Bionano Solve Theory of Operation: Structural Variant Calling

Document Number:  30110
Document Revision:  C
# Table of Contents

- Legal Notice .......................................................................................................................... 3
- Introduction .......................................................................................................................... 4
- Structural variant calling with Bionano Solve v3.1 ............................................................... 5
  - De novo assembly .................................................................................................................. 5
  - SV calling ............................................................................................................................. 5
- Zygosity classification ........................................................................................................... 6
- Confidence modeling ............................................................................................................. 6
  - Confidence for insertions and deletions .............................................................................. 6
  - Confidence for inversion and translocation breakpoints .................................................... 7
- Masking ................................................................................................................................ 7
- SVMerge ............................................................................................................................... 7
- Structural variant calling performance .................................................................................... 8
- Structural variant calling performance using simulated data ................................................ 8
  - Performance for simulated insertions and deletions ......................................................... 8
  - Performance for simulated translocation breakpoints ...................................................... 9
  - Performance for simulated inversion breakpoints ............................................................ 9
- Performance for duplications .............................................................................................. 12
- Structural variant calling performance using real data .......................................................... 13
  - Performance for insertions and deletions ......................................................................... 13
  - Performance for translocation breakpoints ...................................................................... 14
  - Performance for inversion breakpoints ........................................................................... 15
- Bionano Access integration .................................................................................................. 16
- SMAP-to-VCF converter support ........................................................................................... 18
- Additional Considerations .................................................................................................... 18
- FAQs .................................................................................................................................. 19
- Appendix ............................................................................................................................. 22
  - Method for assessing SV calling performance ................................................................. 22
    - SV calling performance with simulated data ............................................................... 22
    - SV calling performance with experimental data ......................................................... 23
  - Mask generation ................................................................................................................. 24
  - Compiling common translocation breakpoint list ............................................................ 24
  - Compiling annotated segmental duplication regions ....................................................... 24
Legal Notice

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

This material is protected by United States Copyright Law and International Treaties. Unauthorized use of this material is prohibited. No part of the publication may be copied, reproduced, distributed, translated, reverse-engineered or transmitted in any form or by any media, or by any means, whether now known or unknown, without the express prior permission in writing from Bionano Genomics. Copying, under the law, includes translating into another language or format. The technical data contained herein is intended for ultimate destinations permitted by U.S. law. Diversion contrary to U. S. law prohibited. This publication represents the latest information available at the time of release. Due to continuous efforts to improve the product, technical changes may occur that are not reflected in this document. Bionano Genomics reserves the right to make changes in specifications and other information contained in this publication at any time and without prior notice. Please contact Bionano Genomics Customer Support for the latest information.

BIONANO GENOMICS DISCLAIMS ALL WARRANTIES WITH RESPECT TO THIS DOCUMENT, EXPRESSED OR IMPLIED, INCLUDING BUT NOT LIMITED TO THOSE OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. TO THE FULLEST EXTENT ALLOWED BY LAW, IN NO EVENT SHALL BIONANO GENOMICS BE LIABLE, WHETHER IN CONTRACT, TORT, WARRANTY, OR UNDER ANY STATUTE OR ON ANY OTHER BASIS FOR SPECIAL, INCIDENTAL, INDIRECT, PUNITIVE, MULTIPLE OR CONSEQUENTIAL DAMAGES IN CONNECTION WITH OR ARISING FROM THIS DOCUMENT, INCLUDING BUT NOT LIMITED TO THE USE THEREOF, WHETHER OR NOT FORESEEABLE AND WHETHER OR NOT BIONANO GENOMICS IS ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

Patents

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

Trademarks

The Bionano Genomics logo and names of Bionano Genomics products or services are registered trademarks or trademarks owned by Bionano Genomics in the United States and certain other countries.

Bionano Genomics®, Irys®, IrysView®, IrysChip®, IrysPrep®, IrysSolve®, Saphyr™, Saphyr Chip™, and Bionano Access™ 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 Genomics is given or implied. Users are not permitted to use these trademarks without the prior written consent of Bionano Genomics. 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.

©Copyright 2017 Bionano Genomics, Inc. All rights reserved.
Introduction

Structural variation (SV) is a common source of sequence variation, involving deletion, insertion, and rearrangement of genomic material. It has the potential to impact large stretches of 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.

To enable comprehensive SV analysis, Bionano Genomics' latest de novo assembly and SV calling pipeline (Bionano Solve™ v3.1) features a haplotype-aware assembler specifically designed to detect and differentiate allelic differences, thus enabling effective and comprehensive detection of homozygous and heterozygous SVs. All major SV types are supported, and extensive validation based on simulated and experimental data showed high sensitivity and positive predictive value. In this latest Bionano Solve v3.1 release, sensitivity for inversions has been much improved, and detection of duplications has been added.

