User Guide for GASV Software Package
GASV (Version 2.0) and GASVPro (Version 1.0)

August 3, 2012

Contents

1 Introduction

GASV and GASVPro are software to identify structural variants from paired-end mapping data.

If you use GASV in your research, please cite:

S. Sindi, E. Helman, A. Bashir, B.J. Raphael. A Geometric Approach for Classification and Comparison of Structural Variants. Joint: 17th Annual International Conference on Intelligent Systems in Molecular Biology and 8th Annual International European Conference on Computational Biology (ISMB/ECCB 09). Bioinformatics 2009 25(12):i222-i230. doi:10.1093/bioinformatics/btp208

This product includes software developed by the Solution Engineering, Inc. (`http://www.seisw.com/`).

Contact: Benjamin Raphael at `gasv@cs.brown.edu`

Version: 2.0
Version Date: May 30 2012

GASVPro is a probabilistic version of the GASV algorithm, combining read depth information with discordant paired-read mappings into a single model. It is comprised of three software programs: GASVPro-CC, GASVPro-graph, and GASVPro-mcmc (see Section 6).

If you use GASVPro in your research, please cite:

S. Sindi, S. Onal, L. Peng, H. Wu, B.J. Raphael. An Integrative Probabilistic Model for Identification of Structural Variation in Sequencing Data. Genome Biology 2012 13(3):R22. doi:10.1186/gb-2012-13-3-r22.

This product contains software developed by Erik Garrison (`http://github.com/ekg`).

Version: 1.0
Version Date: August 3, 2012

1.1 Requirements
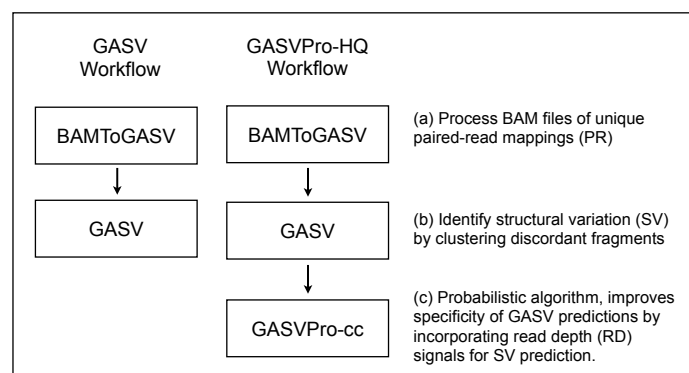
Requires: Perl, Java 5 (or higher), *nix OS,
Tested with: Perl v5.10.0, Java 6, Linux 2.6.26

Building GASV from the source requires ant, available at: http://ant.apache.org/

1.2 Summary

The GASV software utilizes a geometric algorithm to predict structural varia-

Figure 1: **GASV and GASVPro-HQ Workflow**

tion (SV) from a set of discordant mappings of paired-reads (PRs). GASVPro is a probabilistic algorithm consisting of two pipelines: GASVPro-HQ (HQ = High-Quality) combines depth of coverage information to improve specificity of GASV predictions; GASVPro in addition considers multiple possible read alignments and utilizes an MCMC algorithm to sample over the space of alignments.

1.3 GASV Workflow

The primary GASV workflow is an analysis pipeline for SV detection from a set of paired-read mappings (PR) given in the form of a BAM file. In most workflows the BAM file is pre-processed for analysis with GASV using BAMToGASV (Figure 1(a)) followed by analysis with GASV (Figure 1(b)) to determine a set of potential SVs genome wide. Further downstream analysis of GASV predictions may be conducted with GASVPro, a probabilistic algorithm which integrates read-depth (RD) to improve specificity by eliminating false-positive predictions.

1.4 GASV Algorithm

The GASV algorithm reports structural variant events defined by clusters of discordantly mapped fragments and explicitly represents uncertainty variant endpoints. We briefly overview the approach used in GASV; for a more detailed discussion of the GASV algorithm, refer to Sindi et al. (2009).

