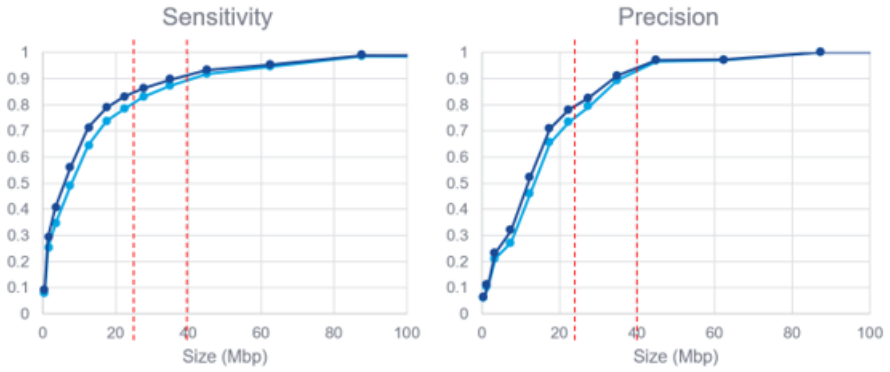

**Figure 23**. Heterozygous sensitivity of various SV types with simulated 80X DLE-1 data.

# AOH/LOH and CNV Calling in VIA

Analysis of OGM data in VIA software includes the capability to apply a new algorithm, SNP-FASST3, for the detection of CNVs and AOH. Detailed description of the algorithm concept and performance data leveraging the simulated data is provided in the document *VIA software Theory of Operations* (CG-00042). The following performance data are for AOH/LOH detection native to the *de novo* and guided assembly pipeline and the FractCNV algorithm which is part of all whole genome pipelines. These data are visible through Access.

# AOH/LOH Detection Performance

Bionano evaluated the performance of AOH detection in two ways. First, we simulated datasets by splicing together data from healthy controls and haploid genomes. We found that we were able to detect AOH regions in the 40-50 Mbp size range with 92% sensitivity and 97% precision, where true positives were defined as an 80% overlap between simulated and predicted regions (**Figure 24**).




| Aberrations (Total=23000) | Sensitivity (Call rate %) 50% overlap | 80% overlap |
|---|---|---|
| >40 Mb | 96.6% | 95.9% |
| 30-40 Mb | 89.6% | 87.3% |
| 25-30 Mb | 86.3% | 83.1% |
| 20-25 Mb | 83.1% | 78.5% |
| 15-20 Mb | 78.9% | 73.8% |
| **Precision (Accuracy)** | | |
| | 50% overlap | 80% overlap |
| >40 Mb | 98.6% | 98.4% |
| 30-40 Mb | 91.2% | 89.4% |
| 25-30 Mb | 82.7% | 79.2% |
| 20-25 Mb | 78.1% | 73.3% |
| 15-20 Mb | 70.9% | 65.4% |

**Figure 24**. Performance of AOH/LOH detection with simulated human data, for simulated events of a given size.

Based on this evaluation, calls less than 25Mbp are not reported unless they are within 25 Mbp of another call. Second, we ran AOH/LOH detection on samples with known events that were previously identified using microarrays. In three samples with four events greater than 25 Mbp, 4/4 events were called correctly, some with 94% overlap, and some were called as multiple smaller regions with a total of 60-70% overlap between the known and predicted regions (**Table 11**). A fourth sample had four smaller known events, three of these were detected and one was filtered out from the output due to its small size. The detected AOH regions that were smaller than 25 Mbp were not filtered out from the output because there were other small calls nearby.

**Table 11**. Performance of AOH/LOH detection with constitutional human samples. Large events (>25 Mbp) are in bold, all large events were detected.

| Sample | CNLOH identified using microarray | AOH/LOH detection results from OGM | Putative FP calls |
|---|---|---|---|
| 1 | 7q11.22qter(71748536_159119707)x2 hmz, [CNLOH 87 Mb; EZH2] | 7: 76,496,409 - 143,709,638 (67.2 Mbp)<br>7 :144,329,449 - 159,345,973 (15.0 Mbp)<br>**Total 82.2 Mb (94.5% overlap)** | 1 = 14.9 Mbp<br>6 < 6.5 Mbp |
| 2 | 7q22.1qter(98411980_159119221)x2 hmz[0.7], [CNLOH 60.7 Mb; CUX1 and EZH2] | 7:101,355,164 - 108,191,148 (6.8 Mbp)<br>7:124,805,963 - 143,709,638 (18.8 Mbp)<br>7:148,325,410 - 159,345,973 (11.0 Mbp)<br>**Total 36.8 Mb (60.6% overlap)** | 1 = 17.8 Mbp<br>5 < 6.2 Mbp |
| 3 | 11q12.1qter(57994955_134942626)x2 hmz, [CNLOH 77 Mb; CBL]<br>13q13.1qter(32553841_115107733)x2 hmz [CNLOH 83 Mb] | 11:56,370,292 – 99,821,083 (43.5 Mbp)<br>11:21,178,985 – 29,596,627 (8.4 Mbp)<br>11:33,078,807 - 35,086,622 (2.0 Mbp + 1.9 Mbp)<br>**Total 53.7 Mb (69.7% overlap)**<br><br>13:27,998,292 – 114,364,328 (86.4 Mbp)<br>**98.6% overlap, reciprocal overlap is 94.7%** | 13 < 8.5 Mbp |
| 4 | 1q25.3q31.1(181,349,929-189,805,990)x2 hmz<br>5q23.1q31.3(119,409,742-139,707,439) x2 hmz<br>14q24.3q32.11(78,367,380-91,128,309)x2 hmz<br>22q12.1q12.3(27,696,986-34,194,745)x2 hmz | 1:182,268,699 – 207,724,864 (25.4 Mbp)<br>14:73,748,304 - 95,896,776 (22.1 Mbp)<br>22:27,136,865 – 39,043,969 (11.9 Mbp) | 16 < 7.2 Mbp |

AOH/LOH calling has been validated only for constitutional disorders. Additional development and validation are needed to enable AOH/LOH detection in cancer samples.

# Copy Number Variant Calling Performance: Fractional CN Analysis

CNV detection performance was assessed using simulated data. Detailed methods are included in the Appendix A section Copy number variant calling performance using simulated data.

CNV detection performance for events ranging from 175 kbp to 6.5 Mbp was assessed using *in silico* mixtures of simulated datasets. The fractional copy number pipeline provides improved sensitivity to smaller events and to lower variant allele frequency events (**Figure 25**).

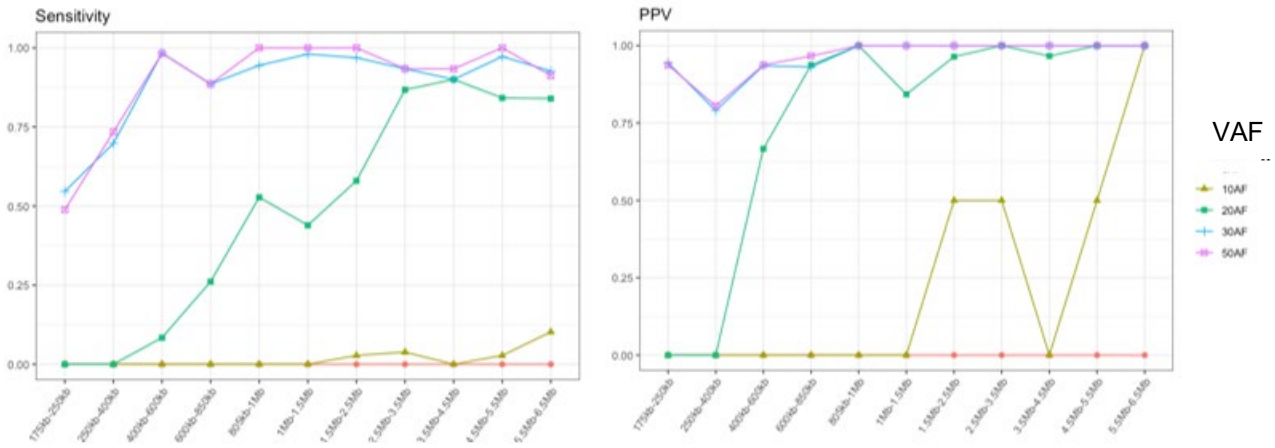



**Figure 25**. CNV detection sensitivity and PPV with simulated 300X DLE-1 data.

Breakpoint accuracy and VAF estimation accuracy are shown in **Figure 26**. The accuracy improves with variant allele frequency. Intuitively, changes in the copy number profiles are more obvious when VAF is higher, making it easier to infer breakpoint locations and VAF.

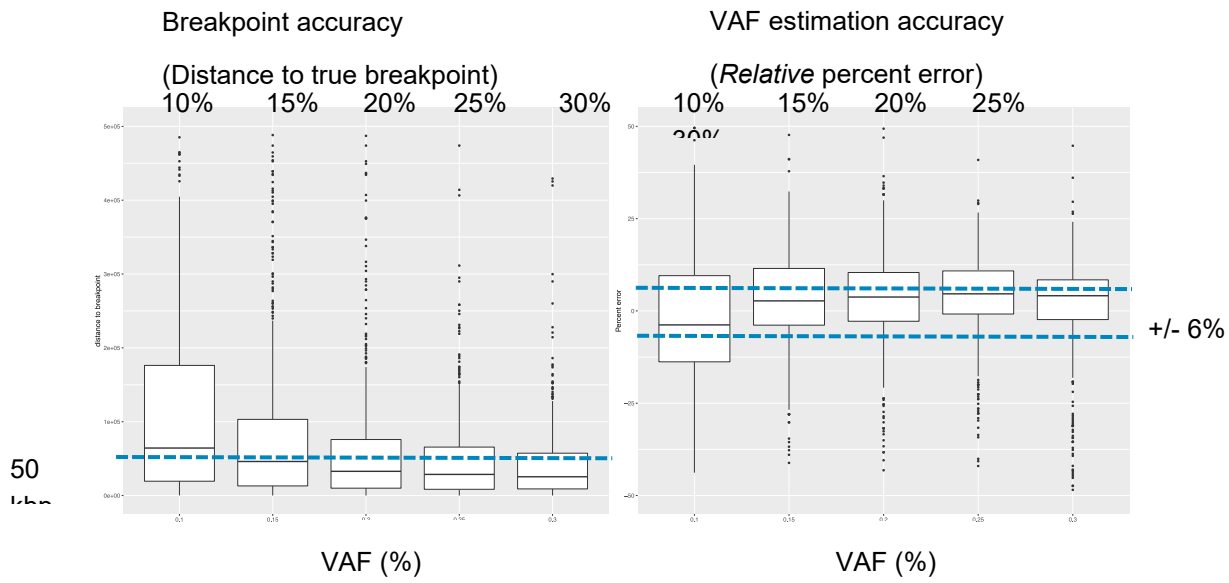



**Figure 26**.Breakpoint and VAF estimation accuracy with simulated 300X DLE-1 data.

# Copy Number Variant Calling Performance: Chromosomal Aneuploidy Detection

### Chromosomal Aneuploidy Detection Performance Using Simulated Data

Chromosomal aneuploidy events were simulated for each of the 22 autosomes at VAFs 5-50% at 1.5 Tb / 300x effective coverage. Performance is shown in **Table 12**.

**Table 12**. Chromosomal aneuploidy performance with simulated 300X DLE-1 data for events at 5-50% allele frequency.

| Variant Type | VAF | True Positives | False Negatives | False Positives | Sensitivity | PPV |
|---|---|---|---|---|---|---|
| Gains | 5% | 16 | 6 | 0 | 72.7% | 100% |
| | 10% | 20 | 2 | 0 | 90.9% | 100% |
| | 20% | 22 | 0 | 0 | 100% | 100% |
| | 30% | 22 | 0 | 0 | 100% | 100% |
| | 50% | 22 | 0 | 0 | 100% | 100% |
| Losses | 5% | 14 | 8 | 0 | 68.2% | 100% |
| | 10% | 19 | 3 | 0 | 86.4% | 100% |
| | 20% | 20 | 2 | 0 | 90.9% | 100% |
| | 30% | 21 | 1 | 0 | 95.5% | 100% |
| | 50% | 22 | 0 | 0 | 100% | 100% |

### Chromosomal Aneuploidy Detection Performance using Real Data

Entire-chromosome aneuploidy events were detected in 7/7 constitutional samples with known events (**Table 13**). Single-arm events are not detected. Cancer samples had lower sensitivity and some had a high number of false positive calls.

**Table 13**. Chromosomal aneuploidy performance with constitutional and cancer 300X DLE-1 data.

| Sample Type | Annotation | TP | FN | Comments |
|---|---|---|---|---|
| Constitutional | trisomy 21 | 1 | 0 | Called |
| Constitutional | trisomy 13 | 1 | 0 | Called |
| Constitutional | trisomy 13 | 1 | 0 | Called |
| Constitutional | mosaic Y loss | 1 | 0 | Called |
| Constitutional | XYY | 1 | 0 | Called |
| Constitutional | XXX | 1 | 0 | Called |
| Constitutional | trisomy 13 | 1 | 0 | Called |
| Cancer | chr 6 loss<br>chr 14 gain | 2 | 0 | Called |
| Cancer | chr12 gain | 1 | 0 | Called |
| Cancer | chr 19 loss<br>chr 20 gain | 1 | 1 | chr20 gain called; chr19 loss not called. Cancer sample shows large coverage variation that complicates calls |
| Cancer | chr 3, chr 4 and chr 6 duplicated | 3 | 0 | Called |
| Cancer | whole-chr 4 deletion | 1 | 0 | Called most chromosomes |

# Bionano Access Integration

With Bionano Access, users can select from the dropdown menus the set of assembly parameters for *de novo* assembly, depending on the application. Bionano Access enables import, filtering, and visualization of SV and copy number variant calls. Users can import a mask file to annotate insertion and deletion calls and filter out likely false positive translocation calls. We provide different versions of the mask to support the hg19, hg38 and T2T-

CHM13v2.0 human reference assemblies and the enzymes recommended for human analysis (Nt.BspQI, Nb.BssSI, and DLE-1). We also provide masks for the mouse reference assemblies mm10 and mm39 with the DLE-1 enzyme. The sets of masks for the *de novo* assembly pipeline and the RVA are separately maintained. Users can also import custom gap files.

# VCF Conversion

The Python-based VCF converter supports conversion of both SV and CNV variant calls. The resulting VCF output is compliant with VCF format version 4.2. For more information, please refer to “Appendix G” and *OGM File Format Specification Sheet CG-00008*. The uncertain breakpoint is indicated in the CIEND and CIPOS fields. It can take SMAP files and CNV output from the *de novo* assembly pipeline, Rare Variant Pipeline, and Variant Annotation Pipeline. Variant annotations are included in the VCF output.

Consistent with the VCF v4.2 format, the VCF file output passed testing with `vcf-tools/vcf-validator`. The information that is added to the output includes the reference accession and additional annotation.

# PAR Masked Reference

Solve 3.8 introduces new versions human reference genomes that mask out the pseudoautosomal regions (PAR) on chromosome Y. This is done to address the sequence homology in these regions with the corresponding regions on chromosome X which can interfere with map and molecule alignments to the reference. Masking of these region is done by removing labels from the reference CMAP. Analysis has shown that this can improve structural variant detection for genes such as *CRLF2* that are in or near the region. This approach is like methods used in whole genome and exome sequencing and works by reducing ambiguous alignments between the chromosomes. Masked references are provided as additional, recommended options for hg19 and hg38 in Bionano Access™ and by default for T2T-CHM13v2.0. Solve 3.8 control databases have been analyzed using these masked references. Structural variant calls from analyses done with the unmasked reference can be annotated using these control database versions, however, calls in the PAR I and II regions of the Y chromosome may appear to be overly rare or unique in the control database.

# Additional Considerations

SV sensitivity generally improves with depth of coverage. The minimum recommended effective coverage (defined as the product of raw coverage and molecule alignment rate) for a diploid sample is 80X, assuming that the quality and average length of the input molecules are good. More coverage will yield small additional sensitivity for homozygous and heterozygous SVs with diminishing returns beyond 120X.

For mosaic/heterogeneous samples, Guided Assembly (LAF) or RVA should be used for detection of low allelic fraction SVs. Data provided in this document generally uses the output of a single Saphyr® 2x1300 or 3x1300 flow cell, i.e., 1.3 Tbp. It is recommended to collect 1.5 Tbp of data to ensure that the pipelines provide reliable 5% allele fraction performance even with sub-optimal samples. While additional coverage can further improve sensitivity, performance at higher coverage tiers above 1.5 Tbp has not been systematically evaluated.

The Bionano Solve pipeline was validated on human diploid and heterogeneous samples.

# FAQs

1. How does coverage affect heterozygous and homozygous SV calling performance using the *de novo* assembly pipeline?

Based on consideration of SV calling performance, runtime, and system throughput, we recommend a minimum of 80X effective coverage for haplotype-sensitive assembly for all SV types. Sensitivity does increase with coverage, but PPV is comparable across coverage levels (the lowest coverage level tested was 50X effective coverage).

2. What is the sensitivity to small insertions and deletions?

We recommend that users focus on insertions and deletions larger than 500 bp.

3. Why do alignment boundaries appear to be off sometimes?

Based on visual inspection, we noticed cases where there was room for improvement in the accuracy of the alignment boundaries. Even though we correctly detected that there was an event, there was an impact on the SV boundaries for these calls. We have optimized parameters to improve the accuracy of SV boundaries and maintain overall performance.

4. Do we classify translocations?

A translocation can be balanced or unbalanced, and reciprocal or non-reciprocal. We currently detect single translocation breakpoints and do not attempt to further classify them. Also, orientation information is currently encoded in SMAP. Users may infer the orientation when visualizing the calls in Access. Additional custom secondary analysis may aid pairing and classification of translocation breakpoints.

5. What are the different versions of cluster parameters and assembly parameters files?

Different versions of cluster parameters and assembly parameters files were created for supporting different platforms and different applications. Cluster parameter files are distributed with the python assembly pipelines; assembly parameter files are distributed with `RefAligner` and RVA. To facilitate selection, dropdown menus are available in Access when an analysis is set up.

6. How are reciprocal translocation breakpoints detected?

Each translocation breakpoint of a reciprocal translocation is independently detected. We currently do not pair potential reciprocal translocation breakpoints. However, translocation orientation information is provided; translocation breakpoints in the same region and with opposite orientations may be part of a reciprocal translocation.

7. How does the assembler handle ambiguity associated with segmental duplication regions?

Large segmental duplication regions, a type of CMPRs, appear at least twice in the genome and are connected to different sequences (potentially on different chromosomes). If a given segmental duplication region is larger than an average molecule's length, the assembler likely does not have enough information to resolve the inherent ambiguity in the connectivity. Maps containing CPMRs larger than 140 kbp are split by default; there is also the option to not split those maps.

8. How are masked translocation breakpoint calls annotated?

A suffix ("_common" or "segdupe") would be appended to any masked translocation breakpoint call in the SMAP SV detection output. A suffix of "_common" indicates that the call overlaps with translocation breakpoints detected

in genomes not known to contain translocations. A suffix of "_segdupe" indicates that the call overlaps annotated segmental duplication regions. If a breakpoint overlaps with both types of masked regions, "_common" would be appended.

9. How do I create a custom mask?

Users interested in generating custom masks are encouraged to model the custom masks based on the masks that Bionano provides. The Bionano masks are in BED format (link). They can be opened and edited using a text editor. They contain three types of BED entries. The three sources of data were pre-processed separately and concatenated together to generate the final mask BED files. The data include 1) N-base gaps, 2) annotated segmental duplication regions, and 3) translocation breakpoint regions detected in control genomes not known to contain translocations. The chromosomal locations of N-base gaps can be obtained from analyzing a given reference fasta file and outputting locations of the N bases (this is output by the *in silico* digestion tool included in Bionano Access). The list of annotated segmental duplication regions (> 50 kbp) for human can be obtained from the UCSC Genome Browser database. When we pre-processed the regions, we further verified map-level similarity for these regions. Using a database of control samples, we defined a list of translocation breakpoints that we detected in more than a specified number of genomes.

10. How are translocation and inversion confidence scores different from insertion and deletion confidence scores?

Confidence scores for insertions and deletions are computed as PPV estimates; the confidence scores for inversion and translocation breakpoints are computed as probabilities from the machine learning models. Because they were trained using different methods and training data, they require different thresholds.

11. What is the expected runtime performance?

Below are representative runtime data (**Table 14**)

**Table 14**. Representative runtime data for a Stratys™ Compute with Saphyr datasets.

| | Enzyme | Collected data | Effective coverage | Stratys Compute | Pipeline |
|---|---|---|---|---|---|
| **Sample 1** | DLE-1 | 400 Gbp | 80 x | 5.8 hrs | Guided Assembly |
| **Sample 2** | DLE-1 | 800 Gbp | 160X | 8.6 hrs | Guided Assembly |
| **Sample 3** | DLE-1 | 1.5Tbp | 300X | 9.1 hrs | Guided Assembly – LAF |
| **Sample 4** | DLE-1 | 1.5Tbp | 300X | 9.9 hrs | Guided Assembly – LAF |
| **Sample 5** | DLE-1 | 1.5Tbp | 300X | 9.8 hrs | Guided Assembly – LAF |
| **Sample 6** | DLE-1 | 1.5Tbp | 300X | 12 hrs | Guided Assembly – LAF |

**Table 15**. Representative runtime data for a Stratys™ Compute with Saphyr datasets.

| | Enzyme | Collected data | Effective coverage | Stratys Compute | Pipeline |
|---|---|---|---|---|---|
| **Sample 1** | DLE-1 | 400 Gbp | 80 x | 5.8 hrs | Guided Assembly |
| **Sample 2** | DLE-1 | 800 Gbp | 160X | 8.6 hrs | Guided Assembly |
| **Sample 3** | DLE-1 | 1.5Tbp | 300X | 9.1 hrs | Guided Assembly – LAF |
| **Sample 4** | DLE-1 | 1.5Tbp | 300X | 9.9 hrs | Guided Assembly – LAF |
| **Sample 5** | DLE-1 | 1.5Tbp | 300X | 9.8 hrs | Guided Assembly – LAF |
| **Sample 6** | DLE-1 | 1.5Tbp | 300X | 12 hrs | Guided Assembly – LAF |

See **Table 16** for a Saphyr Compute Gen4 running the pipelines using example human datasets.