Bionano Solve includes additional tools to facilitate downstream analyses. To take advantage of the availability of the second enzyme, Nb.BssSI (BssSI), for human analysis, SVMerge analyzes and merges SV calls from single-enzyme assemblies. The use of two enzymes provides multiple benefits and additional sensitivity gains. Also, the variant annotation pipeline (VAP) annotates SV calls and provides information to assess if a variant is inherited or de novo, which is key to understanding relevance to a presented phenotype (see Bionano Solve Theory of Operation: Variant Annotation, document 30190 for more detail).

Bionano Solve is fully integrated with Bionano Access™ v1.1, which provides a user-friendly interface for streamlining analysis. Bionano Access is useful for managing projects, analyzing run results, and providing data visualization (Figure 1).

Figure 1: Visualization in Bionano Access of a heterozygous deletion confirmed by both the Nb.BspQI and Nb.BssSI assemblies.
Structural variant calling with Bionano Solve v3.1

De novo assembly

Bionano’s *de novo* assembly algorithm is built upon an implementation of the overlap-layout-consensus strategy with a maximum likelihood model for evaluating alignments. 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.

We implemented haplotype-aware components to effectively assemble heterozygous regions. During the extension stages of the assembly, we analyze molecule-to-genome map alignments, identify clusters of molecules with coordinated disrupted alignment and assemble these clusters separately. This is critical for assembling haplotype maps with large differences and for detecting a wide range of variants. It also improves assembly of segmental duplication regions, where large stretches of sequence appear more than once in the genome. In addition, 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.

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.

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 otherwise.

A fusion point between distant regions of the genome is identified as a translocation breakpoint. Intrachromosomal translocation breakpoints involve regions at least 5 Mbp away from each other on the same chromosome. Interchromosomal translocation breakpoints involve regions on different chromosomes.

Inversion breakpoint calls involve neighboring alignments with opposite orientations. Small inversions which could be spanned by single genome maps may be reported as paired inversion breakpoints. Currently, inversions larger than 5 Mbp are called as intra-chromosomal translocation breakpoints. In this Bionano Solve v3.1 release, we improved detection of small inversions (whose inverted regions contained at least four labels) by searching in a limited space for potential inverted alignments. Paired inversion breakpoints ("inversion_paired") are reported for such small inversions. For other inversion breakpoint calls, there are two linked entries ("inversion" and "inversion_partial"), specifying six coordinates of interest (Figure 2).

Duplications are detected based on direct or indirect evidence of duplication within single maps. There is direct evidence when two places on the map align to the same place on the reference. However, the map may not span whole copies of the duplicated sequence. In those cases, the alignments may not overlap on the reference. Based on the alignment signature, one could infer duplication events. The new duplication detection algorithm supports tandem duplications and non-tandem duplications whose copies are sufficiently close to each other on the map.
Zygosity classification

Zygosity is a classification of an SV call as homozygous, heterozygous, or unknown. It is currently assigned to only insertion, deletion, translocation breakpoint, and inversion breakpoint calls based on the following two criteria: 1) whether the call in question overlaps another call, and 2) whether the call in question overlaps alignment of another genome map that shows no SV.

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 ‘unknown’. For this purpose, different sets of criteria are applied for different SV types. For insertion and deletion calls, they must have at least 80% overlap and size similarity. For translocation breakpoint calls, the breakpoints must be within 100 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. Zygosity is not currently assigned to duplication calls.

Confidence modeling
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. It takes into account 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.
Confidence for inversion and translocation breakpoints

Inversion and translocation confidence modeling was performed based on a supervised training approach, anomaly detection. In this case, we trained our classifier to detect false positive (FP) calls as anomalies relative to simulated true positive (TP) calls.

For modeling, we constructed an initial set of predictors likely related to confidence. The predictors included information about the molecule-to-map and map-to-reference alignments. For each of the initial predictors, we performed a t-test to check whether there was a statistically significant difference ($p < 0.05$) between TP and FP calls. The predictors that showed significant differences were kept, and the highly correlated predictors were removed. The classifier was trained on four assemblies based on simulated data.

The confidence score for a translocation breakpoint call, which ranges from 0 to 1, correlates with the Mahalanobis distance of the call to the mean of the empirical distribution constructed based on predictor values from the set of TP calls. The squared Mahalanobis distance is calculated as:

$$D^2 = (x - m)^T C^{-1} (x - m)$$

where $x$ is a vector of predictor values for the query call, $m$ is a vector of means of predictors from the TP calls used for training, and $C$ is the variance covariance matrix of the predictors. The recommended cutoffs are 0.1 for translocation and 0.01 for inversion breakpoints calls.