A discordant mapping indicates a structural variant in the test genome defined by a novel adjacency (a, b), where positions a and b are adjacent in the test genome, but not in the reference genome (see Figure 2). A single fragment alone does not uniquely specify the pair of breakends (a, b) defining the rearrangement, but
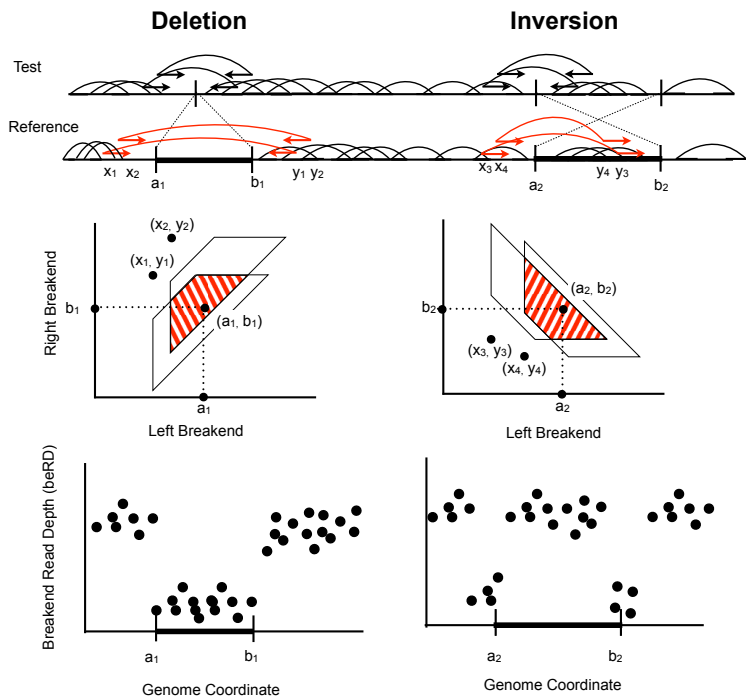
Figure 2: **GASV Algorithm of Structural Variation Detection from Paired-End Sequencing.** (Top) Fragments (black arches) from a test genome are sequenced from both ends and the resulting paired reads are mapped to a reference genome. Fragments containing the breakpoint of a structural variant (black arches with arrows) have a discordant mapping (red). (Middle) GASV represents the uncertainty in breakpoint location as a breakend polygon (black trapezoids). GASV clusters fragments by computing polygon intersections (shaded red trapezoid) which correspond to fragments supporting the same variant. (Bottom) Combining read depth (RD) with GASV polygons improves detection of variant breakends.

4

rather defines uncertainty in the loca-
tion of the breakends. Formally, if we assume that a discordant fragment corresponds to exactly
one structural variant, then the mapped locations, x and y, of the fragment endpoints (without
loss of generality we restrict $x < y$), and the breakends a and b satisfy

$$L_{\min} \le \text{sign}(x)(a - x) + \text{sign}(y)(b - y) \le L_{\max},$$

where $\text{sign}(x)$ and $\text{sign}(y)$ are 1 if the reads align to the positive strand and have convergent ori-
entation and -1 otherwise. Here we assume convergent orientation is when reads have opposite
orientation with the left read forward and the right read reversed as in the case for Illumina se-
quencing technology. (Other sequencing protocols will may have different notions of convergent
orientation.) The inequality (1) defines a trapezoid in the plane; discordant fragments correspond-
ing to the same structural variant will have overlapping trapezoids and their intersection can be
used to further refine the uncertainty in breakend location as in Figure 2. The final set of SV
predictions from GASV corresponds to these polygonal intersections.

1.5 Installation

You may either download executable jar files from

`http://code.google.com/p/gasv/downloads/`

or build our code directly from the source.

Building GASV from the source requires ant, available at: `http://ant.apache.org/`. To install
from the source check out a read-only working copy from

```
$ svn checkout http://gasv.googlecode.com/svn/trunk/ gasv-read-only
```

and install from the command line with the following command:

```
$ ./install
```

During installation the java `*.class` files for BAMToGASV and GASV are built into the `/build`
subdirectory. After installation the following executable files are created:

```
bin/GASV.jar
bin/BAMToGASV.jar
bin/GASVPro-CC
bin/GASVPro-graph
bin/GASVPro-mcmc
```

2 Quickstart Guide to GASV

This Quickstart Guide will demonstrate many features of GASV including determining the struc-
tural variants from paired-read sequences (PRs) whose mappings are encoded in a BAM file.

To demonstrate using BAMToGASV preprocessor in conjunction with GASV, we will use the
file "Example.bam" available as a separate download from the GASV website:

```
http://code.google.com/p/gasv
```

More documentation is available for both GASV and BAMToGASV Preprocessor in sections below.