**Table 16**. Representative runtime data for a Saphyr Compute Gen4

| | Enzyme | Collected data | Effective coverage | Saphyr compute | Pipeline |
|---|---|---|---|---|---|
| **Sample 1** | DLE-1 | 400 Gbp | 80X | 8 hrs | *De novo* assembly |
| **Sample 2** | DLE-1 | 800 Gbp | 160X | 10 hrs | *De novo* assembly |
| **Sample 3** | DLE-1 | 800 Gbp | 160X | 8.5 hrs | Guided Assembly |
| **Sample 4** | DLE-1 | 1.5 Tbp | 300X | 5 hrs | RVA |
| **Sample 5** | DLE-1 | 1.5Tbp | 300X | 10 hrs | Guided Assembly – LAF |

**Table 17**. Representative runtime data for a Stratys™ Compute with Saphyr datasets.

| | Enzyme | Collected data | Effective coverage | Stratys Compute | Pipeline |
|---|---|---|---|---|---|
| **Sample 1** | DLE-1 | 400 Gbp | 80 x | 5.8 hrs | Guided Assembly |
| **Sample 2** | DLE-1 | 800 Gbp | 160X | 8.6 hrs | Guided Assembly |
| **Sample 3** | DLE-1 | 1.5Tbp | 300X | 9.1 hrs | Guided Assembly – LAF |
| **Sample 4** | DLE-1 | 1.5Tbp | 300X | 9.9 hrs | Guided Assembly – LAF |
| **Sample 5** | DLE-1 | 1.5Tbp | 300X | 9.8 hrs | Guided Assembly – LAF |
| **Sample 6** | DLE-1 | 1.5Tbp | 300X | 12 hrs | Guided Assembly – LAF |

**Table 18**. Representative runtime data for a Stratys™ Compute with Stratys datasets.

| | Enzyme | Collected data | Effective coverage | Stratys Compute | Pipeline |
|---|---|---|---|---|---|
| **Sample 1** | DLE-1 | 400 Gbp | 80X | 7.6 hrs | Guided Assembly |
| **Sample 2** | DLE-1 | 400 Gbp | 80X | 13.8 hrs | Guided Assembly |
| **Sample 3** | DLE-1 | 800 Gbp | 160X | 8.9 hrs | Guided Assembly |
| **Sample 4** | DLE-1 | 800 Gbp | 160X | 14 hrs | Guided Assembly |
| **Sample 5** | DLE-1 | 1.5Tbp | 300X | 12.9 hrs | Guided Assembly – LAF |
| **Sample 6** | DLE-1 | 1.5Tbp | 300X | 12.7 hrs | Guided Assembly – LAF |
| **Sample 7** | DLE-1 | 1.5Tbp | 300X | 10.9 hrs | Guided Assembly – LAF |

| | | | | | |
|---|---|---|---|---|---|
| **Sample 8** | DLE-1 | 1.5Tbp | 300X | 14.1 hrs | Guided Assembly – LAF |
| **Sample 9** | DLE-1 | 1.5Tbp | 300X | 15.5 hrs | Guided Assembly – LAF |

12. Why do I see fewer maps in regions where I have seen many more previously? Am I losing any information?

We have optimized parameters for allele separation. We expect that homozygous regions are now more likely to be represented by a single allele map. Heterozygous regions are expected to be represented by allele maps containing allele-specific SVs.

13. Could I take advantage of the new Solve release with my existing assemblies?

The SV detection step may be re-run on the command line using the new Solve tools to take advantage of improved SV calling performance. After SV detection, one would replace the existing SV output directory with the new SV output directory, compress the entire output, and import into Access for SV visualization and downstream analysis.

The computation of the inversion and translocation breakpoint confidence relies on data only generated in more recent versions of the Solve pipeline. If expected data were not available in the input consensus maps for SV detection, “-1” would be output for inversion and translocation breakpoint confidence.

14. How do I take advantage of having both NLRS and DLS data?

Data generated with a second enzyme can provide a valuable, independent validation of SVs of interest. Although Bionano has not developed and tested a particular workflow, one common workflow is to first discover relevant SVs using DLS and the Variant Annotation Pipeline, as well as, potentially, other data and tools. Then, generate a second dataset and *de novo* assembly for any samples that require validation. It should not be necessary to pass the data through the discovery pipeline. Simply query the SV list from the confirmation sample to find a match. Key considerations are that the SV coordinates may not match exactly because the coordinates are based on the enzyme recognition sites and have a median accuracy of ~3-5 kbp. A second key consideration is that because of coverage dropout from NLRS-produced “fragile sites,” confirmation could fail, i.e., be negative. That result should not be considered a refutation of the SV being tested. Users are recommended to use Bionano Access to inspect the region of interest to determine whether there is sufficient coverage and whether there is any SV “signal” that was not called but could partially confirm the SV (i.e., the map diverging from the reference at the SV location).

15. How close are the detected breakpoint coordinates to the actual coordinates?

The detected breakpoints are typically around 3 kbp from the actual breakpoints (with the 90$^{th}$ percentile at around 11 kbp).

16. How are molecule alignment confidence scores calculated?

Confidence scores are useful for evaluating alignments. We estimate the probability that the labels on a map match the labels on the reference purely by chance and that the maps are unrelated. The scores are calculated as -log10 of the probability. Generally, alignments with higher scores are of higher confidence.

17. How do I interpret SV counts in the informatics reports?

The informatics report displayed in Access contains counts of unclustered and clustered SV calls. Sometimes, the same SV may be present in more than one map. For example, a homozygous SV may be present in both allelic maps. We cluster similar SV calls to estimate the number of SVs more accurately in the assembly. In the full

informatics report (part of the compressed assembly output), SV counts for both pre- and post-clustering are available.

18. How do I run the VCF converter?

The VCF converter is packaged with Bionano Solve. It can be run on the command-line; it is typically run as part of the analysis pipelines.

19. How does the *de novo* assembly pipeline handle high-coverage datasets?

To efficiently assemble high-coverage datasets and to make use of the best-quality data, an adaptive downsampling procedure is applied by default (could be turned off via `RefAligner` options specified in the xml files). Briefly, the "`-minlen`" parameter is increased up to 240 kbp to reduce coverage, assuming that the longer molecules are more informative. If coverage is still above 250X, molecules are further randomly subsampled until 250X is reached. In refineB and extension stages, molecules with lowest confidence alignments are not used if local effective coverage exceeds 80X. In the refineFinal1 stage, molecules with lowest confidence alignments are not used if local effective coverage exceeds 100X.

20. What is the maximum input coverage recommended for Guided Assembly – LAF or Rare Variant Analysis?

300X effective coverage is recommended for typical low variant allele frequency applications, where events are expected at as low as 5% allele fraction. To achieve that performance with most samples, we recommend 1.5 Tbp to be collected on Saphyr® or Stratys™. RVA has been tested with up to 1000X effective coverage (equivalent of 5 Tbp of data collected on Saphyr or Stratys) but performance has not been evaluated systematically. We do not recommend running this pipeline with more than 1000X effective coverage.

21. Why am I seeing small deletions that seemed suspicious (weak molecule support, for example) in the RVA?

We observed that with RVA, PPV was slightly lower (~80%) for deletions under 25 kbp. This was not observed with the *de novo* assembly pipeline.

22. Where can I get information about variant allele fraction?

The allele fraction for each SV call is part of the Bionano Access UI output and can also be found in the output SMAP file, under the column VAF.

23. Why are effective coverage estimates different between MQR and the assembly report?

Slightly different alignment parameters are used for MQR and during the assembly pipeline, so the effective coverage estimates may differ. The coverage values used in this document refer to the MQR estimates. Please refer to *Data Collection Guidelines* (CG-30173) for more details.

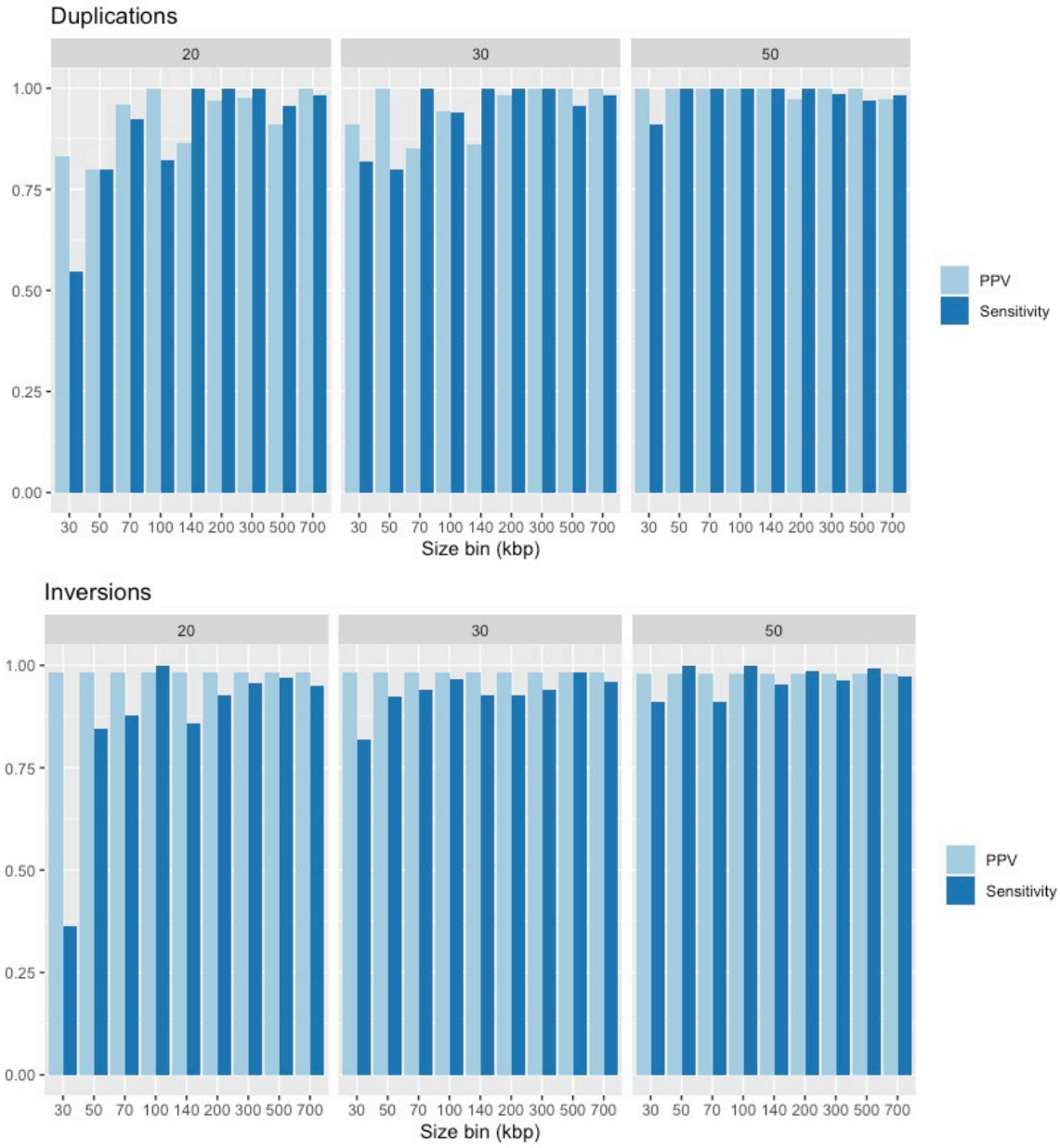
24. Why do I get different results when analyzing the same sample?

There is some non-determinism in the analysis software so that repeated analyses of identical input may result in slightly different outputs. This is due to randomness in the *de novo* assembly and haplotype assignment processes. We have tried to quantify the expected amount of variability (see the Analytical Repeatability section in Appendix A). We observed seven differences in clustered structural variant calls between repeated runs of *de novo* assembly. RVA replicates had identical structural variant calls with minor differences in the SVFreq calculation. These differences may be increased when comparing results produced by analyses performed on different compute hardware. **NOTE**: Technical replicates of samples that include repeating lab procedures and data collection will yield additional differences as each step in the process introduces its own level of variation. We are working on quantifying the expected variance of these type of replicates.

# Appendix A: SV Calling Performance Evaluation

### Guided Assembly SV Calling Performance at Lower Coverage Levels

We evaluated guided assembly SV calling performance at different variant allele frequencies at the 160X (800 Gb) coverage level. **Figure 27** below shows sensitivity performance at 160X effective coverage.



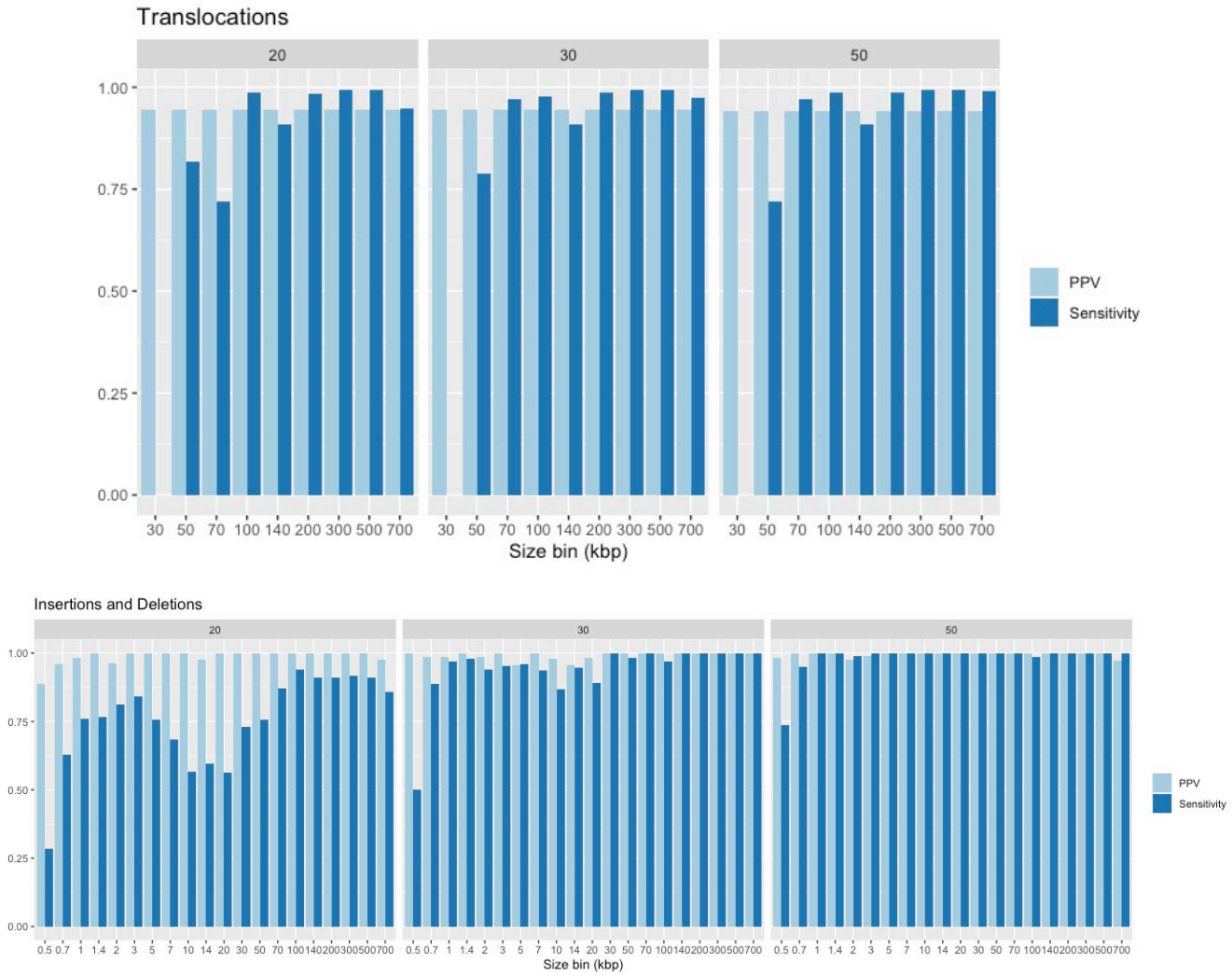
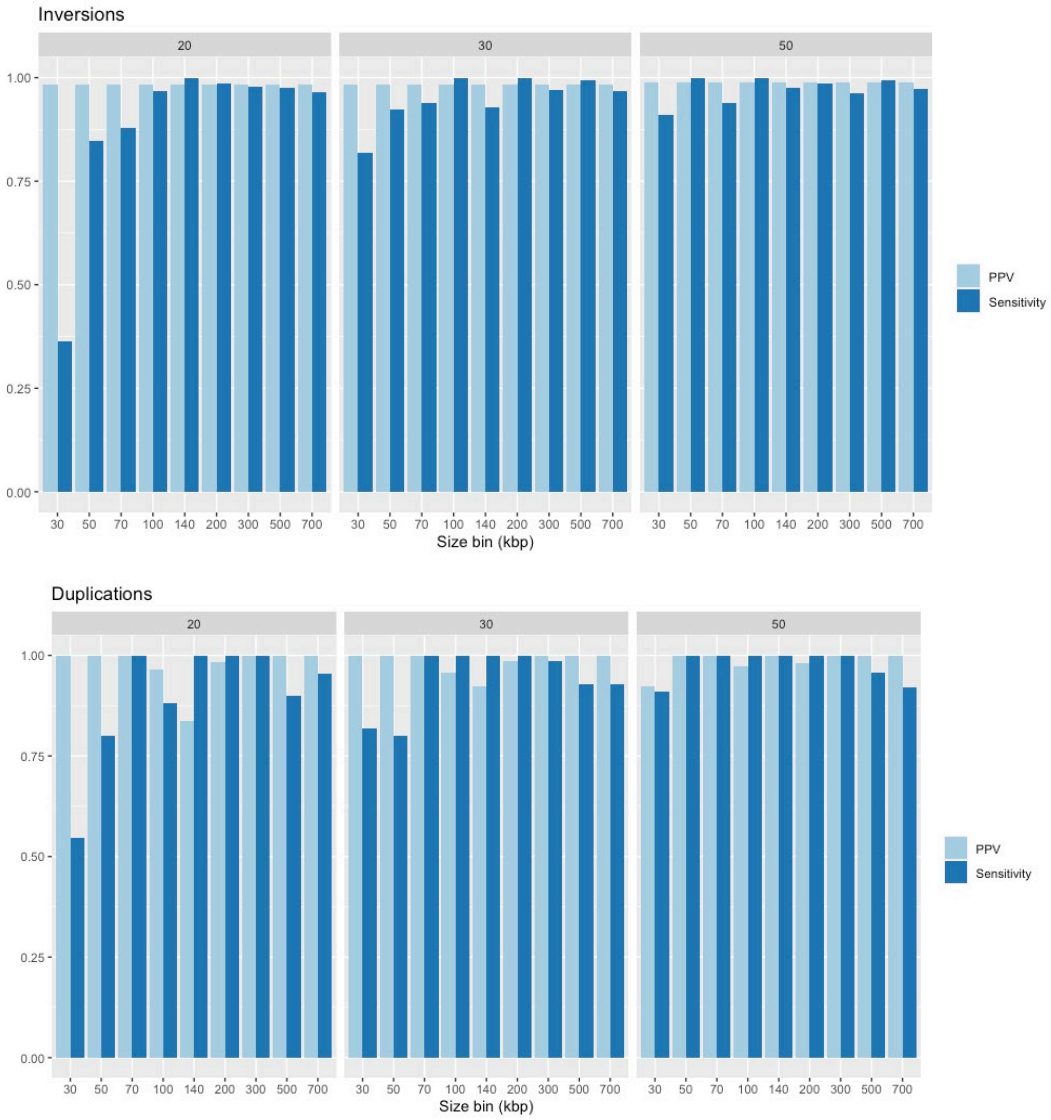



**Figure 27**. Guided Assembly SV calling performance with simulated 160X DLE-1 data at 20%, 30% and 50% variant allele frequencies

### *De novo* Assembly SV Calling Performance at Lower Coverage Levels

We evaluated *de novo* assembly SV calling performance at different variant allele frequencies at the 160X (800 Gb) coverage level. **Figure 28** below shows sensitivity performance at 160X effective coverage.


Inversions
20
30
50
1.00
0.75
0.50
0.25
0.00
30 50 70 100 140 200 300 500 700
Size bin (kbp)
PPV
Sensitivity
Duplications
20
30
50
1.00
0.75
0.50
0.25
0.00
30 50 70 100 140 200 300 500 700
Size bin (kbp)
PPV
Sensitivity

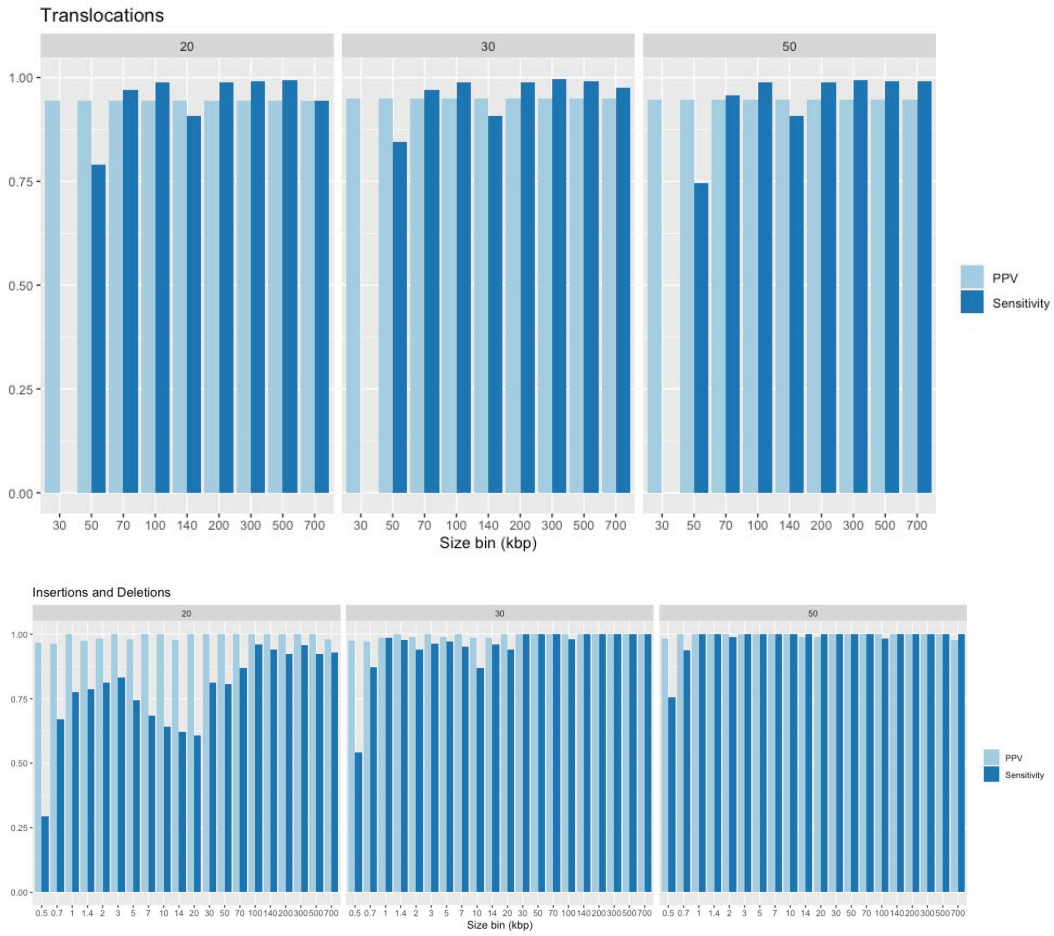



**Figure 28.** SV calling performance with simulated 160X DLE-1 data at 20%, 30% and 50% variant allele frequencies.

## Methods for Assessing SV Calling Performance

### SV CALLING PERFORMANCE WITH SIMULATED DATA

We simulated random SV events so that we could estimate our genome-wide SV calling performance accurately. The human reference assembly hg19 was used as an “SV-free” base genome (in our performance analyses, SVs were called against hg19).

### SIMULATION OF INSERTIONS AND DELETIONS

We randomly introduced 1600 insertions and 1600 deletions into an in-silico map of hg19. The insertions involved addition of new simulated material (random sequences of defined sizes) and deletions involved removing material; we did not simulate replacement of sequence (substitutions). The simulated events were at least 500 kbp from each other or N-base gaps. They ranged from 200 bp to 1 Mbp, with smaller SVs more frequent than larger ones.