Masking

Using a BED file provided by Bionano, insertion and deletion calls overlapping N-base gaps in the reference and putative FP translocation breakpoint calls can be masked. The former would be annotated with a suffix “_nbase” in the SV type. The latter would be annotated with a suffix “_common” or “_segdupe” in the SV type, depending on whether they overlap with common FP calls or annotated segmental duplication regions, respectively. The hg19 BspQI version of the translocation mask covers 81.8 Mbp, or 2.6% of the reference. For example, it includes selected sub-centromeric and sub-telomeric regions that are prone to generating putative FP translocation breakpoint calls. We also provide BED files for hg38 and BssSI. The BED files have been updated for this release with additional control datasets. Details on BED generation are included in the Appendix and FAQ sections.

SVMerge

SVMerge allows users to take advantage of having two single-enzyme datasets (for example, BspQI and BssSI). It currently supports merging of insertion, deletion, translocation breakpoint, and inversion breakpoint calls. SVMerge provides several potential benefits. The complementary nature of the two enzymes helps improve sensitivity compared to one-enzyme detection. Cross-confirmation by two independent datasets provides a useful means of validating SV calls. SVMerge also improves SV breakpoint accuracy. For insertions and deletions, SV size estimates are expected to be more accurate. The confidence scores for insertions and deletions reflect whether just one or both enzymes support an SVMerge call.

Briefly, SVMerge examines SV calls from single-enzyme assemblies and evaluates whether the calls are overlapping. The software merges the overlapping calls and outputs refined breakpoint coordinates.
Structural variant calling performance

For performance validation, we took two complementary approaches. We used simulated data and real data to assess SV detection performance. Detailed methods are included in the Appendix section.

Structural variant calling performance using simulated data

Performance for simulated insertions and deletions

Molecules were simulated at 70X 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. SVMerge provides additional sensitivity improvement relative to single-enzyme detection (Figure 3, Tables 1-2).

Figure 3. Heterozygous insertion and deletion calling performance from simulated datasets.

Table 1. SVMerge insertion (>1 kbp) calling performance from simulated datasets under different conditions. All insertions were expected to be heterozygous except when specified otherwise in the Zygosity column.

<table>
<thead>
<tr>
<th>Sample prep</th>
<th>Sample Prep Type</th>
<th>Coverage (X)</th>
<th>Sensitivity</th>
<th>PPV</th>
</tr>
</thead>
<tbody>
<tr>
<td>Plug lysis</td>
<td>Het.</td>
<td>70</td>
<td>75.0%</td>
<td>98.6%</td>
</tr>
<tr>
<td>OptiDNA</td>
<td>Het.</td>
<td>70</td>
<td>90.2%</td>
<td>98.6%</td>
</tr>
<tr>
<td>OptiDNA</td>
<td>Het.</td>
<td>110</td>
<td>90.2%</td>
<td>98.6%</td>
</tr>
</tbody>
</table>

Performance data for insertion, deletion, and translocation calling were published in the previous version of the Theory of Operation (Part Number 30110 Rev B) and are included here for completeness.
Table 2. SVMerge deletion (>1 kbp) calling performance from simulated datasets under different conditions. All deletions were expected to be heterozygous except when specified otherwise in the Zygosity column.

<table>
<thead>
<tr>
<th>Sample Prep Type</th>
<th>Zygosity</th>
<th>Coverage (X)</th>
<th>Sensitivity</th>
<th>PPV</th>
</tr>
</thead>
<tbody>
<tr>
<td>Plug lysis</td>
<td>Het.</td>
<td>70</td>
<td>90.6%</td>
<td>98.9%</td>
</tr>
<tr>
<td>OptiDNA</td>
<td>Het.</td>
<td>70</td>
<td>95.1%</td>
<td>98.6%</td>
</tr>
<tr>
<td>OptiDNA</td>
<td>Homo.</td>
<td>50</td>
<td>97.2%</td>
<td>99.3%</td>
</tr>
<tr>
<td>OptiDNA</td>
<td>Het.</td>
<td>70</td>
<td>95.1%</td>
<td>98.6%</td>
</tr>
<tr>
<td>OptiDNA</td>
<td>Het.</td>
<td>70</td>
<td>95.1%</td>
<td>98.6%</td>
</tr>
<tr>
<td>OptiDNA</td>
<td>Het.</td>
<td>110</td>
<td>96.7%</td>
<td>98.2%</td>
</tr>
</tbody>
</table>

Performance for simulated translocation breakpoints

Molecules were simulated from unedited and edited versions of the hg19 reference genome (with 918 translocated fragments, or transposition events, of different sizes) and used for assembly and SV calling with the Bionano Solve pipeline. Sensitivity for expected breakpoints associated with translocated fragments larger than 300 kbp are shown (Table 3).