Based on the edited hg19, molecules were simulated to resemble actual molecules collected on a Bionano system. This entailed adding sizing error in accordance with the model used in `RefAligner`, outliers created by stitching and DNA knots and folds, and fragile sites. In addition, we simulated molecules from the unedited hg19

and mixed with molecules from the edited hg19 such that all events would be heterozygous. Four such datasets were generated. The simulated molecules were used as input for the analysis pipelines and SV calls were compared to the ground truth.

### SIMULATION OF TRANSLOCATIONS

Random translocation events were simulated to form an edited genome (**Figure 29**). 918 segments were randomly selected across hg19 and randomly inserted elsewhere into the genome. We avoided N-base gaps, and the size of the translocation fragments ranged from 50 kbp to 1 Mbp. Breakpoints were at least 500 kbp away from each other. For simulated intrachromosomal fusions, the breakpoints were at least 5 Mbp away. Each translocated fragment is expected to generate two translocation breakpoints. These can also be considered transpositions.

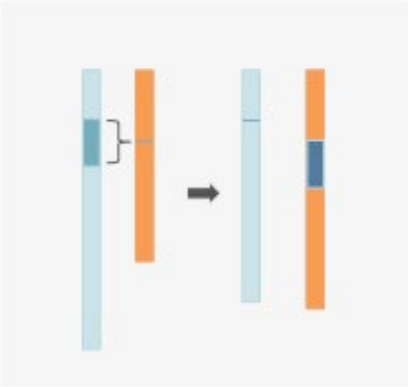


**Figure 29.** Simulation of translocated fragments. Random fragments are removed from a donor chromosome and inserted into an acceptor chromosome.

### SIMULATION OF INVERSIONS

About 900 intervals of 5 kbp to 1 Mbp in size were randomly sampled across hg19 and inverted to create an edited genome with simulated inversions. The inversion events were at least 500 kbp away from each other and were required to not overlap with N-base gaps.

### SIMULATION OF DUPLICATIONS

About 900 intervals of 5 kbp to 1 Mbp in size were randomly sampled across hg19. For each sampled interval, an extra copy of the sequence was inserted in tandem next to the original segment. The new copy could either be in the same or opposite orientation such that performance for detecting tandem and inverted duplications could be assessed. The duplication events were at least 500 kbp away from each other and were required to not overlap with N-base gaps.

### SIMULATION OF MOLECULES

Molecules were simulated from edited genomes according to empirically derived error and size characteristics. Error-free molecules were simulated; then, errors (such as sizing errors and FP and FN labels) were added. Datasets with effective coverage levels were generated at effective coverage levels of approximately 80x, 160x and 300x. These datasets were combined with other simulated samples without SVs at different concentrations to

simulate variants at lower allele fractions (5%, 10%, 20% and 30%). The simulated molecules were used as input for the analysis pipelines and SV calls were compared to the ground truth. SV calls were filtered to include high confidence calls only using the recommended confidence filters per data type.

## Simulation of Large Structural Variants

### DATA SIMULATION

We validated the large SV calling (>200kb) by simulating 726 duplication, deletion, inversion, and intra-chromosomal fusion SVs with varied sizes in hg19 human reference (**Table 19**).

**Table 19**. Simulation Summary

| Type | SVs Simulated | Size | Distance between copies | Distance between SVs |
|---|---|---|---|---|
| **Duplication** | 190 | 200 kb - 1Mb | 300 kb - 7 Mb | 500 kb-1Mb |
| **Deletion** | 175 | 5 Mb - 10Mb | NA | 500 kb-1Mb |
| **Inversion** | 175 | 5 Mb - 10Mb | NA | 500 kb-1Mb |
| **Intra-chromosomal fusion** | 186 | 30 kb - 7Mb | < 1Mb | 500 kb-1Mb |

### ANALYTICAL PERFORMANCE

We measured the sensitivity and precision of large SV calling with 20% size tolerance and 20 kb breakpoint error and showed that the sensitivity and precision across all types are > 95% and > 85% respectively (**Table 20**).

**Table 20**. Large SV calling sensitivity & precision per variant type.

| Type | Sensitivity (%) | Precision (%) |
|---|---|---|
| **Deletions > 200 kbp** | 98.5 | 87.4 |
| **Inversions > 200 kbp** | 98.3 | 97.6 |
| **Duplications > 200 kbp** | 98.1 | 86.1 |
| **Intra-chromosomal fusion > 200 kbp** | 95.5 | 85.1 |

## Terminal Deletions

Solve 3.7 added preliminary support for detection of deletions at the terminal end of chromosomes for human samples only.

### DATA SIMULATION

We validated this module by simulating terminal deletions using the hg38 human reference. This simulation involves 730 synthetic homozygous terminal deletions of sizes > 50Kb in the genomic regions corresponding to 13 terminal deletion syndromes.

### ANALYTICAL PERFORMANCE

We measured the sensitivity of our method per deletion size and per syndrome and showed that the average sensitivity across the syndromes is 95.1% with average sensitivity of > 95.7% on deletions of size >100Kb. Please see **Table 21** and **Table 22** for detailed results.

**Table 21**. Sensitivity/Syndrome

| Syndrome | simulated | called | sensitivity |
|---|---|---|---|
| **1p36 microdeletion syndrome** | 63 | 58 | 92 |
| **2q37 deletion syndrome** | 63 | 62 | 98 |
| **3q29 deletion syndrome** | 54 | 52 | 96 |
| **4p deletion syndrome - Wolf-Hirschhorn** | 63 | 52 | 83 |
| **5p deletion - Cri-du-Chat** | 45 | 45 | 100 |
| **6p25 deletion syndrome** | 49 | 47 | 96 |
| **9q34.3 sub telomere deletion syndrome** | 46 | 46 | 100 |
| **15q26 overgrowth syndrome** | 63 | 59 | 94 |
| **22q13 deletion syndrome - Phelan** | 63 | 61 | 97 |

| Syndrome | simulated | called | sensitivity |
|---|---|---|---|
| **Cat-eye syndrome** | 63 | 56 | 89 |
| **Xq28 microduplication syndrome** | 63 | 56 | 89 |
| **11q deletion - Jacobsen** | 63 | 61 | 97 |
| **1q terminal deletion** | 54 | 49 | 91 |
| **Average** | | | 95.1% |

**Table 22**. Sensitivity per deletion size

| | 50 kb | 100 kb | 250 kb | 500 kb | 1 Mb | 5 Mb | > 8 Mb |
|---|---|---|---|---|---|---|---|
| **simulated** | 55 | 117 | 117 | 117 | 117 | 144 | 63 |
| **called** | 37 | 107 | 114 | 109 | 115 | 137 | 62 |
| **sensitivity (%)** | 67 | 92 | 97 | 93 | 98 | 95 | 98 |

## ARTIFACTS OF THE SIMULATOR

In **Table 23**, we explain artifacts of our molecule simulator affecting the analytical performance of the terminal deletion caller module. Our simulator currently generates low coverage data towards the end of the chromosomes, which limits our analysis in two ways: 1- we could only simulate and analyze the terminal deletions of size > 50kb and 2- the unrealistic low coverage in terminal deletions of size 50Kb causes low sensitivity of 67%.

**Table 23**. Artifacts of molecule simulator

| Artifact | Effects on the analysis | Affects |
|---|---|---|
| **Extreme low coverage towards the end of chromosome.** | Insufficient coverage to call smaller deletion sizes<br>Analysis restricted to > 50Kb | Sensitivity |

## Analytical Repeatability

We repeatedly ran *de novo* Assembly and RVA pipelines with the same input to evaluate the determinism in results.

### RVA

We ran RVA pipeline five times with the same input and compared the outputs of the SV calling, CNV and VAP analysis in a pairwise manner between five runs (resulting in ten different combinations). Based on our results the number of clustered variants called by RVA pipelines is the same in all runs. However, comparing the raw smaps showed that while entries in 4 of 10 pairwise comparisons were identical, in the other six the value of column SVfreq differed by almost 0.001 in 0.24% of the entries. CNV and VAP outputs were identical in all runs.

### *DE NOVO* ASSEMBLY

Like the RVA pipeline evaluation, we ran the *de novo* assembly pipeline five times with the same input and compared the outputs of SV calling, CNV and VAP analysis in a pairwise manner between five runs (resulting in ten different combinations). Based on our results the number of clustered variants called by *de novo* Assembly pipelines differ by at most seven clusters across all runs. Similarly, comparing the raw smaps showed that the number of called SVs was 7717 in 2 of 5 runs and 7734 in the other three runs, differing by seventeen entries. We also compared the contents of the entries in the runs with same number of called SVs and found out that the values of columns Zygosity and VAF differ in at most 12 and 10 entries, respectively. Moreover, while CNV outputs were identical across all runs, VAP outputs differ according to the differences between the smaps.

## Method for Assessing CNV Calling Performance

### CNV CALLING PERFORMANCE WITH SIMULATED DATA

The simulations are used for performance validation and for constructing confident tables. Briefly, we randomly introduced copy number events across the genome. Starting from a molecule-to-reference alignment, molecules that overlap with simulated copy number events are sampled accordingly based on their simulated copy number states. The sampled molecules were used as input to the copy number analysis pipeline, and the detected events were compared with the simulated ground truth to derive sensitivity and PPV data.

### CNV CALLING PERFORMANCE WITH REAL DATA

Samples with known copy number events were analyzed with BspQI, BssSI, and/or DLE-1. The datasets are analyzed using the Solve pipeline, and the copy number analysis output was assessed based on coordinates where CNV events were expected.

# Appendix B: Mask Generation

### Compiling Common Translocation Breakpoint List

Genome map assemblies for control human samples were used as input. Given that they were phenotypically normal samples, we assumed that the control samples did not contain translocations. We performed SV detection on each assembly and detected translocation breakpoints were compiled.

Common translocation breakpoint calls were merged and included into the masks if they were observed in ten or more samples in the control database. Two neighboring breakpoints were merged if they were within 25 kbp of each other. After merging, for each breakpoint or merged region, a 10-kbp buffer was added on both sides. These common breakpoint regions were labeled as “common” in the masks.

### Compiling Annotated Segmental Duplication Regions

Annotated segmental duplication regions of at least 50 kbp were compiled from the UCSC Genome Browser database. Sequences from the annotated segmental duplication regions were extracted and converted into *in silico* maps. For each pair of segmental duplication regions, we checked for map-based similarity. If the *in silico* maps aligned with each other with a p-value of less than 1E-4, the segmental duplication region would be included in the mask and labeled as “`segdupe`”.

### Tools for Generating Mask BED Files

Bionano Solve provides tools for generating custom mask BED files in the “`process_control_datasets`” directory in the Solve build. There are three R scripts in the directory, containing functionalities for creating mask BED files, and checking and applying these masks to an SMAP file. These scripts require the R package "maskTranslocation_0.0.1.0.tar.gz" also included in the directory; they were tested with R version 3.6.3. Selected parameters are described below.

1. **`“makeTranslocationMasks.R”`** is used to generate mask BED files based on a set of control samples.

Example command: Rscript --vanilla path-to-makeTranslocationMasks.R --inputFie inFile --outputDir outDir --reference mm10 --enzyme DLE1 --bedtoolPath path-to-bedtools --minCount 10

**`inputFile:`** a CSV file with the first column containing the names of the samples and the second column the paths for their SMAP files.

**`"reference": "hg19"`** or **`"hg38"`** for human samples and **`"mm10"`** for mouse samples.

**`"enzyme": "DLE1", "BSPQI"`** or **`"BSSSI"`** for human samples, and **`"DLE1"`** for mouse samples.

Two other parameters **`"segdupPath"`** and **`"gapPath"`** are not required for **`"hg19", "hg38"`** or **`"mm10"`** since they were included in the R package. However, if the gap and segmental duplication annotations are updated on the UCSC Genome Browser, they can be extracted and prepared for this input with the script **`"make_bed4masks_translocation.R".`**

2. **`"check_set_transMask2Samp.R"`** is used for either checking masked calls in an SMAP file overlapping with chromosomal locations in a mask BED file, or masking calls in an SMAP file with an input mask BED file.

Example command: Rscript --vanilla path-to-check_set_transMask2Smap.R --smapFile --maskFile --outputDir outDir --check[/set]

"check" and "set" are switches and cannot both be presented in the same command.

**"make_bed4masks_translocation.R"** is a utility script for preparing gap and segmental duplication annotations to be used by the script **"makeTranslocationMasks.R"**.

**Example command:** **Rscript --vanilla path-to-makeTranslocationMasks.R inputFie --outputDir outDir --gap -- refName referenceName [ --segdup --refName referencename --refPath path-to-reference --RefAlignerPath path-to-RefAligner]**

"inputFile": text file downloaded from the UCSC Genome Browser, with gap or segmental duplication annotations for a particular reference genome.

**"gap"** and **"segdup"** are switches and cannot both be presented in the same command. **"refName"** is a shortened name for the reference genome in the parameter **"refPath"**. They could either be **"hg19"**, **"hg38"**, or **"mm10"**. The two other parameters **"segdupSize"** and **"confidence"** are for the parameter **"segdup"**. Their default values (50 kbp and 5) are optimized for the human references (hg19 and hg38).

# Appendix C: Descriptions of Output Files

The following sections detail output files contained in the compressed output of all analysis pipelines. A subset of the files is used for Bionano Access visualization. Some are considered intermediate files that may be helpful for troubleshooting.

### Output Files from the *de novo* and Guided Assembly Pipelines

Here is an example of the output directory structure after unzipping the assembly results (*pipeline_results.zip):

```
output/

output/contigs/alignmolvref

output/contigs/alignmolvref/copynumber

output/contigs/annotation*

output/contigs/auto_noise

output/contigs/exp_refineFinal1

output/contigs/exp_refineFinal1/alignref_final

output/contigs/exp_refineFinal1_sv/merged_smaps
```

*Present if Variant Annotation Pipeline was run

In the output directory:

**exp_informaticsReport.txt** – summary statistics of the assembly stages. Final assembly statistics are in the section "Stage Summary: CharacterizeF_refineFinal1".,

**exp_informaticsReportSimple.txt** – a simplified version of exp_informaticsReport.txt. This is shown as the *de novo* assembly report in Bionano Access.

**exp_informaticsReportSimple.json** – contents of exp_informaticsReportSimple.txt in JSON format.

**exp_optArguments.xml** – parameters used in the assembly.

**exp_pipelineReport.txt** – assembly pipeline log file.

**status.xml** - assembly pipeline status log.

In the output/contigs/alignmolvref subdirectory:

**exp_ogm.bam** – BAM file containing molecule-to-reference alignments.

In the output/contigs/alignmolvref/copynumber subdirectory:

**cnv_calls_exp.txt** – unfiltered copy number calls. This is shown in the Copy Number tab in Bionano Access.

**cnv_chrAneuploidy_exp.txt** – per-chromosome aneuploidy calls. This is shown in the **Aneuploidy** tab in Bionano Access.

**cnv_rcmap_exp.txt** - coverage data for each label in the reference (values are not computed for map ends).

**cnv_chr_stats.txt** – per chromosome coverage statistics

**cnv_mask.bed** – BED file used to mask CNV calls

If the Variant Annotation Pipeline was run, the following files will be in **output/contigs/annotation**:

**variants_combine_filters_inMoleRefine1.smap** – structural variants detected against the reference with additional annotation columns.

**variants_combine_filters_inMoleRefine1.ogm.vcf** – annotated structural variant calls in VCF format

The **output/contigs/auto_noise** subdirectory contains error estimation of molecules:

**auto_noise1.errbin** - error estimation of molecules

**autoNoise1_rescaled.bnx** – molecule file generated after scaling and applying other filters (e.g., min len, min label density) defined in the configuration file (exp_optArguments.xml).

In output/contigs/exp_refineFinal1 subdirectory:

EXP_REFINEFINAL1.cmap – the full set of assembled genome maps.

The **output/contigs/exp_refineFinal1/alignref_final** contains alignments between the assembled consensus genome map and the reference:

EXP_REFINEFINAL1.err – human readable error parameter estimates for each EM iteration

**EXP_REFINEFINAL1.errbin** – binary version of error parameter estimates

In output/contigs/exp_refineFinal1_sv subdirectory:

**EXP_REFINEFINAL1_full.xmap** – contains complete set of alignments used for SV calling, before filtering overlapping alignments on query maps. Used for debugging SV calling.

The **output/contigs/exp_refineFinal1_sv/merged_smaps** contains alignments and SV calls between the assembled consensus genome map and the reference:

**exp_refineFinal1_merged_filter_inversions.smap** – structural variants detected against the reference. This is shown in the **SV** tab (no SV filtering) in Bionano Access.

**exp_refineFinal1_merged_r.cmap** – the reference maps used in SV calling.

**exp_refineFinal1_merged_q.cmap** – the genome maps aligning to the reference.

**exp_refineFinal1_merged.xmap** – detailed alignment information used for SV calling between the genome maps and the reference maps.

**sv_mask.bed** – BED file used to mask SV calls

### Output Files from RVA

Here is an example of the output directory structure after unzipping Rare Variant Pipeline results (_pipeline_results.zip):

```
output/
```

`output/data`

`output/data/alignmolvref/`

`output/data/alignmolvref/copynumber`

`output/data/annotation*`

`output/data/auto_noise`

`output/data/consensus_check/extension/refine1/merge`

`output/data/consensus_check/extension/sv/merged_smaps`

*Present if Variant Annotation Pipeline was run

In the ***`output`*** directory:

**`exp_informaticsReportSimple.txt`** – summary statistics of the pipeline stages and SV results. This is shown as the *Rare Variant Analysis Report* in Bionano Access.

exp_informaticsReportSimple.json – contents of `exp_informaticsReportSimple.txt` in JSON format.

**`exp_optArguments.xml`** - parameters used in the analysis run

In the `output/data/alignmolvref/` subdirectory:

**`exp_ogm.bam`** – BAM file containing molecule-to-reference alignments.

In the `output/data/alignmolvref/copynumber` subdirectory:

**`cnv_calls_exp.txt – unfiltered`** copy number calls. This is shown in the **Copy Number** tab in Bionano Access.

**`cnv_rcmap_exp.txt`** – coverage data of each label in the reference (values are not computed for map ends).

**`cnv_chrAneuploidy_exp.txt`** – per-chromosome aneuploidy calls. This is shown in the **Aneuploidy** tab in Bionano Access.

**`cnv_chr_stats.txt`** – per chromosome coverage statistics

**`cnv_mask.bed`** – BED file used to mask CNV calls

If the Variant Annotation Pipeline was run, the following files will be in **`output/contigs/annotation`**:

**`variants_combine_filters_inMoleRefine1.smap`** – structural variants detected against the reference with additional annotation columns.

**`variants_combine_filters_inMoleRefine1.ogm.vcf`** – annotated structural variant calls in VCF format

The **`output/data/autonoise`** subdirectory contains error estimation of molecules:

`autoNoise1.err` – error estimation of molecules

`autoNoise1_rescaled.bnx` – molecule file generated after scaling and applying other filters (e.g., min len, min site density) defined in the configuration file (`optArguments.xml`).

The `output/data/consensus_check/extension/refine1/merge` contains molecules alignments to the assembled genome maps:

`exp_refine1_contig*_r.cmap` – the assembled genome map (the anchor of the alignment, by genome map IDs).

`exp_refine1_contig*_q.cmap` – the molecules aligning to the genome maps (the query of the alignment).

`exp_refine1_contig*.xmap` – detailed alignment information between the molecules and the genome maps.

In `output/data/consensus_check/extension/sv` subdirectory:

`SV_LOCI_CLUSTER_full.xmap` – contains complete set of alignments used for SV calling, before filtering overlapping alignment on query maps. Used for debugging SV calling.

The `output/data/consensus_check/extension/sv/merged_smaps` contains alignments and SV calls between the assembled consensus genome maps and the reference:

`EXP_REFINEFINAL1.smap` – structural variants detected against the reference. This is shown in the SV tab (no SV filtering) in Bionano Access.

`EXP_REFINEFINAL1.cmap` – the full set of maps used in SV calling.

`EXP_REFINEFINAL1_r.cmap` – the reference used in SV calling.

`EXP_REFINEFINAL1_q.cmap` – the genome maps aligning to the reference (may be a subset `EXP_REFINEFINAL1.cmap`).

`EXP_REFINEFINAL1.xmap` – detailed alignment information used in SV calling between the genome maps and the reference. This is shown in the Match tab in Bionano Access.

`sv_mask.bed` – BED file used to mask SV calls

# Appendix D: In-Depth Description of the *de novo* Assembly Pipeline

This is advanced material for users interested in the details of the assembly process. We recommend reading the "*De novo* assembly" in the main text first. The following sections consider features available in the Bionano Solve 3.8 release. Some features mentioned may be missing in earlier releases.

### Setting Up a *de novo* Assembly Run

A *de novo* assembly run can be set up with a "molecules" object in Bionano Access or a BNX molecules file on the command line. For more detail, please refer to the *Bionano Access Software User Guide* (CG-30142) and the *Guidelines for Running Bionano Solve on the Command Line document (CG-30205)* for the two ways of setting up a *de novo* assembly run. A reference *in silico* map (CMAP) can be provided but is optional. If a reference is provided, the molecules and the consensus maps would be aligned to the reference. The alignments could provide helpful information for quality assessment. Also, the reference is necessary for structural variant (SV) calling, and a mask BED file can be selected for annotating the resulting SV calls.

Several default assembly parameter sets are provided for different use cases. The parameter sets are specified in XML files, whose names suggest the target usage. Which parameter set to use depends on several factors. In the Access interface, the dropdown selection menus reflect the specific considerations that impact the selection of parameter sets. For example, certain parameter sets have been optimized for larger genomes (typically > 5 Gbp) that require more computation and memory.

For genome assembly or finishing projects, if a high-quality reference is not available, we recommend using the pre-assembly option. When this option is enabled, a preliminary assembly is constructed first, and this assembly is used as reference for estimating noise parameters in the molecules in the subsequent assembly steps.

For SV analysis projects for typical diploid samples (such as human), we recommend using the haplotype-aware options and cutting complex multiple-path regions (CMPRs) in the map. When haplotype-aware options are enabled, we try to separate possible alleles. Because of the presence of large segmental duplications in the genome, which can confuse the assembler, we recommend cutting maps that cover these CMPR regions to avoid assembly errors.

### Key Stages of the *de novo* Assembly Pipeline

Here are brief descriptions of the key steps in the basic *de novo* assembly pipeline (**Figure 30**).By default, the output data from the intermediate pipeline steps are not imported or visualized in Bionano Access. For Bionano Access runs, the key files in the final steps are compressed before being imported into Access. The intermediate files are only kept temporarily on the compute server to reduce disk space use. However, when needed and available, certain intermediate files may be useful for troubleshooting.

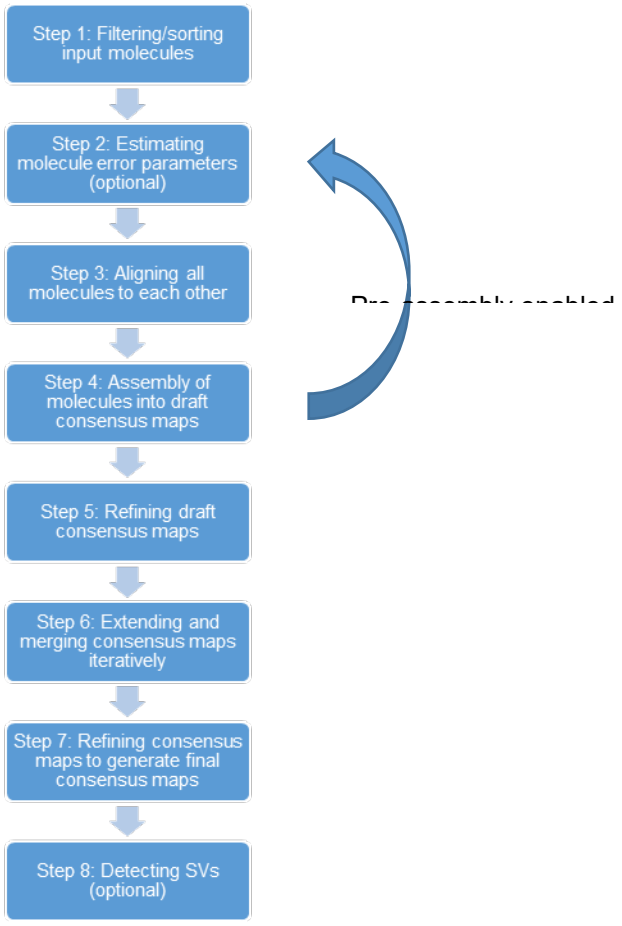



**Figure 30**. *De novo* assembly workflow.

The pipeline uses **RefAligner** and Assembler for alignment and assembly operations. It serves to chain **RefAligner** and Assembler operations together, handle job submissions, and generate intermediate data summaries. In addition to the pipeline report (typically named `exp_pipelineReport.txt`), **RefAligner-** and Assembler-generated output includes information in the headers about the command to generate the output and other useful information (such as the version of the `RefAligner`/Assembler). There are `.stdout` files that log information during `RefAligner`/Assembler operations. Although expected to be rare, assertion errors can occur, and error messages can be found in the `.stdout` files. Please contact Bionano Support if assertion errors are observed.

If the pre-assembly option is enabled, **Steps 1, 3, 4** from below are performed first. Then, the consensus maps would be used as the input reference, and the assembly would then restart from **Step 2**.

1. Filtering and sorting input molecules

The raw input to the assembly pipeline would be an `all.bnx` file typically. Depending on what filters have been applied previously, it might contain short molecules or molecules that do not have sufficient labels for alignment and assembly. Therefore, the pipeline filters out some of the molecules using a length (typically with -minlen 120) and a site (typically with -minsites 9) filter. This filtering step helps filter out molecules that are likely not useful and limit the file size. The pipeline would generate an `all_sorted.bnx` file in the main output directory after the filters are applied. As the .bnx name suggests, the molecules are sorted. The sorting is done based on the numeric molecule IDs.

2. Estimating molecule error parameters (optional)

In the auto-noise stage, the pipeline attempts to align all molecules in the `all_sorted.bnx` file to the reference *in silico* map (.cmap) to estimate molecule noise parameters. The output is stored in the `"auto_noise"` directory under the main output directory. The parameters include for example, the amount of missing or extra labels compared to the reference, and the interval sizing differences. They are represented in the .err files. These parameters are helpful for scoring alignments in subsequent steps. Conceptually, they form an expectation of the errors and help quantify whether an alignment is good or bad. Using a good-quality reference allows the pipeline to set the expectation correctly. If the reference quality is questionable, we would expect the molecules to be different from the reference (even if the molecules were perfect), thus artificially inflating the error estimates.

The alignment is done in several iterations, and the noise estimates are updated in each round. After the iterations, the noise estimates should converge. The very first iteration starts with default noise parameters. In rare cases, if the default parameters are too different from the actual parameters, it is possible that very few alignments are found, making `RefAligner` unable to estimate the parameters. In such cases, one may adjust the default noise parameters in the parameters .xml. If the error rates are too high (because either the molecules or the reference is of inferior quality), the auto-noise stage can take a long time, and the noise estimates may not converge. This might impact subsequent assembly stages.

For computational efficiency, the auto-noise stage is split up into two sub-stages: auto-noise0 and auto-noise1. The output from auto-noise1 is used for subsequent steps.

Another calculation done during the autonoise stage is the determination and correction of the per-cohort stretch and the per-scan stretch factors for the molecules, measured in basepairs per pixel (bpp). During image acquisition, each scan is divided into a number of subgroups (“cohorts”) for real-time analysis. All the molecules in the same cohort or scan (one scan contains multiple cohorts) may be stretched differently on average compared to molecules in a different cohort or scan due to a variety of reasons. Initially a BNX file is generated per cohort, but all BNX files are merged before assembly, so the calculation of the stretch factors aims to correct for these differences. **NOTE**: The correction refers to the overall or average stretch of molecules in a cohort or scan, which is different from local, or per-interval stretch variation in the individual molecules. To convert the pixel-based data from the molecule images into basepairs, the pipeline uses information from the molecule-to-reference alignment. The bpp for each cohort and scan is normalized to account for the average cohort or scan stretch differences. After normalization, the molecules are expected to be more uniform.

The main output of the auto-noise stage are the error parameters (.err) and a rescaled BNX file with stretch-corrected molecules. These are used for subsequent steps. The BNX file is sometimes split into pieces (named with pattern `all_X_of_N.bnx`) for more efficient computation. If a reference is not provided the auto-noise stage is skipped and default error parameters are used instead. This can be sub-optimal if the default error parameters are not appropriate for the dataset of interest. We recommend enabling the pre-assembly option such that the preliminary assembly could be used for noise estimation.

3. Aligning all molecules to each other (pairwise alignment)

The input is the rescaled molecules, and the output is stored in the "align" directory. The pipeline pairwise-aligns the molecules to other molecules to form the basis of the overlap graph needed for the overlap-layout-consensus, OLC, assembly. The output files are binary (.align); they contain information about which molecules are aligned to which molecules, the alignment confidence, and the offsets. The fraction of molecules that have pairwise alignments should be comparable to the map rate. The statistics from the pairwise alignment step are available in the `.stdout` output.

4. Assembling molecules into draft consensus maps

An initial assembly is created from the pairwise alignments of the previous step. The output is a set of draft consensus maps (.cmap files) in the `"exp_unrefined"` directory. The assembler uses the pairwise alignments to form an overlap graph and analyzes it for potential paths. The graph can be quite complex due to both genuine genomic features (such as repeats) and false positive pairwise alignments, so we apply several clean up steps to output the longest paths (which consensus maps represent). The assembly size should be close to the genome size, even though maps continue to be extended and refined in subsequent stages. The N50 (which is an indication of consensus map lengths) does tend to increase in the later stages. If a reference is provided, the consensus maps are aligned to it for quality control purposes.

5. Refining draft consensus maps

The initial refinement of the draft consensus maps is done by analyzing molecule-to-map alignments during the stages refineA and refineB. The input is the draft consensus maps and the output is refined consensus maps. `RefAligner` tries to determine whether the labels on the draft consensus maps need to be adjusted (added, deleted, or moved) and whether the maps need to be split due to lack of molecule support in certain regions of the map. The difference between the stages refineA and refineB is that the former uses only molecules that were in the initial assembly. The latter tries to recruit additional molecules by aligning all molecules to the refineA consensus maps (assumed to be of higher quality) and refining the maps based on the larger set of molecules.

6. Extending and merging consensus maps iteratively

Extension and pairmerge are paired stages that are run iteratively to make the maps more contiguous. Input molecules are aligned to the ends of the consensus maps to "extend" them. After maps are extended, some of them are expected to overlap with each other. The pipeline aligns the extended maps to each other and merges overlapping maps. The map N50 is expected to increase because of this process. This is similar in concept to the gap closing step that some sequence assembly tools perform. By default, the extension and merge process is repeated five times, as the consensus map N50 tends to plateau after 5 rounds.

During the extension step, if the haplotype-aware options are turned on, the pipeline also looks for alternative alleles. If molecules align but diverge from the maps, the pipeline forms new maps that represent alternative alleles. This tends to split off large SVs, allowing them to be detected. Rare alleles such as those found in cancer clones may be found, but the pipeline requires at least a certain number of molecules to support an allele and makes certain assumptions about the minimum fraction of molecules that support an allele. The *de novo* assembly is not optimized for detection of rare alleles. Rare Variant Pipeline would be more suited for that purpose.

In the final merging stage, the CMPR-related option would become active if enabled. `RefAligner` would analyze the pairwise alignments of the consensus maps. If pairs of maps align to each other but do diverge, they likely contain CMPR sequences, and they may be cut.

7. Refining consensus maps to generate the final set of maps

> This stage of final refinement of the consensus maps generates the final product of the assembly pipeline, a set of refined maps. It tends to the be most time-consuming stage. The pipeline aligns the molecules again to the maps and analyzes the alignment to see if any maps need to be updated. If the haplotype-aware options are turned on, the pipeline also looks for alternative alleles. This tends to split off smaller SVs.

The pipeline makes the default assumption that there can be two alleles, so one often sees a pair of maps per genomic region. For genome assembly or finishing projects, the presence of these seemingly redundant maps may create issues during scaffolding. We recommend not using the haplotype-aware assembly options for those projects.

8. Detecting SVs using the final consensus maps (optional)

The final consensus maps are used for SV calling, if a reference is provided. The pipeline aligns the maps to the reference using a multiple alignment algorithm (where all possible alignments are found) and analyzes the differences. The SV calls are recorded in the .smap files.

Runtime performance of the *de novo* assembly pipeline

The runtime performance data are included in earlier sections. Generally, runtime depends on several factors. The amount of input data, the size of the genome, the data quality, and the load of the compute server can contribute to runtime differences. The *de novo* assembly automatically down-samples the dataset if the amount of data is greater than 220X. Please refer to *Data Collection Guidelines* (CG-30173) for more detail. The downsampling step helps limit the runtime for high coverage datasets, and it should not impact the quality of the assembly. The pre-assembly option requires additional steps in the assembly process and might take longer. The haplotype-aware options also lengthen the assembly time because of the extra analyses needed to detect alternative alleles.

Evaluation of *de novo* assembly results

Please refer to Bionano De Novo Assembly Informatics Report Guidelines (CG-30255) for more detail.

# Appendix E: Interpretation of Inversion Breakpoint Calls

## Introduction

All the structural variant (SV) calls are obtained by aligning consensus genome maps (query maps) to a reference map using a Multiple Local Alignment algorithm and analyzing the alignments for SV signatures. Pairs of alignments within a map are analyzed and inconsistencies representing possible SV events between the genome maps and the reference are identified. The signature for inversion breakpoints is a pair of neighboring alignments with opposite orientations (each alignment has an orientation relative to the reference), indicating that two locations (breakpoints) on the reference map are fused in the query map (**Figure 31**). The orientation information is in the XMAP file; it can either be "+" or "-", relative to the reference.

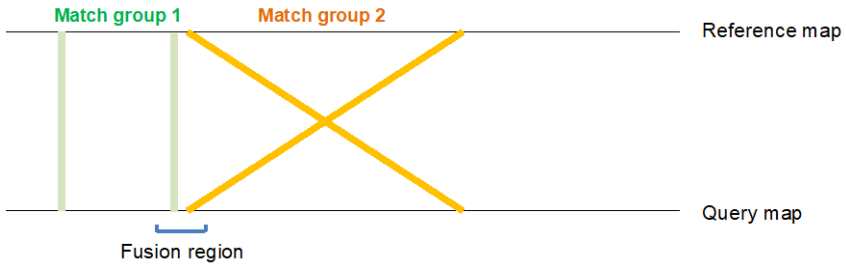



**Figure 31**. Signature of an inversion event. Two alignments from the XMAP file (yellow and green solid lines) are close to each other with opposite orientations.

There are two modules for detecting inversion breakpoints: one for detecting breakpoints of relatively "large" inversions, and one for detecting breakpoints of relatively "small" inversions (typically involving fewer than five labels in the inverted region). Both modules are automatically run during SV detection; there is not an explicit switch or parameter to turn them on. The definitions of large and small inversions are informal but are relevant in the context of inversion breakpoint detection using Bionano data. Whether an inversion is considered small and large depends on whether the map spans both breakpoints of the inversion. A map could span both breakpoints of a small inversion, and both breakpoints are called together. If a map does not span both breakpoints, but there is evidence of an inversion, a single breakpoint is output. The other breakpoint of the inversion may be detected in another map (which likely also does not span both breakpoints of the inversion).

We generally use the term "breakpoints" to describe the two sides of an inversion. Conceptually, an inversion involves two breaks in the sequence (relative to the reference) and a subsequent inversion of that sequence. This could happen in diverse ways biologically. Alternatively, the concept of "fusions" may be helpful. There would be two fusion points where the inverted sequence is joined with the surrounding non-inverted sequence. By analyzing the alignment data, one could infer breakpoints and which sequences are fused together.

SMAP output for inversion breakpoints

The SMAP file summarizes information about the labels involved in an inversion, both from the reference and the query maps. To fully interpret inversion events, we need to also consider the corresponding XMAP entries of the alignments used to make the call. Each line in the SMAP references four coordinates (two on the query map and two on the reference map). The coordinates of the labels in the query and reference maps are given in the `QryStartPos`, `QryEndPos`, `RefStartPos`, and `RefEndPos` columns, respectively. Please refer to *OGM File Format Specification Sheet* (CG-00008*)* for the specific meaning of those columns.

SV types other than inversion breakpoints are represented by single SMAP lines. For inversion breakpoints, additional relevant coordinates are reported. To do so, all inversion breakpoints are represented by pairs of two lines in the SMAP file. The pairing can be identified using the link ID column. The two lines have link IDs that point to each other.

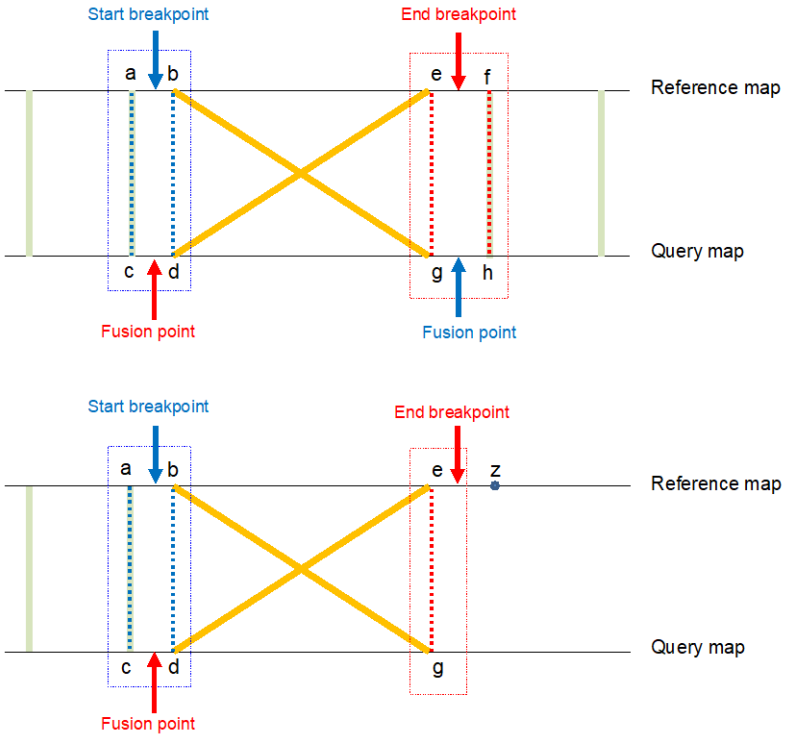



**Figure 32**. Signatures of inversions.

A) Signature of a fully described inversion:

The blue box contains the four labels described in the first `inversion_paired` line of the SMAP, and the red box contains the four labels described in the second `inversion_paired` line. The dotted lines represent the pairing of coordinates in the columns of the Smap text line. E.g., in this diagram a=RefStartPos, b=RefEndPos, c=QryStartPos, and d=QryEndPos, and similarly, e=RefStartPos, f=RefEndPos, g=QryStartPos, and h=QryEndPos. The coordinates at basepair resolution of the start (start breakpoint) and end (end breakpoint) of the inverted material are not observed, but they are bounded by the intervals (a,b) and (e,f). In the query map, the start of the inversion is fused to the downstream material (right of the blue fusion point), and the end breakpoint is fused to the upstream material (left of the red fusion point). The coordinates of the fusion points are bounded by (c,d) and (g,h), respectively.

B) Signature of a partially described inversion:

Where there is evidence of a single fusion event. The blue box contains the four labels appearing in the *inversion* line of the SMAP file, and the red box contains the two coordinates of the labels in the `inversion_partial` line. As described previously, the dotted lines represent the pairing of reference and query map coordinates. An end breakpoint in the interval (e,z) is fused with the material upstream of the red fusion point, bounded by (c,d). The red fusion point would align to a start breakpoint bounded by (a,b). The label z is the label of the reference map immediately after e. **NOTE**: The coordinate of label z is not provided in the SMAP file. **NOTE**: The situation where

the coordinates of the *inversion* line are greater than those in the `inversion_partial` line is also possible, and the interpretation is the same as in 2B.

Small inversions (whose inverted regions contained fewer than five labels; typically, less than 50 kbp) can be identified by searching in a limited space for two inversion signatures involving the same inverted alignment. They may be spanned by single genome maps, and there are eight coordinates (four on the query map and four on the reference map) involved in their descriptions. In the SMAP file, they are reported as two linked lines where the type is `inversion_paired`. They represent two inversion breakpoints that form the full inversion event. Paired inversions fully describe the material inverted and its bounds.

Neither the start nor the end breakpoints are observed at basepair resolution. However, the uncertainty in the breakpoint coordinates is bounded by the distances between the boundary labels. In **Figure 33A**, there is an inversion event where the genomic coordinate of the start is bounded by labels **a** and **b**, and the genomic coordinate of the end is bounded labels **e** and **f.** In the query map, the start of the inversion is fused to the material downstream the inversion, and the end of the inversion is fused to the upstream material.

In some cases (often for larger inversions), the signature for only one inversion breakpoint is found. These single inversions breakpoints are described by six label coordinates (**Figure 32B**). In the SMAP file, they are reported as two linked lines where the first one is of type *inversion* and the second one is of type `inversion_partial`. In the `inversion_partial` lines, some coordinate fields are not used (output as -1), hence the name "_partial".

In the scenario of **Figure 32B**, there is a fusion in an inverted orientation between a point bounded by labels **a** and **b,** and a point bounded by labels **e** and **z**, but there is no evidence of a reciprocal fusion or some other type of fusion.

Currently, inversions larger than 5 Mbp are called as intra-chromosomal translocation breakpoints. The 5-Mbp cutoff can be considered arbitrary (and may be adjusted on the command line). These large inversions are typically only partially spanned, so there is more uncertainty for the SV type assignment.

Inversion breakpoints report file

Apart from the SMAP file, we also provide an intermediate file with a different format specifying just the bounds of the inversion breakpoints in the reference map and the key fusion points (InversionBreakpoint1 and InversionBreakpoint2). The file is generated during confidence score calculations (available in Solve 3.6). Each inversion call is described with a single line and contains the following fields:

- `RefcontigID1`: The reference map (or chromosome) containing the inversion. It has the same value as the SMAP file.
- `QryContigID`: The query map used to call the inversion. It has the same value as the SMAP file.
- `InversionBreakpoint1`: The coordinate of the label in the reference map closest to the reference start breakpoint based on the alignment.
- `Bp1LowerBound`: Lower bound for InversionBreakpoint1
- `Bp1UpperBound`: Upper bound for InversionBreakpoint1
- `InversionBreakpoint2`: The coordinate of the label in the reference map closest to the reference end breakpoint based on the alignment.
- `Bp2LowerBound`: Lower bound for InversionBreakpoint2
- `Bp2UpperBound`: Upper bound for InversionBreakpoint2

- `Bp1SmapId`: The `SmapEntryID` of the SMAP line used to infer InversionBreakpoint1.
- `Bp2SmapId`: The `SmapEntryID` of the SMAP line used to infer InversionBreakpoint2.

The uncertainty for the bounds is based on a "closest label" criterion. The location of the true fusion point is within the label used to detect the inversion event and the next/previous label. Refer to **A** for a graphical interpretation. For all entries in the file, it is guaranteed that:

- InversionBreakpoint1 < InversionBreakpoint2
- Bp1LowerBound <= InversionBreakpoint1 <= Bp1UpperBound
- Bp2LowerBound <= InversionBreakpoint2 <= Bp2UpperBound

**There is no guarantee that `Bp1SmapId < Bp2SmapId`.**

### Inversion Confidence Score

The confidence score aims to give the user a measure of the correctness of an inversion call, assigning a high value to the score for calls that are true positives, i.e., events that are present in the sample, and low value of the score for false positives, i.e., calls that are made by a pipeline that are not present in the sample.

### FAQs

1. Is there a way to get information about the orientation of the inversion breakpoint call? Could I tell whether I am looking at the left or right breakpoint?

   The **inversion_partial** coordinates in the Smap refer to the inverted matchgroup. This determination is made by the rule that the matchgroup lines should not cross. See **Figure 33**.

2. Why is the output in the SMAP file different from what I see in the SV table in Access?

   Within Bionano Access, there is a script to "flatten" the two-line inversion entries into single-line entries for Access visualization. The entries for the other SV types are largely not modified. When an SMAP file is imported into Bionano Access, this script is run, and the resulting output is used for visualization. Any filtering applied in Access is then based on this flattened output instead of on the SMAP explicitly.

   The `"SuperType"` and `"Mask"` columns are added to handle SV calls with additional annotations and for downstream filtering. Insertion and deletion calls may be output as `"_nbase,"` if they overlap with N-base gaps; their `SuperType` would be “insertion” and “deletion,” respectively. The `SuperType` is coded in numeric values:

**Table 24**. SV Type and SuperType Values

| SV type | Supertype value |
|---|---|
| **Insertion** | 1 |
| **Deletion** | 2 |
| **Inversion breakpoint** | 3 |
| **Translocation breakpoint** | 4 |
| **Duplication** | 5 |
| **Catch all** | -1 |

Depending on their types, the SV calls are classified as masked or not masked. For example, translocation breakpoint calls that overlap with annotated segmental duplication regions are output as `"_segue"` calls. These are considered masked calls. This allows Access to interactively filter out masked calls in the interface.

The script separates the SMAP entries based on the SV types. For all "inversion" types, it saves them in a hash table (base data structure in Perl) because they need to be flattened separately. After building the hash table, the script checks that the number of entries in the table is an even number (assuming that all inversion breakpoints are represented in paired entries). The flattened output has six columns for inversion visualization specifically: `"BrkptEdges_Start_Ref_1", "BrkptEdges_Start_Ref_1_Pos", "BrkptEdges_Start_Ref_2", "BrkptEdges_Start_Ref_2_Pos", "BrkptEdges_End_Qry_1", and "BrkptEdges_End_Qry_2"`. The additional columns to be added contain information about where to draw the “hourglass,” or the inverter alignment that is characteristic of an inversion. For example, in **Figure 33**, **a** and **e** are reported to be the inversion boundaries.

The additional columns from the Variant Annotation Pipeline are carried over. However, for inversion entries, there is some loss in information during the flattening process. The flattening script only retains annotations from the first of the paired inversion entries.