Table 3. Translocation calling performance from simulated datasets under different conditions. All translocation breakpoints were expected to be heterozygous except when specified otherwise in the Zygosity column.

<table>
<thead>
<tr>
<th>Sample prep type</th>
<th>Enzyme</th>
<th>Coverage (X)</th>
<th>Zygosity</th>
<th>Sensitivity</th>
</tr>
</thead>
<tbody>
<tr>
<td>OptiDNA</td>
<td>BspQI</td>
<td>70</td>
<td>Het.</td>
<td>90.7%</td>
</tr>
<tr>
<td>OptiDNA</td>
<td>BspQI</td>
<td>70</td>
<td>Het.</td>
<td>83.0%</td>
</tr>
<tr>
<td>OptiDNA</td>
<td>BspQI</td>
<td>70</td>
<td>Het.</td>
<td>97.9%</td>
</tr>
<tr>
<td>Plug lysis</td>
<td>BspQI</td>
<td>70</td>
<td>Het.</td>
<td>77.3%</td>
</tr>
<tr>
<td>Plug lysis</td>
<td>BspQI</td>
<td>70</td>
<td>Het.</td>
<td>73.6%</td>
</tr>
<tr>
<td>Plug lysis</td>
<td>BspQI</td>
<td>70</td>
<td>Het.</td>
<td>91.9%</td>
</tr>
<tr>
<td>OptiDNA</td>
<td>BspQI</td>
<td>50</td>
<td>Het.</td>
<td>88.4%</td>
</tr>
<tr>
<td>OptiDNA</td>
<td>BspQI</td>
<td>70</td>
<td>Het.</td>
<td>90.7%</td>
</tr>
<tr>
<td>OptiDNA</td>
<td>BspQI</td>
<td>90</td>
<td>Het.</td>
<td>90.7%</td>
</tr>
<tr>
<td>OptiDNA</td>
<td>BspQI</td>
<td>110</td>
<td>Het.</td>
<td>91.9%</td>
</tr>
<tr>
<td>OptiDNA</td>
<td>BspQI</td>
<td>70</td>
<td>35/35</td>
<td>90.7%</td>
</tr>
<tr>
<td>OptiDNA</td>
<td>BspQI</td>
<td>70</td>
<td>30/40</td>
<td>84.9%</td>
</tr>
<tr>
<td>OptiDNA</td>
<td>BspQI</td>
<td>70</td>
<td>20/50</td>
<td>61.7%</td>
</tr>
<tr>
<td>OptiDNA</td>
<td>BspQI</td>
<td>110</td>
<td>Het.</td>
<td>91.9%</td>
</tr>
<tr>
<td>OptiDNA</td>
<td>BspQI</td>
<td>55</td>
<td>Hom.</td>
<td>88.9%</td>
</tr>
</tbody>
</table>

Performance for simulated inversion breakpoints

Molecules were simulated from unedited and edited versions of the hg19 reference genome (with about 900 inversions from 5 kbp to 1 Mbp) and used for assembly and SV calling with the Bionano Solve pipeline. The new Bionano Solve 3.1 pipeline provides much improved sensitivity for inversions larger than 30 kbp (Figure 4). Additional sensitivity gain is expected with the use of a second enzyme (Figure 5). We found sensitivity to be correlated with input coverage; however, little gain was observed beyond 70X (Figure 6).
Figure 4. Comparison of heterozygous inversion sensitivity between Bionano Solve 3.0 and Solve 3.1 using simulated 70X BspQI OptiDNA data.

Figure 5. Comparison of heterozygous inversion sensitivity between 1-enzyme plug lysis and 2-enzyme OptiDNA workflows using simulated 70X data.
Figure 6. Comparison of heterozygous inversion sensitivity at different input coverage levels.
Performance for duplications

Sensitivity for heterozygous duplications (of which about half were tandem and half were inverted) depends on the duplication size. High sensitivity was observed for duplications larger than 50 kbp (Figure 7).

Figure 7. Heterozygous duplication sensitivity based on 70X of simulated OptiDNA data. 2-enzyme performance was based on whether a duplication was detected by either enzyme.
Structural variant calling performance using real data

Performance for insertions and deletions

CHM1/13 analysis was performed on an in silico mixture of CHM1 and CHM13 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 1.5 kbp (Table 4).

Table 4: Insertion and deletion calling performance from CHM1/13 datasets.

<table>
<thead>
<tr>
<th>Size cut-off</th>
<th>Type</th>
<th>Expected zygosity in mixture</th>
<th>In silico mixture</th>
<th>Individual assemblies</th>
<th>Fraction captured (%)</th>
<th>PPV (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.5 kbp</td>
<td>Insertions</td>
<td>Homozygous</td>
<td>700</td>
<td>707</td>
<td>99.0</td>
<td>97.9</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Heterozygous</td>
<td>554</td>
<td>663</td>
<td>83.6</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Deletions</td>
<td>Homozygous</td>
<td>268</td>
<td>269</td>
<td>99.6</td>
<td>97.1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Heterozygous</td>
<td>477</td>
<td>517</td>
<td>92.3</td>
<td></td>
</tr>
</tbody>
</table>

CEPH trio datasets used in Mak et al. 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.

Table 5: Insertion and deletion calling performance from CEPH trio datasets.

<table>
<thead>
<tr>
<th>SV type</th>
<th>Sample</th>
<th>Number (%) of Mak et al. calls detected</th>
<th>Number (%) of calls that were new</th>
</tr>
</thead>
<tbody>
<tr>
<td>Deletions</td>
<td>NA12878</td>
<td>386 (93%)</td>
<td>794 (74%)</td>
</tr>
<tr>
<td></td>
<td>NA12891</td>
<td>337 (81%)</td>
<td>802 (78%)</td>
</tr>
<tr>
<td></td>
<td>NA12892</td>
<td>310 (80%)</td>
<td>721 (78%)</td>
</tr>
<tr>
<td>Insertions</td>
<td>NA12878</td>
<td>382 (89%)</td>
<td>2089 (88%)</td>
</tr>
<tr>
<td></td>
<td>NA12891</td>
<td>355 (81%)</td>
<td>1949 (88%)</td>
</tr>
<tr>
<td></td>
<td>NA12892</td>
<td>336 (78%)</td>
<td>2011 (89%)</td>
</tr>
</tbody>
</table>
Performance for translocation breakpoints

Nine samples with annotated single translocations were tested for translocation calling performance (Table 6). 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 7).

Table 6. Translocation calling performance from real datasets (at close to or less than 70X effective coverage).

<table>
<thead>
<tr>
<th>Sample</th>
<th>Diagnosis</th>
<th>Annotation</th>
<th>Sample prep</th>
<th>Found expected breakpoint?</th>
<th>Found reciprocal breakpoints?</th>
</tr>
</thead>
<tbody>
<tr>
<td>GM16736</td>
<td>Deafness with DNA repair deficiency</td>
<td>FISH: t(9;22)</td>
<td>OptiDNA SVMerge and plug lysis BspQI</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>GM21891</td>
<td>Prader-Willi syndrome</td>
<td>FISH: t(4;15)</td>
<td>OptiDNA SVMerge and plug lysis BspQI</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Sample01</td>
<td>CML</td>
<td>FISH: t(9;22)</td>
<td>Plug lysis BspQI</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Sample02</td>
<td>CLL</td>
<td>FISH: t(11;14)</td>
<td>Plug lysis BspQI</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Sample03</td>
<td>CML</td>
<td>FISH: t(9;22)</td>
<td>Plug lysis BspQI</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Sample04</td>
<td>CML</td>
<td>Kary.: t(9;22)</td>
<td>Plug lysis BspQI and BssSI</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Sample05</td>
<td>AML</td>
<td>Kary.: t(7;11)</td>
<td>Plug lysis BspQI</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Sample06</td>
<td>AML</td>
<td>Kary.: t(8;21)</td>
<td>Plug lysis BspQI</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Sample07</td>
<td>AML</td>
<td>Kary.: t(4;5)</td>
<td>Plug lysis BspQI and BssSI</td>
<td>No</td>
<td>N/A</td>
</tr>
</tbody>
</table>

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 70X effective coverage assemblies, we found the expected translocation breakpoints for both samples and additional translocation breakpoints. 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 7. Translocation breakpoint call masking and filtering for Coriell samples GM16736 and GM21891.