3. How is inversion size estimated?

For `inversion_paired` calls:

In this case, both breakpoints are bounded, and the inversion size is estimated as the reference inverted alignment size plus 50% of the gaps on both sizes (**Figure 33**). For small inversions, the gaps are often a significant fraction of the size. Taking into the gaps improves the accuracy of the size estimate.

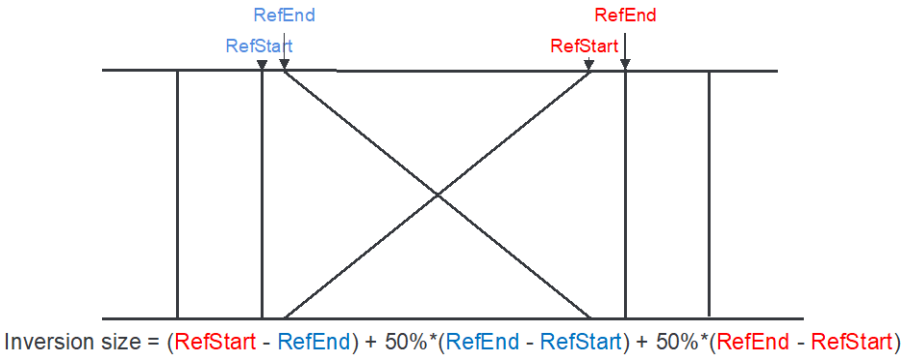



**Figure 33**. Estimation of inversion size on paired inversion calls.

For inversion/`inversion_partial` calls without overlap of the alignments on the reference:

The inversion size estimate is simply the size of the inverted alignment (**Figure 34**). This represents the lower bound estimate of the inversion size. Because the map does not fully span the inversion, we do not have full information about the entire inversion events. The inverted sequence may in fact be larger than the inverted alignment.

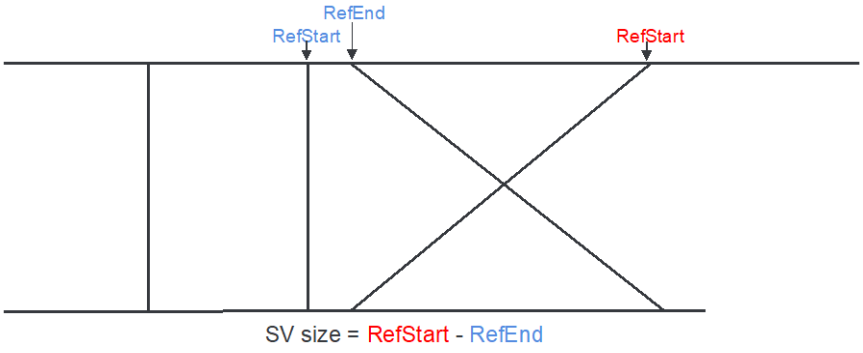



**Figure 34**. Estimation of inversion size on inversion/`inversion_partial` calls.

For inversion/`inversion_partial` calls with significant overlap of the alignments on the reference:

The overlap is considered, and the inversion size is estimated differently than described in the previous section. Instead of taking the size of the full inverted alignment, it is “trimmed” to where the straight, non-hourglass alignment starts **(Figure 35**).

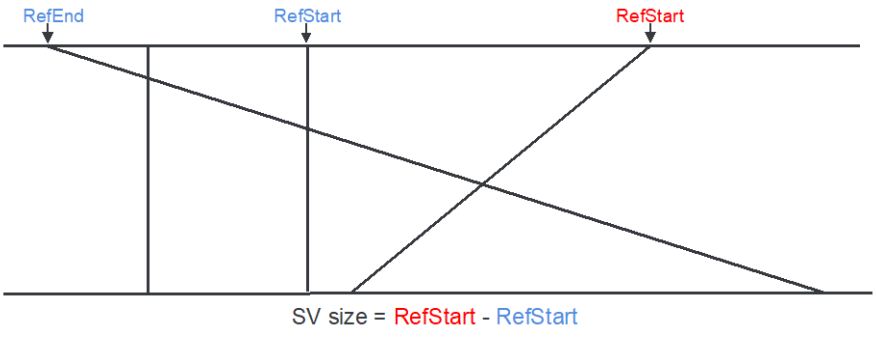



**Figure 35**. Estimation of inversion size on inversion/`inversion_partial` calls with alignment overlap.

# Appendix F: Confidence Modeling for Inversion and Translocation Breakpoints

## Introduction

The confidence scores aim to give the user a measure of how likely a structural variant (SV) call is correct, assigning high scores for calls that are likely true positives (TP), i.e., events that are present in the sample, and low scores for likely false positives (FP), i.e., calls that are made by a pipeline that are not present in the sample. Here we describe the algorithms used to produce the confidence scores for translocation and inversion breakpoints, together with their performance both in simulated and real datasets. The new confidence models were introduced in the Solve 3.6 release and updated for the Solve 3.7 release.

## Datasets

For developing the models, we considered as source of truth real samples processed by the Bionano pipelines, either the *de novo* assembly pipeline or the Rare Variant Analysis pipeline (RVA), having orthogonal datasets of SV calls made with other technologies, as well as ten simulated genomes containing SVs. For the real samples, we used the proband data from the Ashkenazi trio (NA24385) from the integrative benchmark from Genome in a Bottle (GIAB; Zook et al., 2019), five cancers samples (Dixon et al., 2018), seven samples sequenced with Pacific Biosciences (Audano et al., 2019), nine internal samples with manually curated translocations, three samples from the integrative analysis from 1000 Genomes (Chaisson et al., 2019), SK-BR-3 (a breast cancer cell line) with orthogonal data from Pacific Biosciences, and 2 colorectal cancer samples with orthogonal data from Oxford Nanopore.

The datasets used to validate the models consisted of seventeen runs of both the assembly and RVA pipelines containing mixtures of simulated SVs together with the hg19 human reference genome and twenty-one samples with confirmed SV events detected by cytogenetic analysis.

## Models for the Scores

### MODEL FOR THE TRANSLOCATION BREAKPOINT SCORES

The translocation scoring model was retrained in Bionano Solve 3.7. The input dataset contained 7,914 translocation calls, of which 3,004 were confirmed as TP by the orthogonal datasets or manual curation, and 4,910 were FP calls. The features were derived from maps alignments, reference coverage, and genomic location. In particular:

- The SV type: inter- or intrachromosomal translocation breakpoints
- The confidence score assigned by `RefAligner` (Shelton et al., 2015) to each of the 2 alignments used for calling the translocation breakpoints.
- The values of label coverage and occurrence from the query map in the region used to call the translocation breakpoints (±30 kbp buffer). These are proportional to the number molecules aligning to a query map. These values are divided by the global effective sequence coverage to make them relative and thus independent of the input coverage depth that may be different for different datasets.

- The average values of `ChimQuality`, `SegDupL`, `SegDupR`, `FragileL`, and `FragileR` in the consensus map region (±30 kbp), as described in CMAP section of the *OGM File Format Specification Sheet* (CG-30039).
- The genomic position of the translocation breakpoints bucketed into regions of 1 Mbp.
- Boolean variables indicate if the translocation breakpoints are in the centromeric or segmental duplication regions. The segmental duplication regions used during training have been updated.

The full dataset was randomly split into training (60%), validation (12%), and test (28%) datasets. The algorithm employed was Gradient Boosted Trees as implemented in LightGBM (Ke et al., 2017). After assessing model convergence and ability to improve its performance (**Figure 36**), it was optimized for the depth of the tree representation, the learning rate, and the number of node leaves using the package Optuna (Akiba et al., 2019) with the early-stopping convergence rule applied to the validation dataset. The area under the curve (AUC) measure on the receiver operator curve (ROC) was 0.991 (**Figure 37A**). Using the TPR/FPR plots (**Figure 37B**) we selected individual thresholds of 0.02 for interchromosomal translocations and 0.02 for intrachromosomal translocations (**NOTE:** These new thresholds for Bionano Solve 3.8 are different from those used in previous versions). Users may select a distinct set of thresholds depending on the project goals.

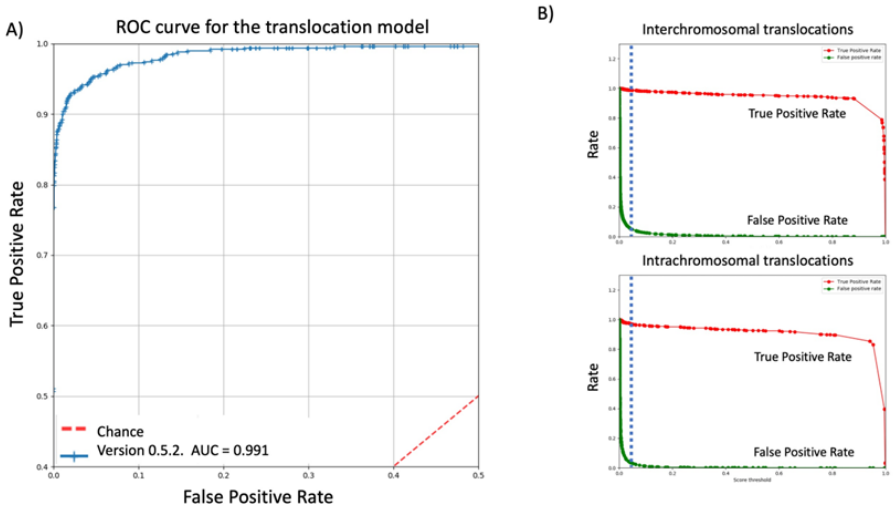



**Figure 36**. Learning curve of the model for the translocation score, built with the leave-one-out cross-validation schema (5 partitions). The x-axis shows the number of examples used for each of the models produced, and the y-axis is the f1-

score(harmonic mean between sensitivity and PPV). The width of the shaded region shows the standard deviation of the f1-score for each of the cross-validation partitions.

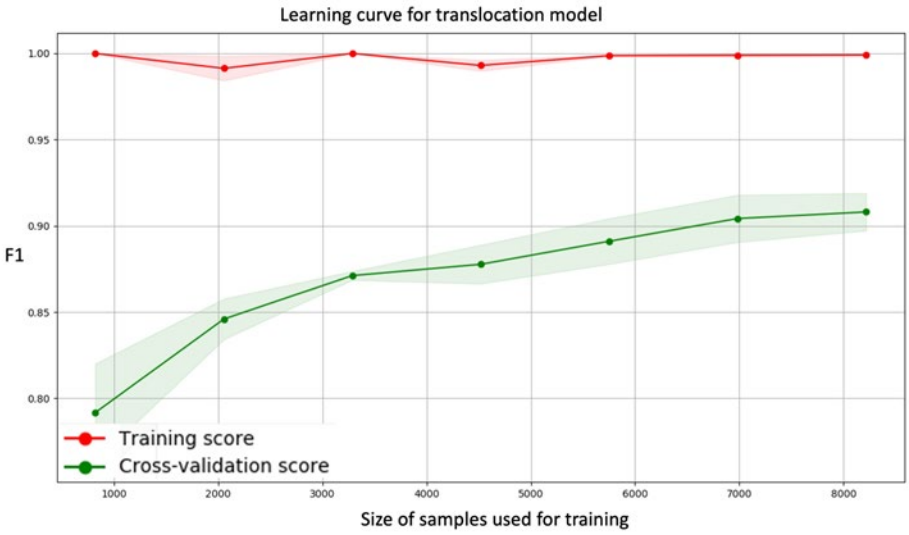



**Figure 37**. **A**) Receiver operator curve (ROC) for the translocation score model for its human version. The x-axis is the False Positive Rate (FPR) and the y-axis is the True Positive Rate (TPR). **B**) Plots of the TPR and FPR for the human model as functions of the threshold used for classification. The dashed line indicates the selected threshold.

We also built a non-human version of the confidence score model that does not consider the human-specific features. Namely, the genomic position and the location in the centromeric/segmental duplication regions are not considered. This model shows an AUC=0.939.

### MODEL FOR INVERSION BREAKPOINT SCORES

To build the confidence score model for inversion breakpoints, each of the breakpoints of an inversion is considered separately, and thus each inversion call contributes two data points to the model. We used 27,699 inversion breakpoints, 21,194 TP and 6,505 FP. The features considered for each breakpoint were:

- Minimum of the `RefAligner` confidences for the two alignments used to call the inversion.
- Length of the shortest alignment.
- Number of labels aligned in the inverted alignment.
- Distance between the lower and upper bounds for the position of the breakpoint.
- Number of labels in the query map of the inverted alignment that are in complex multi-path regions (CMPRs). RVA maps are not expected to have CMPR annotations.
- Number of upstream/downstream labels in the reference that are unaligned. If the breakpoint is the start of the inversion, this number is calculated by counting unaligned labels upstream of the position. For the end breakpoint, the number of unaligned labels is counted downstream of the position.
- Mean of the outlier fraction (`OutlierFrac` column of the CMAP file) for the region of the query map contained in the inverted alignment (±30 kbp buffer).

- Boolean variables indicating if the two alignments overlap in the reference map or overlap in the query map. An overlap in the alignments for the query map is a strong indication of a problematic call, as it implies that the query map aligns to the reference in both directions or that there is ambiguity in the alignment.
- The values of label coverage and occurrence from the query map in the inverted region, normalized by the effective sequence coverage. This feature is calculated the same way it is calculated for translocations.
- Boolean variables indicating if the breakpoint is in the centromeric or segmental duplication regions (the list of annotated regions was downloaded from the UCSC Genome Browser).

We again split the full dataset into training, validation, and test datasets, with the same percentages, and used `LightGBM` and `Optuna`. Due to the imbalanced nature of the available dataset (77% TP, 23% FP), we added a pre-processing step of random oversampling with replacement for the minority class (FP) during each model training. Each time a model was optimized in search of the best hyperparameters, the training dataset contained the same data points for the TP calls, but different datasets of FP calls. The learning curve shows that the model cannot learn more with the data at hand (**Figure 38**). Compared to the model for translocations, the model for inversions obtains better performance metrics (**Table 23**) and increased area under the ROC curve (**Figure 39A**), which allows to set a classification threshold with extremely high TPR and low FPR. We selected as such threshold 0.7 (**Figure 39B**).

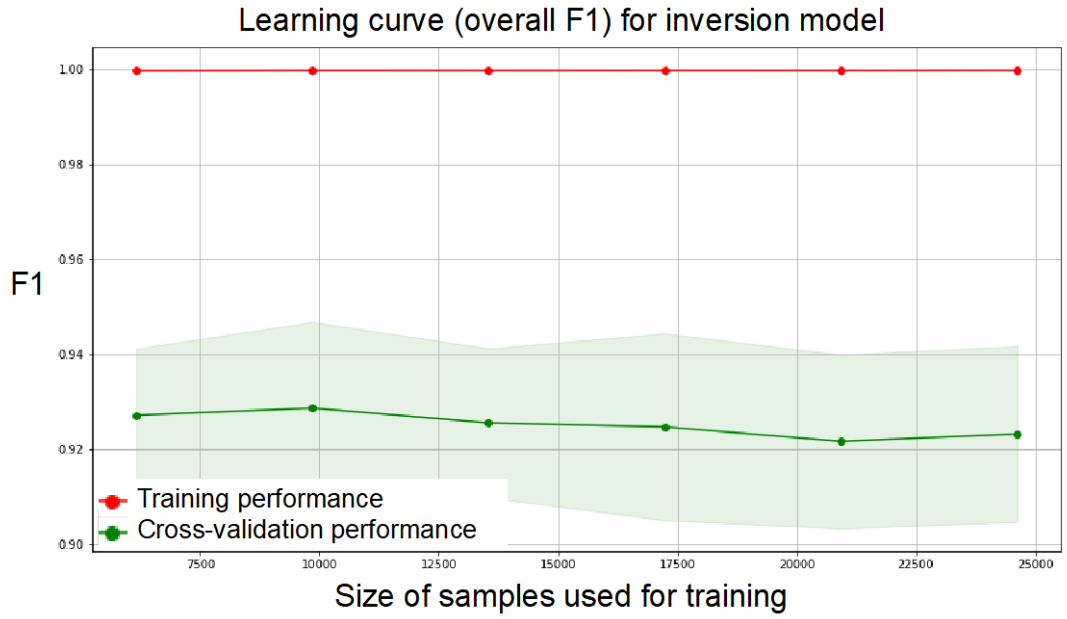



**Figure 38**. Learning curve of the model for the inversion score, built with the leave-one-out cross-validation schema (5 partitions). The x-axis shows the number of examples used for each of the models produced, and the y-axis shows the f1-score. The width of the shaded region shows the standard deviation of the f1-score for each of the cross-validation partitions.

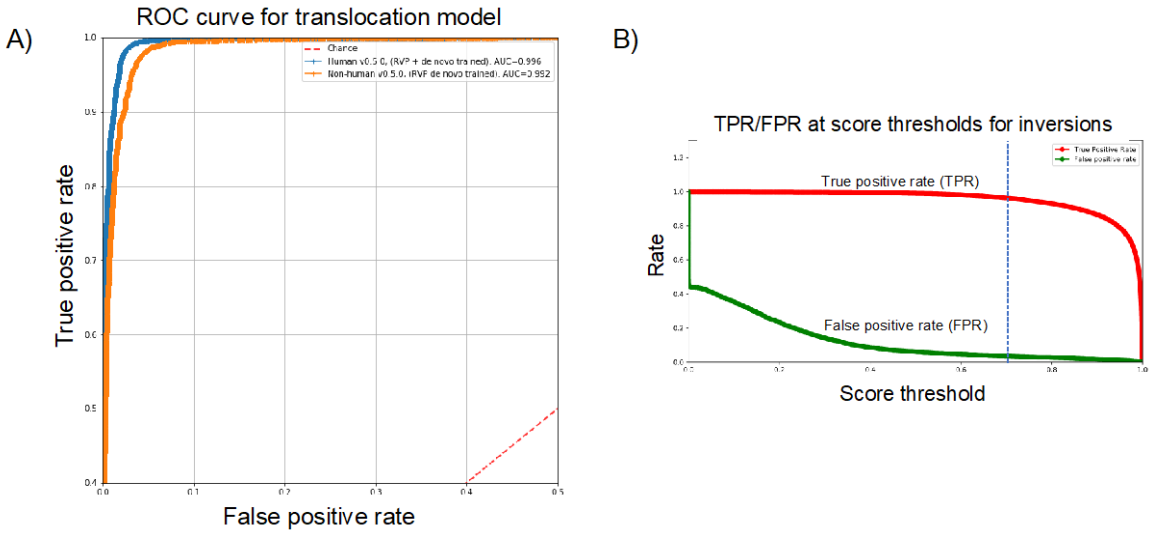



**Figure 39**. A) Receiver operator curves (ROC) for the inversion score model, both for its human and non-human versions. The x-axis shows the FPR and the y-axis shows TPR. B) Plot of the TPR and FPR for the human model as functions of the threshold used for classification. The dashed line indicates the selected threshold.

**Table 25**. Performance of the human version of the confidence score model for inversions.

| Dataset | Data points | Threshold | Percentage | Sensitivity | PPV | Accuracy |
|---|---|---|---|---|---|---|
| **Training** | 16619 | 0.70 | 60% | 100% | 100% | 100% |
| **Test** | 7756 | 0.70 | 28% | 95% | 97% | 94% |

A full inversion call is scored as the maximum of the individual scores of its two breakpoints.

### Validation

We validated our models with two independent datasets not used during model building: A) Mixtures of simulated genomes based on introducing SVs into the hg19 human reference, and B) Real samples with translocations and inversions confirmed by cytogenetic analyses.

### Simulated Datasets

This dataset consisted of two simulated genomes, one containing only translocations and the other containing only inversions. We first ran each genome at 100% allele fraction through both the *de novo* and RVA pipelines, and subsequently mixed with the human reference hg19 at 5%, 10%, 20%, and 30% allele fraction. In total, we used the data from seventeen pipeline runs. The overall performance for each of the SV types (**Table 26**) is like those obtained for the test split during model generation, with an improvement in the PPV of intrachromosomal fusions.

**Table 26**. Performance of the scoring models for seventeen runs of the pipeline on simulated datasets.

| Dataset | Data points | Threshold | Sensitivity | PPV | Accuracy |
|---|---|---|---|---|---|
| **Interchromosomal translocations** | 8,997 | 0.65 | 96% | 98% | 95% |
| **Intrachromosomal fusions** | 1,128 | 0.70 | 99% | 92% | 91% |
| **Inversions** | 22,062 | 0.70 | 94% | 92% | 91% |

## Cytogenetics Datasets

To validate the concordance between calls with high scores and those confirmed by cytogenetic analysis, we ran the scoring algorithms on a dataset composed of twenty-one samples containing twenty-five curated translocation (19) and inversion (6) variants and searched for Bionano calls consistent with those events. A Bionano call describing a certain event is considered consistent with the expected cytogenetic variant if it is of the same variant type, affects the same chromosomes, and it is in the same cytoband described in the cytogenetics report (±10 Mbp buffer). Multiple Bionano calls may be consistent with a given cytogenetic variant. When plotting the maximum confidence score of the calls consistent for each event, there is always at least one call with a score above the recommended thresholds (**Figure 40**). All the cytogenetic events would have been found by looking only at the Bionano calls of high confidence.

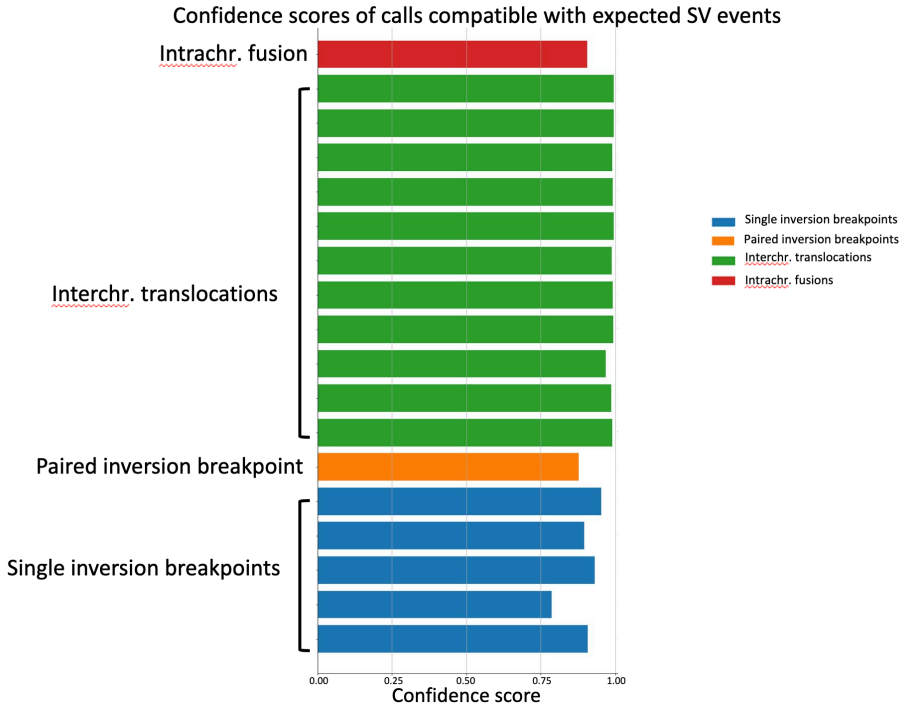



**Figure 40**. Translocation and inversion confidence scores for the Bionano calls consistent with curated cytogenetic variants. Left y-axis: Cytogenetic variant, one per row. x-axis: Maximum value of the confidence score for all consistent Bionano calls.