<table>
<thead>
<tr>
<th>#Translocation calls</th>
<th>Coriell (BspQI)</th>
<th>Coriell (BspSI)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Known Translocation</td>
<td>GM16736</td>
<td>GM21891</td>
</tr>
<tr>
<td>Remaining calls after masking</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>As expected?</td>
<td>Yes</td>
<td>1 add. call</td>
</tr>
<tr>
<td>As expected after applying confidence threshold &gt;= 0.1</td>
<td>Yes</td>
<td>Yes</td>
</tr>
</tbody>
</table>

hg19

hg38

Performance for inversion breakpoints

Analysis of samples with known events

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 inversions events in all three cell lines were observed at the annotated locations (Table 8).

Table 8. Inversion calling performance from real datasets (at close to or less than 70X effective coverage).

<table>
<thead>
<tr>
<th>Sample</th>
<th>Diagnosis</th>
<th>Annotation</th>
<th>Sample prep</th>
<th>Found expected inversion?</th>
</tr>
</thead>
<tbody>
<tr>
<td>GM19238</td>
<td>Phenotypically normal</td>
<td>Inv(15q13.3)</td>
<td>Plug lysis BspQI</td>
<td>Yes</td>
</tr>
<tr>
<td>GM14266</td>
<td>Micrognathia</td>
<td>Inv(4q34.2-35.2)</td>
<td>OptiDNA SVMerge</td>
<td>Yes*</td>
</tr>
<tr>
<td>GM21074A</td>
<td>Developmental delay</td>
<td>Inv(2p23-q31)</td>
<td>OptiDNA SVMerge</td>
<td>Yes*</td>
</tr>
</tbody>
</table>

*Called as intra-chromosomal translocation breakpoints, since those inversions were larger than 5 Mbp.

Comparison to Valor calls

For the NA12878 sample, we compared Bionano calls to inversion calls predicted by a new NGS based algorithm. Recently, Rasekh et. al.² described a novel algorithm, Valor, to detect large inversions based on “split clone” signatures. Out of the 18 inversion calls predicted by Valor, 15 overlapped with Bionano calls. Bionano detected an additional 82 putative inversion regions.

Bionano Access integration

With Bionano Access, users are able to select the set of assembly parameters for *de novo* assembly, depending on the application (Table 9). Three of the eight XML files (colored in red) are packaged with Bionano Solve but are not shown in Access by default. They are not recommended for general use but are made available for advanced users. A flow diagram (Figure 8) is provided to assist selection of the recommended XML files.

Bionano Access enables import, filtering, and visualization of SV calls (Figure 1). The SV Table in Bionano Access contains single-enzyme detection and SVMerge output with corresponding confidence scores. The Bionano Access visualization software can create views of the corresponding genome map to reference alignment from the two input assemblies. Additionally, it produces a simplified assembly summary report.

Users can import a mask file to annotate insertion and deletion calls and filter out likely false positive translocation calls. We provide four versions of the mask to support the (hg19 and hg38) human reference assemblies and the enzymes recommended for human analysis (BspQI and BssSI). Users can also import custom gap files. By interacting with the SV Table, users may show or hide masked calls.

---

Figure 8. Flow diagram for selection of assembly parameters XML.
<table>
<thead>
<tr>
<th>Parameters XML</th>
<th>Argument name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>optArguments_haplotype_saphyr_human.xml</td>
<td>Saphyr, human, SV calling</td>
<td>Generate an assembly for calling homozygous and heterozygous SVs from human datasets collected on a Saphyr. Haplotype-aware refinement and extension is enabled to improve the assembly and separation of alleles.</td>
</tr>
<tr>
<td>optArguments_nonhaplotype_saphyr_human.xml</td>
<td>Saphyr, human, Hybrid Scaffolding</td>
<td>Generate an assembly for building hybrid scaffolds from human datasets collected on a Saphyr. Haplotype-aware refinement is disabled but haplotype-aware extension is enabled to better handle large allelelic differences.</td>
</tr>
<tr>
<td>optArguments_haplotype_saphyr.xml</td>
<td>Saphyr, non-human, SV calling</td>
<td>Generate an assembly for calling homozygous and heterozygous SVs from non-human datasets collected on a Saphyr. Haplotype-aware refinement and extension is enabled to improve the assembly and separation of alleles. Please note that haplotype-aware refinement on non-human datasets is not a supported feature and its use may have unintended consequences. Any SV results for non-human genomes need to be thoroughly validated.</td>
</tr>
<tr>
<td>optArguments_nonhaplotype_saphyr.xml</td>
<td>Saphyr, non-human, Hybrid Scaffolding /SV calling, standard</td>
<td>Generate an assembly for building hybrid scaffolds and calling SVs (if a reference is supplied) from non-human datasets collected on a Saphyr. Haplotype-aware refinement is disabled but haplotype-aware extension is enabled to better handle large allelelic differences. Please note that any SV results for non-human genomes need to be thoroughly validated.</td>
</tr>
<tr>
<td>optArguments_nonhaplotype_noES_saphyr.xml</td>
<td>Saphyr, non-human, Hybrid Scaffolding, fast</td>
<td>Generate an assembly for building hybrid scaffolds from non-human datasets collected on a Saphyr. Haplotype-aware components are disabled to maximize runtime performance. Use only if runtime is a critical consideration and the resulting assembly quality is deemed sufficient for the targeted genome.</td>
</tr>
<tr>
<td>optArguments_haplotype_irys.xml</td>
<td>Irys, SV calling</td>
<td>Generate an assembly for calling homozygous and heterozygous SVs from human and non-human datasets collected on an Irys. Haplotype-aware refinement and extension is enabled to improve the assembly and separation of alleles. Please note that any SV results for non-human genomes need to be thoroughly validated.</td>
</tr>
<tr>
<td>optArguments_nonhaplotype_irys.xml</td>
<td>Irys, Hybrid Scaffolding, standard</td>
<td>Generate an assembly for building hybrid scaffolds from human and non-human datasets collected on an Irys. Haplotype-aware refinement is disabled but haplotype-aware extension is enabled to better handle large allelelic differences.</td>
</tr>
<tr>
<td>optArguments_nonhaplotype_noES_irys.xml</td>
<td>Irys, Hybrid Scaffolding, fast</td>
<td>Generate an assembly for building hybrid scaffolds from non-human datasets collected on an Irys. Haplotype-aware components are disabled to maximize runtime performance. Use only if runtime is a critical consideration and the resulting assembly quality is deemed sufficient for the targeted genome. It is highly recommended that human datasets be assembled with haplotype-aware extension enabled.</td>
</tr>
</tbody>
</table>
SMAP-to-VCF converter support

The python-based SMAP-to-VCF converter supports conversion of insertion, deletion, translocation breakpoint, and inversion breakpoint calls. The resulting VCF output is dbVar-compliant. The SV breakpoints are output as the midpoint between the last aligned label and the next label. The uncertain breakpoint is indicated in the CIEND and CIPOS fields.

Consistent with the VCF v4.2 format, the VCF file output passed preliminary testing with vcf-tools/vcf-validator, and the VCF headers have been updated accordingly. The information that is added to the output includes the reference accession, orientation information for translocations, and additional dbVar-specific tags. A separate SVMerge-to-VCF converter is provided on Bionano’s GitHub site.

Additional Considerations

SV sensitivity improves with depth of coverage. The minimum recommended effective coverage (defined as the product of raw coverage and molecule alignment rate) is 70X, assuming that the quality and average length of the input molecules are good. More coverage will yield small additional sensitivity with diminishing returns beyond 100X.

In order to resolve the alleles, haplotype-aware assemblies require more processing time, relative to non-haplotype-aware assemblies. With typical good-quality molecules, one should expect 2X to 2.5X increase in processing time.

The Bionano Solve pipeline was validated on human diploid samples. When running more complex samples (e.g., cancer) or non-human samples with the haplotype-aware assembler (with haplotype-aware assembly parameters selected), we recommend that the SV calls be validated.
FAQs

1. How does coverage affect SV calling performance?

Based on consideration of SV calling performance, runtime, and system throughput, we recommend a minimum of 70X 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 40X effective coverage). Although Bionano Solve is sensitive to smaller insertions and deletions, we recommend that users focus on insertions and deletions larger than 1 kbp. Translocation breakpoint calling performance appears to plateau at or around 70X coverage; we expect minimal gain with higher coverage levels.

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

Based on visual inspection, we noticed cases where there was room for improvement in the precision 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.

3. 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. Custom secondary analysis may aid pairing and classification of translocation breakpoints.

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

Cluster parameters and assembly parameters files have been moved and renamed. Different versions were created for supporting different platforms. Cluster parameter files are now distributed with the python assembly pipeline. There are separate cluster parameters for IrysSolve and Bionano Compute servers. Assembly parameter files are now distributed with RefAligner. There are separate assembly parameters recommended for Irys and Saphyr datasets. For human assemblies, we recommend label SNR-thresholding based on alignment to the reference. For non-human assemblies, we recommend histogram-based SNR-thresholding. Both methods are implemented as part of the Bionano Access workflow.

5. What is the probability of detecting reciprocal translocation breakpoints?

We have 90.7%, 83.0%, and 97.9% sensitivity for translocations using single-enzyme BspQI, BssSI, and SVMerge data, respectively. If we considered the two versions of a reciprocal translocation to be independent, the probability of detecting both with single-enzyme BspQI data would be (90.7%)² = 82.4%. The probability of detecting at least one would be 1-(1-90.7%)² = 99.1%.