## References

Akiba, T., Sano, S., Yanase, T., Ohta, T., and Koyama, M. (2019). Optuna: A Next-generation Hyperparameter Optimization Framework. In Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, (Anchorage AK USA: ACM), pp. 2623–2631.

Audano, P.A., Sulovari, A., Graves-Lindsay, T.A., Cantsilieris, S., Sorensen, M., Welch, A.E., Dougherty, M.L., Nelson, B.J., Shah, A., Dutcher, S.K., et al. (2019). Characterizing the Major Structural Variant Alleles of the Human Genome. Cell *176*, 663-675.e19.

Bionano Genomcis (2020). CMAP File Format Specification Sheet.

Chaisson, M.J.P., Sanders, A.D., Zhao, X., Malhotra, A., Porubsky, D., Rausch, T., Gardner, E.J., Rodriguez, O.L., Guo, L., Collins, R.L., et al. (2019). Multi-platform discovery of haplotype-resolved structural variation in human genomes. Nat. Commun. *10*, 1784.

Chen, T., and Guestrin, C. (2016). XGBoost: A Scalable Tree Boosting System. In Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining - KDD '16, (San Francisco, California, USA: ACM Press), pp. 785–794.

Dixon, J.R., Xu, J., Dileep, V., Zhan, Y., Song, F., Le, V.T., Yardımcı, G.G., Chakraborty, A., Bann, D.V., Wang, Y., et al. (2018). Integrative detection and analysis of structural variation in cancer genomes. Nat. Genet. *50*, 1388–1398.

Ke, G., Meng, Q., Finely, T., and Wang, T. (2017). LightGBM: A Highly Efficient Gradient Boosting Decision Tree. Adv. Neural Inf. Process. Syst.

Shelton, J.M., Coleman, M.C., Herndon, N., Lu, N., Lam, E.T., Anantharaman, T., Sheth, P., and Brown, S.J. (2015). Tools and pipelines for BioNano data: molecule assembly pipeline and FASTA super scaffolding tool. BMC Genomics *16*, 734.

Zook, J.M., Hansen, N.F., Olson, N.D., Chapman, L.M., Mullikin, J.C., Xiao, C., Sherry, S., Koren, S., Phillippy, A.M., Boutros, P.C., et al. (2019). A robust benchmark for germline structural variant detection (Genomics).

# Appendix G: VCF Conversion

A standalone VCF conversion script is provided for converting Bionano SV and CNV calls into VCF format (version 4.2). The Python script typically runs automatically during the *de novo* assembly pipeline and Rare Variant Pipeline, but it can be run as a standalone conversion tool on the command line. The script requires Python 3.7 and Python libraries including pandas and numpy. Additionally, the PYTHONPATH environment variable must include the "VariantAnnotation/1.0" subdirectory of the Solve installation.

The minimum required input is the SMAP file (via the "-s" parameter), which contains SV calls that the pipelines generate. Based on the SMAP input, the script will look for the corresponding alignment files (`.xmap` and `_r.cmap`), if they are not explicitly provided as input. The VCF conversion script also optionally takes in CNV calls and converts them. If a required input is not found, the script will exit with an error message.

There are also several optional parameters:

- `-b`: path to cytoband database to define chromosomal coordinates
- `-r`: path to reference CMAP which is referred to in the SMAP supplied as "-s." It defaults to `_r.cmap in SMAP` directory.
- `-x`: path to alignment xmap which is referred to in the SMAP supplied as "-s." It defaults to `.xmap in SMAP` directory.
- `-n`: sample name for genotype data column. It defaults to `"Sample1"`.
- `-o`: prefix for the output VCF. It defaults to having the same prefix as the input SMAP.
- `-a`: RefSeq assembly accession version. It defaults to `"GCA_000001405.1"`, which corresponds to Genome Reference Consortium Human Build 37 (GRCh37).
- `-i`: dbVar-required experiment ID. It defaults to 1.
- `-m`: whether to filter out masked SVs (typically with additional underscored annotations such as "`_nbase`"). Use 0 to output all calls and 1 to filter out masked calls. It defaults to 1, where masked calls are filtered out.
- `-c`: path to CNV output file (typically named "`cnv_calls_exp.txt`" from the copy number analysis pipeline).
- -C: path to SV clusters file (typically named "**`exp_cluster_molecule_variant.txt`**")
- `-u`: whether to estimate breakpoint uncertainty [True/False]. Default = True.
- `-cu`: Uncertainty value used to correct breakpoint uncertainty for start and end positions. Default is 30kb
- `--species_reference`: Combined species and genome build information [`human_hg38 / human_hg19 / human_t2t-chm13-v2.0 / mouse_mm10 / mouse_mm39 / other`]. If provided with `-chr_cnv_stat`, zygosities on sex chromosomes are adjusted to be hemizygous as needed.
- `--chr_cnv_stat`: Path to cnv_chr_stats.txt file produced by fractional CNV caller.
- `--skip_confidence_filter`: Do not apply recommended confidence filters and list all variants as PASS [True/False]. Default = False
- `--keep_duplicates`: Keep duplicate SMAP entries in file without filtering. [True/False]. Default = False

The VCF converter can be run stand-alone using the command:

```
PYTHONPATH=$SOLVE/VariantAnnotation/1.0 \

   python3 $SOLVE/bionano_packages/VCFConverter/src/bionano_vcf_converter.py
```

where $SOLVE is the installation path for the Bionano Solve software. When Bionano pipelines automatically run the converter, they produce files ending with .ogm.vcf. These can be found inside the pipeline_results.zip produced by the pipelines. The header lists the command that the pipeline used to produce the given VCF.

The VCF converter is based on the variant calls in the input SMAP/CNV data; however, several modifications are applied to coordinates during the conversion, such that the values in the VCF may appear to be different than those in the SMAP. These modifications are discussed below.

### How are coordinates in the VCF computed?

This depends on the SV type. *For insertions and deletions*, the coordinates in the VCF will appear different from the coordinates in the SMAP. In the VCF, we try to indicate the uncertainty in the breakpoints. Consensus maps are aligned to reference and “outlier” regions are output as structural variants.

We center the variant in the region defined by SMAP RefStartPos and RefEndPos and make the uncertainty values the distance from the original label position in the SMAP to the newly centered variant. This harmonizes various position calculations so that:

For deletions:

POS - CIPOS = RefStartPos

END + CIEND = RefStopPos - 1

SVLEN = END - POS (which matches deletion length in the SMAP)

For insertions:

POS - CIPOS = RefStartPos

POS + CIPOS = RefStopPos

END = POS

CIEND = CIPOS

SVLEN = insertion length in the SMAP

The SMAP provides accurate size information for the net gain or loss of DNA between matches, where a match is one assembled map label aligned to one reference label. Due to the limits of OGM resolution, one cannot determine the exact placement of any insertions/deletions causing the gain/loss or if there are multiple insertions/deletions between the two consecutive matches. The VCF converter assumes only a single insertion or deletion. A deletion will be represented in VCF format with the attributes shown in **Figure 41**.

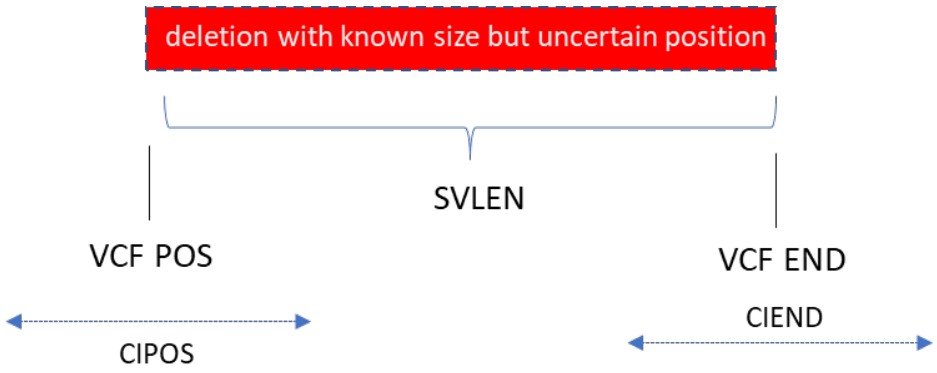



**Figure 41**. VCF representation of a deletion.

Assuming a deletion, such as shown in **Figure 42,** exists between two matches, the true start (POS) could be as far upstream as the left match's reference label. In that scenario, the end (END) could only be far enough upstream to be consistent with the known SVLEN (See the top interval labeled "deletion length" in **Figure 42**). This determines the most upstream CIPOS adjusted coordinate and the most upstream CIEND adjusted coordinate. The same logic is applied, considering the deletion may be as far downstream as to when END is placed at the right match's reference label (the bottom interval labeled "deletion length" in **Figure 42**).

These coordinates do not take into consideration unmatched labels between matches (shown in green in **Figure 42**).

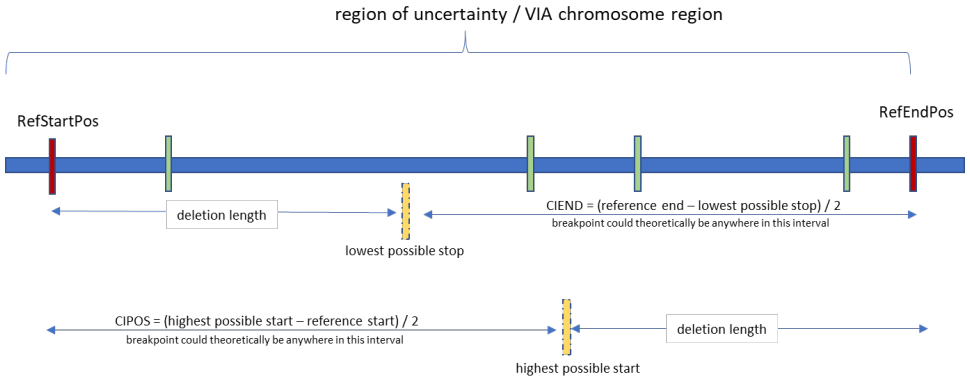



**Figure 42**. Constraints on the placement of a deletion of known size given known length.

For insertion, where length is unrelated to the reference, POS is centered and the full distance between labels is used as the CI.

For translocation and inversion, and duplication breakpoints, the procedure is different. The conversion script looks for the next reference label to the left and to the right for each breakpoint. The assumption is that because

these SV regions tend to be complex, the alignment boundaries may be off by one label. It is equally likely that we over- or under-align by one label. The breakpoint uncertainty is thus the average between the left interval and the right interval. This tends to provide conservative (upper bound) estimates of the breakpoint uncertainty.

For *CNV*s the script simply uses the coordinates from the CNV output without modification, and breakpoint uncertainty is set to 30 kb based on empirical data. Generally, the breakpoint uncertainty for CNV calls is higher than for SV calls.

### How is size computed (SVLEN)?

For insertion and deletion calls, the size SVLEN is defined to be the difference between the reference and the query map intervals in the outlier region. For inversion breakpoints SVLEN is 0 and any gain/loss of DNA flanking the inverted region will be accounted for by separate insertions/deletion VCF entries. For duplications, the SVLEN is defined to be the difference between the reference start and end coordinates of the duplicated regions. For translocation breakpoints, the SVLEN outputs as 0.

### How is orientation for translocation breakpoints computed?

For more recent versions of the SMAP input, the script expects and will use the “orientation” column for encoding translocation breakpoint continuation direction. If this column is not present, the VCF converter will attempt to compute the directionality by searching for the SMAP breakend at either end of the corresponding XMAP alignment. While more informative than the orientation column method (see below), this fallback method only works when the XMAP data lines linked from an SMAP data line were able to be trimmed for that single SMAP data line. For complex maps, there may be multiple SVs, limiting the amount of trimming that can be done on the corresponding XMAPs. In these cases, the directionality cannot be determined by this method and the converter will abort rather than give inconsistent results.

When the orientation column is present, for each side of a translocation breakpoint, the script will compute a VCF continuation direction in the ALT field based on the alignment orientation (either forward or reverse related to the reference as shown in **Figure 2**). The orientation can be “+” or “-“. The final orientation is the combination of the orientation of each side. There are four possible orientations: “+/+,” “+/-“, “-/+,” and “+/+.” However, the SMAP data does not fully characterize the SV, as the orientation column is encoded based on the map while the semantics of the VCF specification are defined in terms of the reference. The SMAP does not currently encode the relationship between the aligning segments on the map and the aligning segments on the reference. Therefore, when the SMAP orientation column is present, directionality is encoded assuming that the upstream side of the map always corresponds to RefContigID1 and RefStartPos columns in the SMAP. When needed, the complete orientation and continuation direction for translocations can be visualized in Bionano Access.

### How is confidence computed?

Confidence in the VCF file is computed as the Phred score of the SMAP column “Confidence”: `-10*log10(1-` confidence in the SMAP). The values are artificially capped at 20 (given that we have limited resolution for the confidence assignment beyond that). When the SMAP confidence score is -1 (e.g., for duplications) the value in the VCF file is `".”`.

### How is the genotyping (the GT field) computed?

GT is output depending on the zygosity annotation in the SMAP. GT would be "1/1" if the zygosity is "homozygous", "0/1" if the zygosity is "heterozygous", "1" if the zygosity is "hemizygous" and "./." if the zygosity is "unknown" except for deletions, where it will be "0/1". For previous versions of the SMAP file, the zygosity column may be absent; if the column is not found, GT would be output as "./..."

### Why are SV counts in RVA or *de novo* assembly report different from VCF?

The RVA or *de novo* assembly report includes SV counts before and after clustering and collapsing similar SV calls. The pipelines provide the optional SV cluster set (-C argument) to VCF conversion which it uses to remove potentially redundant SV calls; It does this by including in the VCF only the highest confidence variant from each cluster for indels, inversions, and duplications. If run without cluster data, every SV in the SMAP will be processed. Users may also notice that the number of entries is different. Inversion breakpoints are represented by two lines in the SMAP, but as a single line in the VCF while intra-chromosomal fusion and inter-chromosomal translocations will be represented as two related breakend entries in the VCF while being represented by a single line in the SMAP.

### What filters are applied?

In the **Filter** column, "Masked" is output for calls with SV types that include "masked," "segdupe," "common," or "nbase," except for "inversion_nbase." "LowConfidence" is output for variants that do not meet the minimum recommended confidence score for that variant type. "PoorMoleculeSupport" is output for variants if there exist annotated values of "Found_in_self_molecules" being false or "Fail_assembly_chimeric_score" being true. "PASS" is output for all other calls.

Masking is performed by default during the SV and CNV calling steps using separate mask databases. The SV mask database includes regions where false positive translocation calls were made in control samples with no known translocations. These regions are often segmental duplication loci and cannot be aligned uniquely. The CNV mask database includes regions with elevated coverage noise, defined based on control samples with no known large CNV events. False positive CNV calls are more common in high coverage noise regions. SV and CNV calls overlapping with the mask database entries are marked as masked and of lower confidence.

### How are Bionano SV types mapped to VCF variant types?

**Table 27**. SV types mapped to VCF variant types

| Bionano SV type | VCF type |
|---|---|
| **Insertion** | **INS** |
| **Deletion** | **DEL** |

| Bionano SV type | VCF type |
|---|---|
| intrachr_fusion | BND |
| translocation_interchr | BND |
| inversion | BND |
| inversion_paired | INV |
| inversion_partial | BND |
| inversion_repeat | INV |
| duplication_split | DUP |
| duplication | DUP |
| duplication_inverted | DUP:INVERTED |

**NOTE**: For inversion_paired, the map spans the entire event and thus both breakpoints were detected and can be represented as the VCF INV type. For partially detected inversions (comprising inversion_partial and inversion SMAP entries), the calling map only spans one breakpoint. These are thus encoded with the BND type.

The SMAP format differentiates between inverted and non-inverted duplications. However, the VCF format's closest top level type describes DUP only as indicating increased copy number without indicating direction. The format does allows for user defined subtypes which the VCF converter uses to preserve this distinction. The VCF converter will represent SMAP duplication_inverted SVs with an "INVERTED" subtype("DUP:INVERTED"). Because the SMAP and VCF converter preserve this distinction, in Bionano VCFs, users may additionally interpret the "DUP" type to specifically be non-inverted duplications.

### How are AOH/LOH encoded in the VCF?

Individual VCF records will have the value in the FORMAT field for a sample if the variant falls in an AOH/LOH block. The start position of the AOH block will be used as an identifier unless it is not unique.

Below is an example showing two variants in the same block

```
chr1 1000 SMAP1 N <DEL> 20 PASS SVTYPE=DEL;END=1100 GT:ABK  0/1:1000

chr1 1200 SMAP2 N <INV>  20 PASS SVTYPE=INV;END=1300 GT:ABK. 1/1:1000
```

### How are inversion annotations aggregated?

Full inversions are represented in an SMAP as two data lines of type inversion_paired, each with their own annotation. The converted VCF will have a single entry for the SMAP inversion_paired type. Annotation values are aggregated to maximize sensitivity when filters are applied, such that if either inversion_paired entry in an SMAP has an annotation value that passes a filter, then the corresponding aggregated VCF entry should as well. Annotation values of a collection type, such as gene lists, will have their collections combined.

For the inversion/inversion_partial SMAP data line pair, the VCF entry will have the annotation from only the SMAP inversion data line.

# Appendix H: Variant Allele Fraction Calculation

### Method

The VAF calculation uses as input the SV calls in the SMAP file and the alignments in the XMAP file between all the consensus maps and the reference genome created by the pipelines. From the alignments we obtain the coverage for each of the consensus map segments, where a segment is the stretch between two labels. Each segment of a consensus map is supported by a certain number of aligned molecules (**Figure 44**). As multiple SV calls may refer to the same underlying SV detected using different consensus maps, to avoid incorrectly considering multiple equivalent alleles at the same locus, we first cluster the SV calls based on the distance between their breakpoints and their SV type (Bionano Genomics proprietary clustering algorithm). The alignments for the consensus maps of the clustered calls are considered together to calculate the VAF of an SV, with the map of the reference genome providing the common reference frame for aggregation. For example, in **Figure 44** there is a consensus map containing a deletion; if a second consensus map (not depicted in the figure) containing the same deletion would have been clustered together with it, the coverage of both maps for each for the reference labels $l_1, l_2, \cdots, l_8$ would have been aggregated.

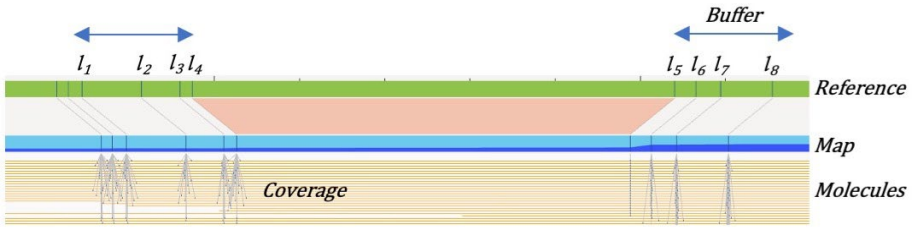



**Figure 43**. Concepts involved in the VAF calculation algorithm. A consensus map containing a deletion is aligned against the reference map, and the consensus map is created by multiple molecules. Each segment of the reference map may have different coverage on the consensus map, depending on the number of aligned molecules containing the segment. A buffer distance applied at both sides of the deletion breakpoints contains 8 labels, 4 before the start of the deletion, and 4 past the end. Those labels will be used to calculate the VAF.

After the aggregation step, the VAF $\propto_{\mathrm{k}}$ of a certain allele $k$ for an SV can be inferred by the differences in coverage of a small set of reference labels $\mathrm{D} = \{l_1, l_2, \ldots, l_L\}$ in the surroundings of the SV breakpoints. In a Bayesian framework, the probability of having a VAF $\propto_{\mathrm{k}}$ for the allele $k$ given the label set $\mathrm{D}$ and the genotype G can be expressed as:

$$\mathrm{P}(\propto_{\mathrm{k}} |\mathrm{D}, \mathrm{G}) = \frac{\mathrm{P}(\mathrm{D}| \propto_{\mathrm{k}}, \mathrm{G})\mathrm{P}(\propto_{\mathrm{k}} |\mathrm{G})}{\mathrm{P}(\mathrm{D}|\mathrm{G})} = \frac{\mathrm{P}(\mathrm{D}| \propto_{\mathrm{k}}, \mathrm{G})\mathrm{P}(\propto_{\mathrm{k}} |\mathrm{G})}{\int_0^1 \mathrm{P}(\mathrm{D}| \propto, \mathrm{G})\mathrm{P}(\propto |\mathrm{G})\mathrm{d} \propto}$$

Under the assumption that each label $l_i$ behaves independently, the probability for the observed coverages of the labels in the set *D* is:

$$\boldsymbol{P}(\boldsymbol{D}| \propto_{\boldsymbol{k}}, \boldsymbol{G}) = \prod_{\boldsymbol{i}=\boldsymbol{1}}^{\boldsymbol{L}} \boldsymbol{P}(\boldsymbol{l_i}| \propto_{\boldsymbol{k}}, \boldsymbol{G})$$

We model the probability for each individual reference label $l_i$ as a binomial distribution:

$$\boldsymbol{P}(\boldsymbol{l_i}| \propto_{\boldsymbol{k}}, \boldsymbol{G}) = \binom{\boldsymbol{M}}{\boldsymbol{M_k}} (\boldsymbol{1} - \boldsymbol{p_k})^{\boldsymbol{M}-\boldsymbol{M_k}} \boldsymbol{p_k^{M_k}}$$

Where M is the total coverage for the segment starting at label $\mathrm{l_i}$, $\mathrm{M_k}$ is the aggregated coverage assigned to allele $k$ after the clustering step, and $\mathrm{p_k}$ is the probability of observing a label from a molecule of the allele $k$ aligned to the reference label $\mathrm{l_i}$. If we consider $\mathrm{p_k}$ to be the same for all the labels in the set D, we have $\mathrm{p_k} = \propto_{\mathrm{k}}$. We find the VAF for the SV by finding the value of $\propto_{\mathrm{k}}$ that maximizes $\mathrm{P}(\propto_{\mathrm{k}} |\mathrm{D}, \mathrm{G})$.

For duplications calls it is not possible to apply Bayesian inference because the consensus maps may not always contain both whole copies of the duplicated material, making the determination of the correspondence between the labels of the reference, first, and second copies impossible. Additionally, there is only one region in the consensus map where the coverage may be different from the reference map: Namely, around the label of the reference aligned to the beginning of the duplicated material (

**Figure 45**).

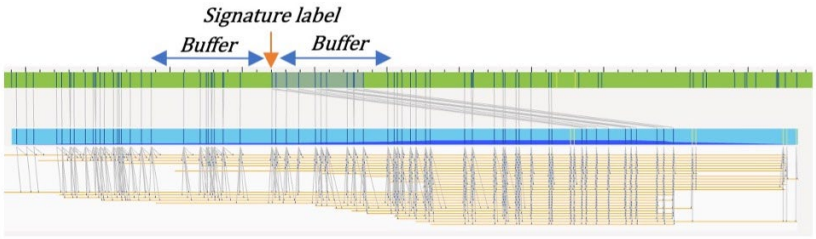



**Figure 44**. Alignment of a consensus map containing a duplication to a reference map. The reference label aligned to the beginning of the duplicated material (signature label) marks the center of the region used to calculate the VAF. The labels used are those contained insider a buffer distance.