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

Large segmental duplication regions by definition 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. The assembler is designed to output all possible combinations if the molecule data cannot definitively differentiate between possible combinations.
7. 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.

8. How to create a custom mask?

The masks are in BED format (https://genome.ucsc.edu/FAQ/FAQformat#format1). They include three components: 1) N-base gaps, 2) annotated segmental duplication regions, and 3) translocation breakpoints detected in genomes not known to contain translocations. The list of N-base gaps can be obtained from analyzing a given reference fasta file and outputting locations of the N bases. The list of annotated segmental duplication regions for human was obtained from the UCSC Genome Browser database. 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.

9. 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, or how likely a given call is a TP. Confidence scores for translocation and inversion breakpoints are computed as p-values, indicating how likely a given call is different (in Mahalanobis distance) from known TP calls. They require different thresholds.

10. What types of SV calls does SVMerge merge?

SVMerge currently supports merging of insertion, deletion, translocation breakpoint, and inversion breakpoint calls. SVMerge does not merge duplication calls but does carry forward the single-enzyme calls in the output.

11. What does SVMerge do if a call from one assembly can be merged with more than one call from another assembly?

SVMerge attempts to merge all calls that satisfy the merging criteria. For example, if a call from the BspQI assembly could be merged with two separate calls from the BssSI assembly, there would be two merged calls in the output involving the same BspQI call. Also, if there are duplicate single-enzyme calls, SVMerge may produce additional duplicate merged calls.

12. Why are certain SV calls not merged by SVMerge, even though they appear to be overlapping?

SVMerge attempts to make a determination of whether two calls are "similar" enough to be merged, and there are SV type specific criteria. For example, for insertions and deletions, SVMerge requires that there be both overlap and size concordance between two calls. For inversions, the boundaries and the difference in the inversion ranges are required to be within a certain cutoff (20 kbp by default).

13. Does the SMAP-to-VCF converter support conversion of SVMerge output?

A separate SVMerge-to-VCF converter is available at Github. The SMAP-to-VCF converter expects that the input be in the standard SMAP format.
14. What is the expected runtime performance?

Below are typical representative runtime data for IrysSolve and Saphyr Compute running the haplotype-aware pipeline through to SV calling.

<table>
<thead>
<tr>
<th>Enzyme</th>
<th>Raw coverage</th>
<th>Effective coverage</th>
<th>IrysSolve Compute</th>
<th>Saphyr Compute</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sample 1</td>
<td>BspQI</td>
<td>91X</td>
<td>68X</td>
<td>20.1 hr</td>
</tr>
<tr>
<td>Sample 2</td>
<td>BssSI</td>
<td>81X</td>
<td>61X</td>
<td>22.4 hr</td>
</tr>
</tbody>
</table>

15. Why do I see fewer maps in regions where I’ve 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.

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

Yes, we recommend that SV detection be re-run on the command line using the new Solve tools in order to take advantage of improved inversion and duplication sensitivity. After SV detection, one would replace the existing SV output directory with the new SV output directory, compress the entire assembly output, and import into Access for SV visualization and downstream analysis.

The computation of the inversion and translocation breakpoint confidence relies on assembly quality scores that are only available in assemblies generated with 2.0 or later releases of the Solve pipeline. If assembly quality scores were not available, “-1” would be output for inversion and translocation breakpoint confidence.
Appendix

Method 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 with effective coverage levels from 87X-110X were generated. The simulated molecules were used as input for the haplotype-aware assembly pipeline, and SV calls were compared to the ground truth.

Simulation of translocations

Random translocation events were simulated to form an edited genome (Figure 9). 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 translocations, 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 9. 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. Simulated molecules were used as input for de novo assembly and SV detection.

SV calling performance with experimental data

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. We also downsampled the dataset to different coverage levels (50X and 70X). 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. We reported SV calling performance based on the 70X dataset (Table1).

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.
Mask generation

Compiling common translocation breakpoint list

152 genome map assemblies for control human samples for hg19 BspQI (151 for hg38 BspQI, 17 for hg19 Bss SI, and 17 for hg38 Bss SI) were used as input. We assumed that control samples did not contain translocations. We performed SV detection on each assembly, and detected translocation breakpoints were collected.

If a translocation breakpoint was detected in at least four assemblies in BspQI (two assemblies in Bss SI), it would be flagged as a putative FP breakpoint. Two neighboring FP 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 1-4, the segmental duplication region would be included in the mask and labeled as “segdupe”.