We average the coverages over the region defined by the signature label (plus buffers) and apply the naïve approach of:

$$\propto_{k} = \frac{C_k}{C_R + C_k}$$

Where $C_R$ is the mean coverage in the region the for the reference map, and $C_K$ is the mean coverage around the critical breakpoint for the consensus maps from allele $k$.

### Performance on Simulated Data

We evaluated our method on 10 samples created by mixing simulated molecules containing SVs inserted randomly in the hg38 reference genome together with non-modified simulated molecules from hg38. We ran the pipelines for values of 80, 180 and 300X effective coverage, and 5, 10, 20, 40, 80, and 100% simulated VAF. For RVA we had 159/160 successful runs (**Table 28**) and for *de novo* assembly 129/130 (**Table 29**). The median error (difference between the simulated and calculated VAF) values were in the range 0-0.11 for all combinations of VAF and coverage, for both pipelines. The third quartile of the error was 0-0.20 for *de novo* assembly and 0-0.50 for RVA, indicating that the VAF values for the RVA pipeline are expected to have wider variability and have poorer performance for the most incorrect SVs. Although minimal, we observed highly incorrect VAF values (Max column) in all runs, due to multiple causes: false positive calls, SVs where the first clustering stage failed to group all equivalent SV calls together, or genomic regions with shallow label coverage that strongly influenced the Bayesian inference procedure.

When adopting as a success criterion that the difference between the simulated and calculated VAFs should be under 10%, we have found as a trend that the method behaves differently in different contexts. For RVA (

**)** the highest success rate is in the VAF range 5-20% (78-96%), with decreased performance afterwards (55-70%). For *de novo* (**Table 27**), the highest success range is at the 100% VAF range (94-96%), followed by the 5-40% VAF range (52-89%). The poorest performance is at 80% VAF, with a success rate 48-52%.

**Table 28**. Performance of the VAF calculation algorithm for RVA runs on 10 simulated samples at different VAF and effective coverage. Only variants of high confidence are considered for the SV counts. The columns for the Median, 3rd quartile, and Max describe the distribution of the difference between the simulated and calculated VAF. The last column shows the percentage of calls that had an error lower than 10% for the VAF.

| Effective Coverage | VAF simulated | Samples | SV count | Median | 3rd quartile | Max | Variants with VAF error <= 10% |
|---|---|---|---|---|---|---|---|
| **300** | 5% | 10 | 12271 | 0.02 | 0.03 | 0.95 | 96% |
| **300** | 10% | 10 | 16849 | 0.02 | 0.05 | 0.90 | 91% |
| **300** | 20% | 10 | 18190 | 0.03 | 0.07 | 0.80 | 84% |
| **300** | 40% | 10 | 21135 | 0.05 | 0.14 | 0.60 | 67% |
| **300** | 80% | 10 | 24366 | 0.07 | 0.22 | 0.80 | 55% |
| **300** | 100% | 10 | 25742 | 0.07 | 0.29 | 1.00 | 56% |
| **180** | 10% | 10 | 13176 | 0.03 | 0.05 | 0.90 | 95% |
| **180** | 20% | 10 | 16143 | 0.04 | 0.08 | 0.80 | 83% |
| **180** | 40% | 10 | 18486 | 0.06 | 0.13 | 0.60 | 67% |
| **180** | 80% | 10 | 19345 | 0.07 | 0.20 | 0.56 | 56% |
| **180** | 100% | 10 | 22533 | 0.06 | 0.24 | 1.00 | 62% |
| **80** | 10% | 9 | 6972 | 0.03 | 0.50 | 0.90 | 97% |
| **80** | 20% | 10 | 12519 | 0.06 | 0.10 | 0.80 | 78% |
| **80** | 40% | 10 | 15223 | 0.08 | 0.15 | 0.60 | 59% |
| **80** | 80% | 10 | 17521 | 0.07 | 0.19 | 0.80 | 59% |

| Effective Coverage | VAF simulated | Samples | SV count | Median | 3rd quartile | Max | Variants with VAF error <= 10% |
|---|---|---|---|---|---|---|---|
| **80** | 100% | 10 | 18374 | 0.04 | 0.15 | 1.00 | 70% |

**Table 29**. Performance of the VAF calculation algorithm for *de novo* assembly pipeline runs on ten simulated samples at different VAF and effective coverage. Only variants of high confidence are considered for the SV counts. The columns for the Median, 3rd quartile, and Max describe the distribution of the difference between the simulated and calculated VAF. The last column shows the percentage of calls that had an error lower than 10% for the VAF.

| Effective Coverage | VAF simulated | Samples | SV count | Median | 3rd quartile | Max | Variants with VAF error <= 10% |
|---|---|---|---|---|---|---|---|
| **300** | 5% | 5 | 705 | 0.08 | 0.14 | 0.95 | 52% |
| **300** | 10% | 5 | 2924 | 0.06 | 0.12 | 0.90 | 63% |
| **300** | 20% | 5 | 8415 | 0.09 | 0.15 | 0.80 | 54% |
| **300** | 40% | 5 | 14370 | 0.08 | 0.12 | 0.60 | 63% |
| **300** | 80% | 5 | 16473 | 0.09 | 0.18 | 0.80 | 48% |
| **300** | 100% | 5 | 23371 | 0.00 | 0.00 | 1.00 | 96% |
| **180** | 10% | 10 | 3557 | 0.05 | 0.08 | 0.90 | 76% |
| **180** | 20% | 10 | 15869 | 0.05 | 0.08 | 0.80 | 85% |
| **180** | 40% | 10 | 22424 | 0.05 | 0.10 | 0.60 | 72% |
| **180** | 80% | 10 | 36445 | 0.09 | 0.20 | 0.80 | 52% |
| **180** | 100% | 10 | 35573 | 0.00 | 0.00 | 0.99 | 94% |
| **80** | 10% | 10 | 1950 | 0.05 | 0.09 | 0.90 | 84% |
| **80** | 20% | 10 | 16104 | 0.04 | 0.07 | 0.80 | 89% |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| **80** | 40% | 10 | 25643 | 0.05 | 0.10 | 0.60 | 74% |
| **80** | 80% | 9 | 30398 | 0.11 | 0.20 | 0.77 | 49% |
| **80** | 100% | 10 | 44533 | 0.00 | 0.00 | 1.00 | 95% |

## Reproducibility

We tested the reproducibility of our method by running the VAF calculations on a triplicated experiment of a tumor/normal pair of samples, for both RVA and *de novo* assembly pipelines. For the RVA runs, it is apparent that there is correspondence between the values of the VAF for each variant across the replicates (**Figure 45**). An analysis of variance (**Table 30**) confirmed that the null hypothesis of the sample not being a factor of variation is plausible, both for the tumor replicates (p-value 0.29) and normal replicates (p-value 0.20). The same results were obtained for the *de novo* assembly pipeline (data not shown).

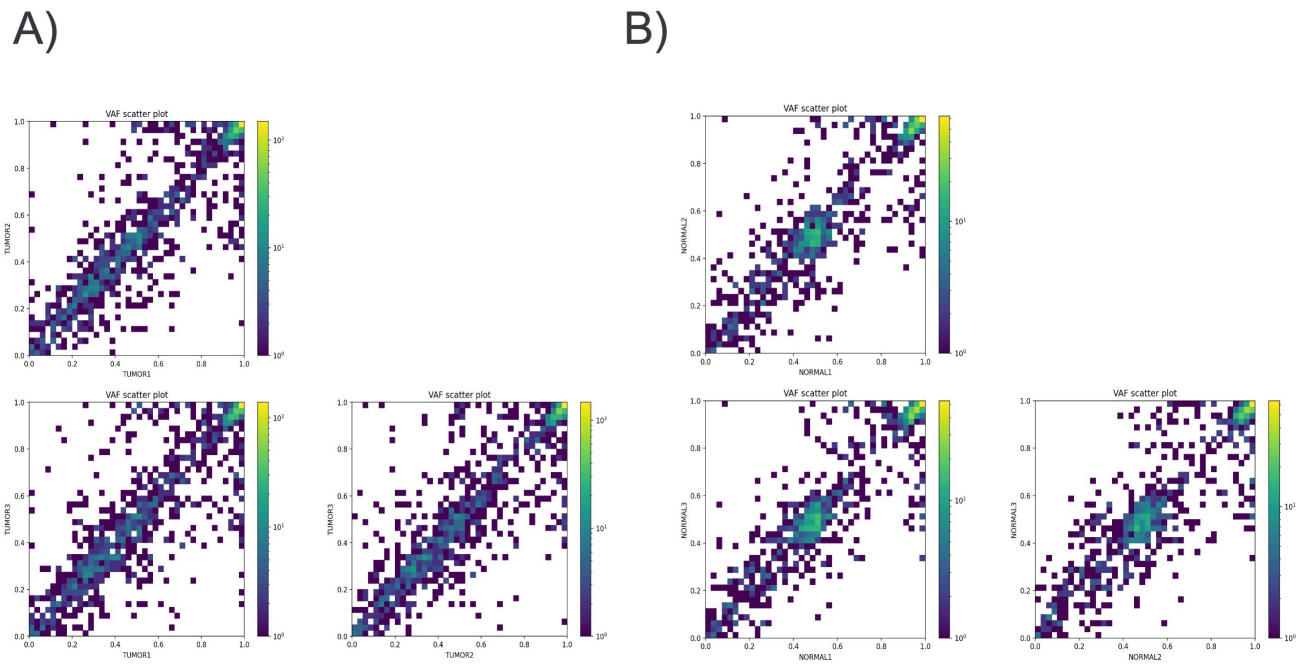



**Figure 45**. Pairwise visual comparison of the VAF values between replicate samples processed with RVA. For each plot, the x-axis contains the VAFs for the first sample replicate, the y-axis the VAFs for the second sample replicate, and each point represents an SV. The color scale indicates the number of SVs with a certain (x,y) value combination. The ideal perfect

concordance would appear as a diagonal in the plots. A) Comparison between the tumor replicates. B) Comparison between the normal replicates.

**Table 30**. Blocked-ANOVA tests for the analysis of the repeatability of VAF results from RVA. Left, grey: Analysis for the three tumor replicates. Right, white: Analysis for the three normal replicates.

| **Tumor samples** | | | | | **Normal samples** | | | | |
|---|---|---|---|---|---|---|---|---|---|
| **Effect** | Sum squares | Degrees of freedom | F statistic | p-value | Effect | Sum squares | Degrees of freedom | F statistic | p-value |
| **Variant** | 824.80 | 1285 | 60.56 | 0.00 | Variant | 552.02 | 956 | 68.58 | 0.00 |
| **Sample** | 0.03 | 2 | 1.23 | 0.29 | Sample | 0.03 | 2 | 1.60 | 0.20 |
| **Residual** | 27.18 | 2565 | | | Residual | 16.08 | 1890 | | |

## Performance on Experimental Data

It is difficult to obtain an experimental dataset where all the VAF values are known for all the SVs present in the sample. To overcome that difficulty and at the same offer results on the performance of the algorithm on experimental data, we created an in-silico dilution experiment where molecules from a tumor sample and its paired normal are mixed in different proportions, and we look at the VAF values for the somatic variants across the different mixes. We observed the expected progressive decrease in the VAF values (**Figure 46**), with values close to 100% when there are only tumor molecules in the mix and VAF distributions located around the simulated tumor purity.

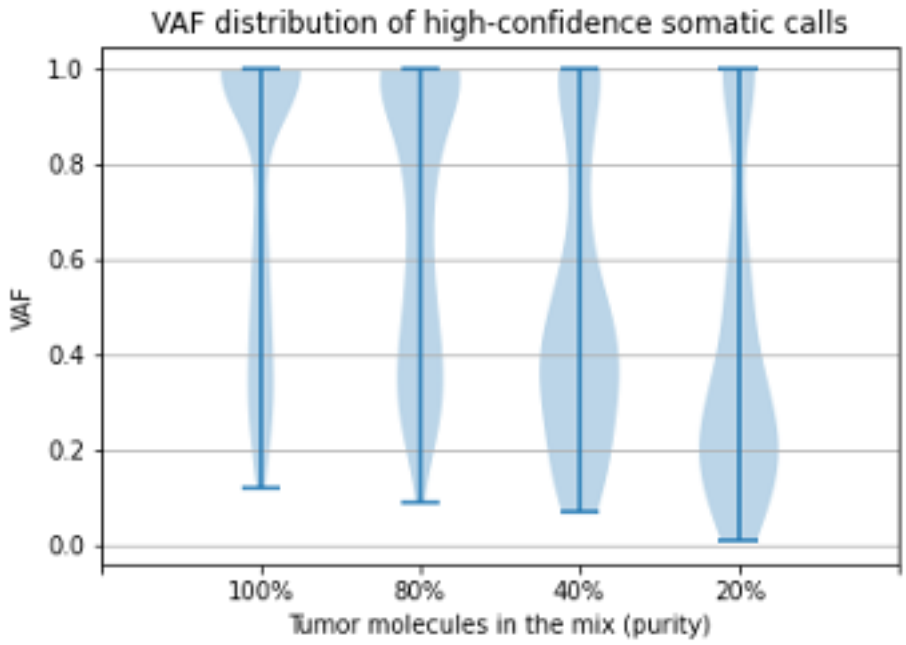



**Figure 46**. Violin plots with the histogram of VAF values for in-silico mixes of paired tumor/normal samples, at different tumor purity. Only somatic high-confidence SV calls are considered.

### Segmentation of the Whole Genome VAF Plot

The plot of VAF vs genomic coordinates shows patterns of variation of the VAF across the genome. For a normal healthy germline sample two main VAF bands are expected at VAFs 50% and 100% for heterozygous and homozygous variants, respectively. When there are changes in the chromosomic structure of the sample, such as aneuploidy events, it is possible to see stretches of VAF values (segments) with different VAF and changes in the number of bands. To help discover and analyze those events, we have developed an algorithm to calculate the boundaries and the median VAF of the segments of the autosomal chromosomes in the plot. The algorithm is as follows:

First, the non-informative homozygous SVs (default threshold VAF 97%) are not considered for segmentation. Next, the raw VAF values are transformed to the “major allele” VAF values by applying the transformation:

$$\mathrm{mVAF} = \begin{cases} \mathrm{VAF} & \text{if } \mathrm{VAF} > 0.5 \\ 1 - \mathrm{VAF} & \text{if } \mathrm{VAF} \leq 0.5 \end{cases}$$

We filter the mVAF outliers using triplet filtering *(Staaf et al., 2008)* with default threshold 0.40 and segment the mVAF plot using circular binary segmentation (CBS) *(Olshen et al., 2004)* on the non-centromeric areas. As a measure more robust to outliers, we consider as the mVAF of a segment the median of the mVAFs from the points spanned by it. Next, we compare the median mVAFs of consecutive segments using the Mann-Whitney test, and two segments are joined if the null hypothesis of having the same median is satisfied (p-value 0.0001). The procedure is repeated until no consecutive segments can be joined. The segment boundaries and median values obtained from the mVAF are transferred to the raw VAF plot. When the median VAF of a segment is below a certain threshold (default 0.56), we consider that the possible allelic imbalance present in the region has not been sufficiently resolved and recalculate the median VAF for the segment, this time using the VAF values and

not the mVAF. The result is a more accurate median value for the segments in normal, heterozygous parts of the genome. Finally, to avoid the creation of many small, noisy segments, or segments in regions with loss of heterozygosity, segments shorter than 10 Mbp, supported by less than 10 SVs, or with variant density lesser than 0.3 SV/Mbp are discarded.

**NOTE:** Currently, the default thresholds values employed by the segmentation algorithm are not customizable.

### References

Olshen,A.B. *et al.* (2004) Circular binary segmentation for the analysis of array-based DNA copy number data. *Biostatistics*, **5**, 557–572.

Staaf,J. *et al.* (2008) Segmentation-based detection of allelic imbalance and loss-of-heterozygosity in cancer cells using whole genome SNP arrays. *Genome Biol*, **9**, R136.

# Appendix I: Custom CNV Control Data

Scripts for pre-processing custom control data are for advanced users to prepare their own control data for the CN pipeline. For example, custom control data are needed for non-human CNV detection, which the pipeline does not support by default. The scripts are in the /preprocess directory under the main CN pipeline directory.

For DLE-1 data, the following two scripts can be used for generating the auxiliary input for the fractional CN analysis module (**Table 31**). There is no need to generate input for the integer CN analysis module.

**Table 31**. Two scripts which can be used for generating the auxiliary input for the fractional CN analysis module.

| Step | Objective | Script name | Input | Output |
|---|---|---|---|---|
| **Generate control data for fractional CN module** | Generate the necessary control reference for fractional CN module from control datasets | **generate_control_FractCNV.R** | A list of paths to alignmolvref output | Processed control r.cmap file |
| **Generate CNV mask for fractional CN module** | Generate files to mask high-variance regions for fractional CN module from control datasets | **generate_cnv_mask_FractCNV.R** | A list of paths to alignmolvref output, control reference file | BED file |

### generate_control_FractCNV.R

This wrapper script generates the necessary model parameter data from control samples for the fractional CN pipeline. It takes as input a file that contains a list of paths where each path is the output folder for the molecule-to-reference alignment for each control sample. The path can be the "`alignmolvref/merge`" folder from the *de novo* assembly or Rare Variant Pipeline. Alternatively, if the user independently generated the molecule-to-reference alignment for the control samples, each output path in the list needs to contain one .xmap file and one r.cmap file for each control sample. The script generates a `r.cmap` file that can be used as input to the CN pipeline. The default for sexChr1 is 23 (`cmapID` of human chrX), default for sexChr2 is 24 (`cmapID` of human chrY).

Usage

```
Rscript generate_control_fractCNV.R  --path <input file> --output <output directory> --sexChr1 <cmapID of homogametic sex chromosome> --sexChr2 <cmapID of heterogametic sex chromosome>
```

### generate_cnv_mask_FractCNV.R

This wrapper script generates CNV masks from control samples for the fractional CN pipeline. The script identifies regions with high variance in CN calls in controls; these often indicate centromeres and other regions difficult to call CNs in. Users may pass this file to the CNV pipeline to mask out these regions when calling CNVs in case samples.

Input

It takes as input a file that contains a list of paths where each path is the output folder for the molecule-to-reference alignment for each control sample. The paths can be the `"alignmolvref/merge"` folder from the *de novo* assembly or Rare Variant Pipeline. Alternatively, if the user independently generated the molecule-to-reference alignment for the control samples, each output path in the list needs to contain one .xmap file and one r.cmap file for each control sample. The input parameter `filePostfix` can be provided to indicate the correct r.cmap file to use. The input parameter `refFile` is the CNV control reference file output by `generate_control_FractCNV.R`. Defaults are: `sexChrType "xy,"` `sexChr1 23` (`cmapID` of human chrX), `sexChr2 24` (`cmapID` of human chrY).

Output

**The script generates a BED file, `masks_cnv.bed`.**

Usage

```
Rscript generate_cnv_mask_fractCNV.R --controlDirsList <input file> --refFile <cmap file> --outputDir <output directory> --SVMask <optional SV mask BED file> --sexChr1 <cmapID of homogametic sex chromosome> --sexChr2 <cmapID of heterogametic sex chromosome> --filePostfix <'merged_r.cmap' or '_r.cmap'>
```

For nickase data, the following scripts can be used for generating the auxiliary input for the integer CN analysis module. The order of the rows in **Table 32** indicates the recommended sequence of use.

**Table 32**. The following scripts can be used for generating the auxiliary input for the integer CN analysis module.

| Step | Objective | Script name | Input | Output |
|---|---|---|---|---|
| **Simulation** | Simulate data for confidence table calculation and cross validation | **001_sim_wrapper.R** | autonoise and alignmolvref files from assembly pipeline | Simulated templates |
| **Control samples preparation** | Filter based on input quality and generated eigenvectors by SVD | **002_control_wrapper.R** | Set of control data r.cmap | Control RData file and eigenvector files |
| **Select control samples** | Select the best control datasets based on cross validation | **003_crossValidation.R** | | Performance by selection of control datasets |
| **(optional) Re-generate control samples** | Re-process the control datasets based on results from previous step | **002_control_wrapper.R** | Set of control data r.cmap | Control RData file and eigenvector files |
| **Determine principal components to use** | Find the best number of principle components to use based on cross validation | **004_componentsOptimization.R** | | Performance by number of PC |
| **Generate confidence tables** | Use the PCs, eigenvectors, and templates from previous steps to generate confidence tables | **002_control_wrapper.R** | | Confidence tables |

After pre-processing, users can use the output to update the `default_param_table.txt`, the eigenvector files and confidence tables `under /parameters` and `/testing_data` in the main CNV pipeline.

Below details the input, output, and usage of each wrapper script. The usage of each script is stated in their top script lines. Here, we demonstrate general usage. Detailed explanations for each parameter are in the `defaultParams` section inside each program file.

### 001_simulation_wrapper.R

This wrapper uses assembly pipeline files as input. It simulates CN events, downsamples the molecules in the corresponding regions, and runs `autonoise` and `alignmolvref` steps to form a new r.cmap containing coverage features that correspond to the simulated CN events. The templates and r.cmap are output to a `testing_data/` directory.

Input

This wrapper requires a csv file pointing to the path of each input files. For example, `inputs/simulation_dle1_saphyr_hg19_11092017.csv`:

```
filePath  sampleId  inputType fileType
/saphyr_dle1_hg19/SAMPLE2/autoNoise1_rescaled.bnx  sample18  simulation  scaledbnx
/saphyr_dle1_hg19/SAMPLE2/autoNoise1.errbin  sample18  simulation  errbin
/saphyr_dle1_hg19/SAMPLE2/alignmolvref_merge.xmap  sample18  simulation  xmap
/saphyr_dle1_hg19/SAMPLE2/alignmolvref_merge_r.cmap  sample18  simulation  rcmap
```

If the first line is a raw bnx file, then the fileType should be replaced by `"bnx."` Also, `sampleId` refers to the column name in the `eigenVector.RData` files generated by `002_control_wrapper.R`. A wrong `sampleId` will lead to errors in subsequent step.

Output

1. testing_data/ containing raw_coverage.`RData  templates.RData  templSignals.RData`
2. New `alignmolvref r.cmap`
3. `Intermediate bnx, moleculeID` and `copyNumberTemplate.RData` based on the sampling strategy

Usage

The default assembly command line call is as below. For more parameters, please see the defaultParams section in `001_sim_wrapper.R`.

```
Rscript 001_sim_wrapper.R --alignmentType [platform] --enzyme [enzyme] --reference [reference] --simulationInput [.csv] --resultsDir [dir] --testingDir [dir] --outputFolderName [string] --nBaseFile [.bed] --referenceCMAP [.cmap] --xml [optArg.xml] --refAligner [dir] --pipeline [dir]
```

### 002_control_wrapper.R

This wrapper filters the control datasets based on --controlSdCutoff value, which is the standard deviation of the scaled coverage across chromosomes. Once the sample selection is determined, it can also generate the confidence table using the `testing_data` files generated by `001_control_wrapper.R`.

#### Input

This wrapper requires a directory containing `alignmolvref r.cmap` of the control samples and a `testing_data` directory containing simulation output.

#### Output

1. **`control.RData`**, a matrix storing scaled control sample coverages
2. `eigenvectors.RData` files
3. **`confidence.RData`** files if --confidence option is used

**Usage**

```
Rscript 002_control_wrapper.R --controlsDir [dir] --enzyme [enzyme] --parametersDir [dir] --outliersProbability [float] --componentsRemoved [1:max PC number] --selectControls --controlSdCutoff [float] --confidence --testingDir [dir]
```

### 003_crossValidation.R

This wrapper helps find the best control datasets to use. By default, it sorts the control datasets by the standard deviation of their scaled coverages across chromosomes and evaluates the performance from using the fewest samples to using all samples. The users can also use a random sampling strategy. In this cross validation, the number of components or principal axes (--components Removed option) is fixed for different sampling. However, if the sample size was smaller than the specified number of components, then the number of components would be reduced to the sample size.

**Input**

This wrapper requires `testing_data` files from `001_sim_wrapper.R`, and scaled coverage file `control.RData` from `002_control_wrapper.R`. In addition, it requires a file specifying the column header (control sample names) in `control.RData` to be included in the control sample selection, and the column header in `testing_data/raw_coverage.RData` (simulated sample names) to be included in the simulation sample selection. An example file is `004_componentsOptimization_input.R`.

**Output**

1. Performance data by sample size in table and pdf plot format
2. Intermediate `cnv_call` and `cnv_rcmap` tables

**Usage**

```
Rscript 002_control_wrapper.R --controlsDir [dir] --enzyme [enzyme] --parametersDir [dir] --outliersProbability [float] --componentsRemoved [1:max component number] --selectControls --controlSdCutoff [float] --confidence --testingDir [dir]
```

### 004_componentsOptimization.R

This wrapper helps optimize the number of components to remove. It tests the performance starting from 0 to n components by default, where n is the control sample size. The input and output files are like those for `003_crossValidation.R`, except that the performance table and plot are by number of components instead of sample size.

**Usage**

```
Rscript 004_componentsOptimization.R --testingDataDir [dir] --enzyme [enzyme] --alignmentType [platform] --reference [reference] --parametersDir [dir] --controlCoverageRData [.RData]  --resultsDir [dir] --nGenomesPerSample [integer] --componentsN [min component number : max component number]
```

# Appendix J: Integer CN Pipeline

## Introduction

The Integer CN module was originally released as part of Bionano Solve 3.2.1 and is intended for human Nt.BspQI and Nb.BssSI datasets. It is intended for the homogenous genomes. The FractCNV fractional CN module is preferred for DLE-1 datasets. Theory and performance data for the Integer CN pipeline is preserved here for historical reference.

### OUTPUT

Results from the CN analysis tool are stored in `alignmolvref/copynumber/`. **`cnv_rcmap_exp.txt`** contains per-label coverage information. The format of this file is like the standard CMAP format, but with several additional columns, the definitions of which depend on which pipeline is run (see **Table 33** below).

**Table 33**. per-label coverage information

| Column name | Integer CN pipeline |
|---|---|
| **ScaledCoverage** | Sample coverage divided mean coverage for each chromosome |
| **Normalized Coverage** | Scaled Coverage normalized by SVD |
| **Copy Number** | Rounded copy number states |
| **fractional Copy Number** | Smooth copy number states |
| **MeanCov** | NA |

**`cnv_calls_exp.txt`** contains the start and end positions of copy number variant (CNV) calls (those whose CN states differ from baseline). For the integer CN pipeline, confidence refers to the probability of a CNV call as being a true call.

**`cnv_calls_exp_full.txt`** is like `cnv_calls_exp.txt`. This file contains the combined results from the integer and fractional CN modules. An additional column named “Algorithm” denotes which module a call is from. Calls from the integer CN module are denoted as “Label-based.”

## Usage

**Table 34**. CNV.R Parameters to Integer CN Module

| Parameter | Notes (corresponding parameters in CNV.R in the first line of each entry) |
|---|---|
| cd | --controlsDir [dir]<br>User-provided control data directory that contains r.cmap output from alignmolvref. We recommend using at least ten control datasets in total and at least five of each gender. This option automatically triggers calculation of new parameters and new confidence tables for hg19 and hg38. If the control data is from an unrecognized source (for example, non-human, generated from a different enzyme), the CN pipeline would output CNV calls without confidence scores. Please refer to the "Pre-process custom control data" section on how to generate custom confidence tables. |
| pd | --parametersDir [dir]<br>User-provided path to either store processed control data and confidence tables when --controlsDir is defined, or as a different parameter input directory from the default parameter directory. |
| op | --outliersProbability [float]<br>"1 - Type I error rate" for outlier detection. |

## Theory

The main steps of the integer CN pipeline are:

- Scaling
- Normalizing
- Outlier detection
- Smoothing
- Confidence score calculation

### SCALING

In the scaling step, the per-label coverage is standardized: coverage at each label is divided by the average coverage of all autosomal labels and then multiplied by 2. The pipeline currently assumes that the input comes from a diploid genome; therefore, the scaled per-label coverage should be non-negative and should average at 2.

### NORMALIZING

The objective of the normalizing step is to reduce the non-CNV related variance of the per-label-coverage, minimizing noise in the coverage profile. To partition that variance out of the total variance, a dimension reduction method is implemented. Features that contribute to the variance in the control coverage profiles (from non-diseased samples that have no known large CNV events) are extracted using singular vector decomposition (SVD). The variance attributed to noise (e.g., rare CNV events, coverage fluctuation around fragile sites (for nickase data), and segmental duplication and N-base regions, and random sampling error) accounts for a sizable

portion of the total variance. Theoretically, they can be explained by the first *k* principal axes, which are removed during normalization by subtracting them out from the scaled coverage profile such that only the variance related to CNV events is retained.

Processed control data are stored in the “parameters” directory. Normalization for a given chromosome may be skipped if the query profile is deemed incompatible with the control data. This is sometimes observed when analyzing the sex chromosomes and for highly rearranged genomes.

#### OUTLIER DETECTION

The pipeline looks for labels with outlier coverage values; presumably, these labels are from CNV regions. Within the outlier detection step, there are two main sub-steps. First, a Hartigan’s dip statistics test for unimodality is applied to each chromosome to check if there is a large CN event up to 25% of the chromosome length. If so, the chromosome is split into segments, each with a separate baseline CN state. This allows the pipeline to identify large CNV changes. Then, a Chi-square test is performed for each label to determine if the coverage of a label is significantly different from the mean of all labels within the same segment. The default type I error rate for the Chi-square test is 0.05, corresponding to parameter [outlierProbability] 0.95.

#### SMOOTHING

In the smoothing step, several methods are used together to “stitch” together the per-label data from **Step 3**, smooth the normalized coverage profile, and generate the fractionalCopyNumber and CopyNumber columns.

The fractionalCopyNumber column is generated by a “gentle” smoothing method. First, the normalized coverage is rounded up to integer CN. Next, all CN values within median CN ± 0.25 inside a segment are smoothed to median CN. Then, the coverage profile is smoothed by applying running median of window size of 11 labels. Lastly, within each segment, neighboring outlier labels of the same direction (for example, either elevation or depletion relative to the segment median) are clustered, and their values are replaced by the extreme value within that cluster.

An “aggressive” smoothing method is applied to obtain CopyNumber. Neighboring segments with similar fractionalCopyNumber levels are merged if 1) there are at least 55 labels in them, and 2) the lengths of the segments that have CN larger (or smaller) than the median CN are longer than 80% of the total size of all close by fractionalCopyNumber. Their values are replaced by the median of the cluster. CNV events are sometimes broken up into small segments; this step ensures that these small segments are merged.

#### CONFIDENCE SCORE CALCULATION

The Confidence column in CNV calls in cnv_calls_exp.txt comes from the confidence tables in /parameters/`confidence_enzyme_alignmentType_reference.RData`. Each confidence table contains positive predictive value (PPV) by CN state and by size, starting from 500 kbp based on simulations. CNV calls falling into the same size bin and the same CN state in the confidence table would be assigned the same confidence score of that bin. Typically, smaller, and heterozygous CNV calls have lower confidence scores.

The simulation was conducted for each enzyme (Nt.BspQI, Nb.BssSI, or DLE-1), platform (Saphyr, or Stratys) and reference (hg19 or hg38) combination, with two base control datasets being chosen from which CNV events were simulated. Duplication and deletion events with CN states 0, 1, 2, 3, and 4 were randomly simulated across the genome. The molecules overlapping with simulated CNV events were sampled according to the simulated CN

states. Then, the sampled molecule set was aligned to the reference, and the alignment output was used as input to the CNV pipeline. CN calling performance was assessed, and PPV data were stored in the confidence tables.

If the input r.cmap is from an unrecognized source, the user needs to provide custom control datasets as input. Confidence tables need to be manually generated as well; otherwise, there would be no confidence output in the cnv_calls_exp.txt file.

**NOTE:** The integer CN pipeline does not run on non-human datasets without the necessary auxiliary data, and corresponding columns in the output will be zeros.

### Copy Number Variant Calling Performance using Simulated Data

The integer CN pipeline is automatically run when it detects that the input is either a BspQI or BssSI dataset. High sensitivity and PPV were observed for events > 500 kbp, seen in **Figure 47**. Generally, increasing coverage increases the signal-to-noise ratio; however, sensitivity for simulated events did not significantly increase beyond 80X (). Higher PPV was observed at lower coverage levels, but sensitivity was significantly lower at those coverage levels.

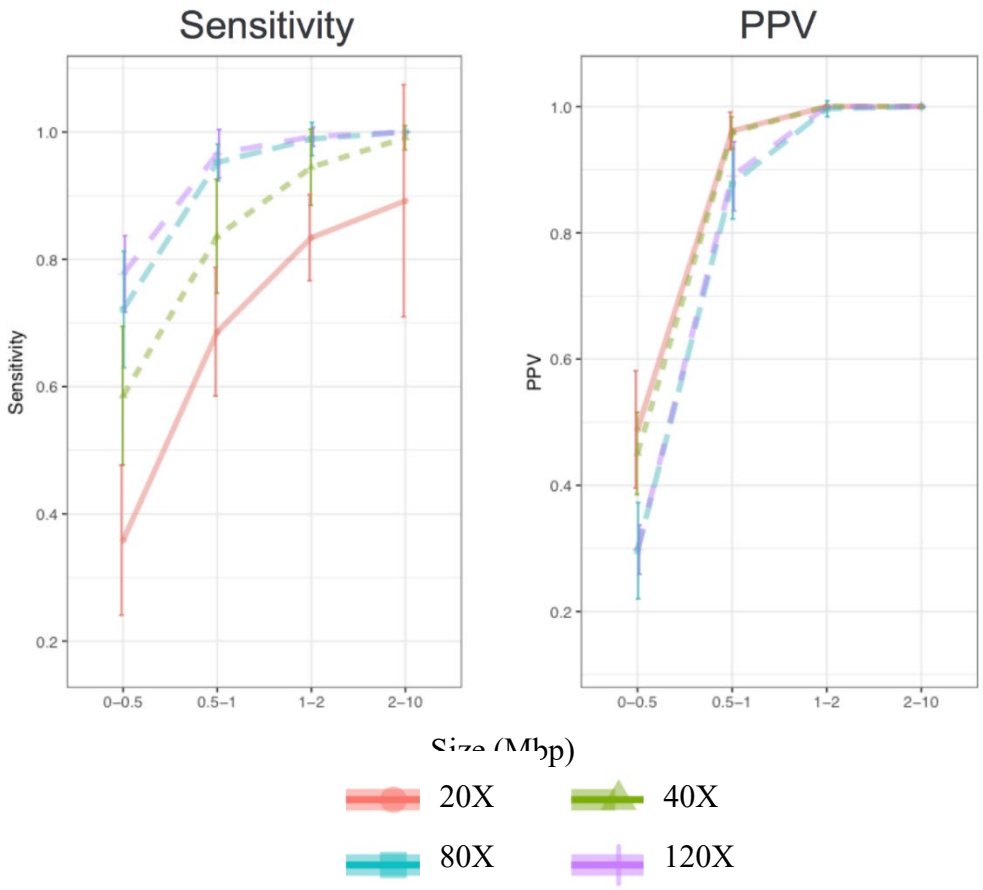



**Figure 47**. Copy number variant calling performance at different coverage levels with simulated DLE-1 data.

### Copy Number Variant Calling Performance using Real Data

Copy number events of varied sizes and zygosity states were expected in five well-characterized samples (**Table 35**). The smallest expected event (160 kbp) was not detected in either BspQI or BssSI data; all other events were detected.

**Table 35**. CNV calling performance from real datasets (at close to or less than 80X effective coverage).

| | Chr | Size | Type | Zygosity | Detected in Nt.BspQI? | Detected in Nb.BssSI? |
|---|---|---|---|---|---|---|
| **Sample 1** | chr2 | 160 kbp | Duplication | Heterozygous | No | No |
| | chr2 | 380 kbp | Duplication | Heterozygous | Yes | Yes |
| **Sample 2** | chr3 | 540 kbp | Duplication | Heterozygous | Yes | Yes |
| **Sample 3** | chr6 | 1 Mbp | Deletion | Heterozygous | Yes | N/A |
| **Sample 4** | chr17 | 18 Mbp | Duplication | Homozygous | Yes | Yes |
| | chr8 | 75 Mbp | Duplication | Homozygous | Yes | Yes |
| **Sample 5** | chr6 | 46 Mbp | Deletion | Heterozygous | Yes | N/A |
| | chr18 | 65 Mbp | Duplication | Heterozygous | Yes | N/A |
| | chr11 | 67 Mbp | Duplication | Heterozygous | Yes | N/A |
| | chr1 | 106 Mbp | Duplication | Heterozygous | Yes | N/A |

# Appendix K: Historical Performance Data

## Introduction

Performance data presented here was included in previous versions of this document. It is retained here for completeness.

## *De novo* Assembly Pipeline

### PERFORMANCE FOR INSERTIONS AND DELETIONS

CHM1/13 analysis was performed on an *in silico* mixture of CHM1 and CHM13 DLE-1 datasets. SV calls were compared from pure CHM1 and CHM13 assemblies against SV calls from the mixture. Overall performance is presented for SVs larger than 700 bp (**Table 36**).

### CHM1/13 ANALYSIS

Data was generated from homozygous CHM1 and CHM13 cell lines initially derived from hydatidiform moles. Single-molecule maps from CHM1 and CHM13 were evenly sampled and combined *in silico* to simulate a diploid genome at 80X effective coverage. The CHM1, CHM13, and the CHM1/13 mixture molecule sets were assembled separately.

A three-way SV comparison analysis was performed for each trio set of CHM1, CHM13, and CHM1/13 mixture SV calls. The SV calls from the CHM1 and CHM13 pure assemblies were considered as the (conditional) ground truth. The sensitivity and positive predicted value (PPV) at different coverage levels were analyzed. Sensitivity was defined as fraction of SV calls in pure assemblies that were called in the mixture assembly, and PPV was defined as fraction of calls in mixture assembly that were called in the pure assemblies.

**Table 36**. Insertion and deletion (> 700 bp) calling performance from CHM1/13 datasets using DLS.

| Size cut-off | Type | Expected zygosity in mixture | In silico mixture | Individual assemblies | Fraction captured (%) | PPV (%) |
|---|---|---|---|---|---|---|
| **700 bp** | Insertions | Homozygous | 1,975 | 1,963 | 99.4 | 97.5 |
| | | Heterozygous | 1,737 | 1,480 | 85.2 | |
| | Deletions | Homozygous | 695 | 687 | 98.9 | 97.1 |
| | | Heterozygous | 1,189 | 1,085 | 91.3 | |

### CEPH TRIO ANALYSIS

Using genome mapping, Mak *et al.* analyzed a Caucasian trio from the 1000 Genomes Project (the parents NA12891 and NA12892, and the daughter NA12878) and published an expert-curated SV list. We reanalyzed the same starting data and compared the resulting SV list from the automated pipeline against the curated list.

CEPH trio Nt.BspQI datasets used in Mak *et al*[4]. were re-analyzed, and the resulting SV lists were compared with the published SV lists. Overall, we detected most of the published calls and made a large number of new SV calls. For example, we detected 93% of the published deletions in NA12878 and made an additional 764 deletion calls.

### PERFORMANCE FOR TRANSLOCATION BREAKPOINTS[2]

Nine samples with annotated single translocations were tested for translocation calling performance (). We found the expected translocation breakpoints in eight out of nine samples. For the remaining sample, only 10% of the cells contained the expected translocation based on karyotyping results. Reciprocal breakpoints were found in six out of eight samples for which expected translocations were found. In addition, putative FP calls were masked and/or filtered (**Table 37**).

**Table 37**. Translocation calling performance from real datasets (at close to or less than 80X effective coverage).

| Sample | Diagnosis | Annotation | Sample prep | Found expected breakpoint? | Found reciprocal breakpoints? |
|---|---|---|---|---|---|
| Sample01 | CML | FISH: t(9;22) | Plug lysis BspQI | Yes | Yes |
| Sample02 | CLL | FISH: t(11;14) | Plug lysis BspQI | Yes | No |
| Sample03 | CML | FISH: t(9;22) | Plug lysis BspQI | Yes | Yes |
| Sample04 | CML | Kary.: t(9;22) | Plug lysis BspQI and BssSI | Yes | Yes |
| Sample05 | AML | Kary.: t(7;11) | Plug lysis BspQI | Yes | No |
| Sample06 | AML | Kary.: t(8;21) | Plug lysis BspQI | Yes | Yes |

[4] Mak AC et al. ”ak AC et alger enzyme.ion breakpoints, inversion as 0. be valid.)t resolutionncy estimates. es that take more than 10 days.izatGenome-Wide Structural Variation Detection by Genome Mapping on Nanochannel Arrays. Genetics. 2016 Jan;202(1):351-62.

| Sample | Diagnosis | Annotation | Sample prep | Found expected breakpoint? | Found reciprocal breakpoints? |
|---|---|---|---|---|---|
| Sample07 | AML | Kary.: t(4;5) | Plug lysis BspQI and BssSI | No | N/A |

Cell lines GM16736 and GM21891 obtained from the Coriell Institute for Medical Research each contained one known translocation t(9;22) and t(4;15), respectively. Based on 80X effective coverage assemblies, we found the expected translocation breakpoints for both samples and additional translocation breakpoints (see **Table 38**). The additional breakpoint in GM21891 was filtered out after applying a confidence threshold of 0.1. An additional breakpoint with confidence score of 0.19 remained in the GM16736 BssSI assembly, which may be an unannotated translocation call in the sample. The BssSI masks were generated based on a small control database; thus, they may be less effective than the BspQI masks.

**Table 38**. Translocation breakpoint call masking and filtering for Coriell samples GM16736 and GM21891.

| | | Coriell (BspQI) | | Coriell (BssSI) | |
|---|---|---|---|---|---|
| | #Translocation calls | GM16736 | GM21891 | GM16736 | GM21891 |
| | Known Translocation | 1 | 1 | 1 | 1 |
| hg19 | Remaining calls after masking | 1 | 2 | 2 | 2 |
| hg19 | As expected? | Yes | 1 add. call | 1 add. call | 1 add. call |
| hg19 | As expected after applying confidence threshold >= 0.1 | Yes | Yes | 1 add. call | Yes |
| hg38 | Remaining calls after masking | 1 | 2 | 5 | 2 |
| hg38 | As expected? | Yes | 1 add. call | 4 add. call | 1 add. call |
| hg38 | As expected after applying confidence threshold >= 0.1 | Yes | Yes | 1 add. call | Yes |

## PERFORMANCE FOR INVERSION BREAKPOINTS

Three Coriell cell lines with known inversion events were used for assessing inversion calling performance. Two-enzyme data were generated for two of the three samples and expected inversion events in all three cell lines were observed at the annotated locations (**Table 39**).

**Table 39**. Inversion calling performance from real datasets (at close to or less than 80X effective coverage). *Called as intra-chromosomal translocation breakpoints, since those inversions were larger than 5 Mbp.

| Sample | Diagnosis | Annotation | Sample prep | Found expected inversion? |
|---|---|---|---|---|

| | | | | |
|---|---|---|---|---|
| GM19238 | Phenotypically normal | Inv(15q13.3) | Plug lysis BspQI | Yes |
| GM14266 | Micrognathia | Inv(4q34.2-35.2) | SVMerge | Yes* |
| GM21074A | Developmental delay | Inv(2p23-q31) | SVMerge | Yes* |

## Copy Number Variant Calling Performance using Real Data

Performance data are shown in **Table 40** and **Table 41** based on nine cancer samples (ranging from 200X to 300X coverage) showed high sensitivity to events at > 0.15 VAF. Detection of 0.1 VAF events was impacted by local noise in and complexity of CN profile.

**Table 40**. Fraction CNV detection sensitivity with real DLE-1 datasets for duplications.

| Duplications | | | | |
|---|---|---|---|---|
| Size/AF | 0.1 | 0.15 | 0.2 | 0.3 |
| 0 – 500 kbp | 0.5 | 0.68 | 0.79 | 0.91 |
| 500 kbp – 1 Mbp | 0.5 | 0.9 | 0.9 | 1 |
| 1 – 3 Mbp | 0.69 | 0.92 | 0.98 | 1 |
| 3+ Mbp | 0.78 | 0.97 | 0.97 | 0.97 |
| Deletions | | | | |
| Size/AF | 0.1 | 0.15 | 0.2 | 0.3 |
| 0 – 500 kbp | 0.85 | 0.85 | 0.87 | 0.95 |
| 500 kbp – 1 Mbp | 0.97 | 0.97 | 0.97 | 1 |
| 1 – 3 Mbp | 0.95 | 0.98 | 0.98 | 0.98 |
| 3+ Mbp | 0.53 | 1 | 0.84 | 0.95 |

**Table 41**. Fraction CNV detection PPV with real DLE-1 datasets for deletions.

| VAF | PPV |
|---|---|
| **0.1** | 0.98 |

| | |
|---|---|
| **0.15** | 0.98 |
| **0.2** | 0.97 |
| **0.3** | 0.97 |
| **1** | 1.00 |

# Technical Assistance

For technical assistance, contact Bionano 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:<br>Monday through Friday, 9:00 a.m. to 5:00 p.m., PST<br>US: +1 (858) 888-7663<br><br>Monday through Friday, 9:00 a.m. to 5:00 p.m., CET<br>UK: +44 115 654 8660<br>France: +33 5 37 10 00 77<br>Belgium: +32 10 39 71 00 |
| **Website** | www.bionano.com/support |
| **Address** | Bionano, Inc.<br>9540 Towne Centre Drive, Suite 100<br>San Diego, CA 92121 |

# Legal Notice

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



### Patents

Products of Bionano Genomics® may be covered by one or more U.S. or foreign patents.

### Trademarks

The Bionano logo and names of Bionano products or services are registered trademarks or trademarks owned by Bionano Genomics, Inc. (“Bionano”) in the United States and certain other countries.

Bionano™, Bionano Genomics®, Saphyr®, Saphyr Chip®, Bionano Access™, Stratys™, Stratys™ Compute, Stratys™ Chip, and Bionano EnFocus™ are trademarks of Bionano Genomics, Inc. All other trademarks are the sole property of their respective owners.

No license to use any trademarks of Bionano is given or implied. Users are not permitted to use these trademarks without the prior written consent of Bionano. The use of these trademarks or any other materials, except as permitted herein, is expressly prohibited and may be in violation of federal or other applicable laws.