2.1 Using BAMToGASV to Preprocess BAM files

The following command creates GASV input files by considering paired-eads (PRs) from the BAM file "Example.bam" (this particular BAM file contains no library information and contains only reads from chr17, so no translocations):

```
$ java -Xms512m -Xmx2048m -jar BAMToGASV.jar Example.bam
```

As we detail further below, there are many options available for BAMToGASV. For example, use a minimum quality of 30 and determine Lmin/Lmax by 3 standard deviations away from the mean, the command is

```
$ java -Xms512m -Xmx2048m -jar BAMToGASV.jar Example.bam -MAPPING_QUALITY 30 -CUTOFF_LMINLMAX SD=3
```

If the BAM file contains multiple libraries, to process them as one set (rather than indvididually) the command is

```
$ java -Xms512m -Xmx2048m -jar BAMToGASV.jar Example.bam -LIBRARY_SEPARATED all
```

Running the above command creates the following files:

Example.bam.info
Example.bam.gasv.in
Example.bam_all.deletion
Example.bam_all.divergent
Example.bam_all.insertion
Example.bam_all.inversion
Example.bam_all.translocation

Example.info contains information about the concordant fragment length distribution. Example.gasv.in is a batch file of all PR files (except insertions) for input to the GASV program. The rest of the files are PR files in the GASV input format, separated by variant type.

2.2 Run GASV to Compute Structural Variants

The following command will report inversion, deletion and divergent structural variants using the batch input file produced by BAMToGASV. Note the default behavior of GASV is to only report variants with at least 4 supporting fragments. In addition, if GASV.jar has not been added to the PATH, then a fully qualified pathname must be used.

```
$ java -jar GASV.jar --batch Example.bam.gasv.in
```

Below is the contents of the *.clusters file created, Example.bam.gasv.in.clusters:

```
#Cluster_ID: LeftChr: LeftBreakPoint:       RightChr: RightBreakPoint:      Num PRS: Localization: Type:
c99          17       660077,660155          17        660177,660255         4        39.8          D
c106         17       685625,685815          17        685996,686186         4        100.6         D
c4552        17       34908617,34908798      17        34908870,34909051     5        68.4          D
c5262        17       40566182,40566317      17        40567335,40567462     4        68.3          IR
c6367        17       50617507,50617679      17        50617548,50617720     4        87.8          D
```

The tab separated fields indicate that there are 5 predictions, 4 deletions and 1 inversion structural variant on chromosome 17. Examining the fourth line more carefully, cluster c5262 supports an inversion with left breakpoint occurs between positions 40566182, 40566316 and the right breakpoint between positions 40567335, 40567462.

In addition, the final columns tell us:

- 4 fragments support this SV (Num PRs = 4)

- The breakpoint localization is 68. The smaller the localization the more precisely the structural variant breakpoint is determined. Refer below and to Sindi, et al. (2009) for more discussion of localization.

- This variant is a reciprocal inversion (Type = IR) Among the 4 PRs at least one is in the ++ orientation (contains the left breakpoint) and one in the -- orientation (contains the right breakpoint).

2.3   Useful GASV Examples

1. --minClusterSize <val> Report only predictions with at least a minimum number of supporting fragments. (By default, this parameter is set to 4.) For increased sensitivity, users can lower the threshold. For example to obtain all predictions supported by at least 2 mappings:

```
$ java -jar GASV.jar --minClusterSize 2 --batch Example.bam.gasv.in
```

2. --maximal In some cases a set of overlapping fragments may correspond to multiple breakpoints. The "--maximal" option will output all maximal breakpoint events.

3. --maxPairedEndsPerWin <val> The speed at which GASV runs corresponds to the number of fragments in a particular genomic region. Often some genomic regions (such as centromeres) will have a high density of discordant fragments. GASV's running time can be significantly increased by skipping such high density regions with the "--maxPairedEndsPerWin" option.

4. --output <val> As we detail below, there are three different output formats that can be specified for *.clusters file. The reads format, "--output reads" will list the names of the reads in a cluster.

```
$ java -jar GASV.jar --output reads --batch Example.bam.gasv.in
```

3 Running the GASVPro Pipeline with Scripts

To aid users in running the GASVPro pipeline, we have provided paramaterized scripts for GASVPro-HQ and GASVPro. These scripts are stored in the \bin directory and are designed for streamlined use. They take as input a .bam file (or files), plus other optional parameters, and run the respective pipeline (Figure 1 and Figure 3).

Most users should begin with the GASVPro-HQ script. However, our scripts can be easily modified to fit a desired workflow and users are encouraged to adapt these scripts or develop their own.

3.1 GASVPro-HQ (High Quality)

For data sets with only unique read mappings, we have created a script called `GASVPro-HQ.sh`. To use this script, change the top section of the script file to match desired parameters. The parameters appear as follows:

```
#REQUIRED:
BAMFILE=example.bam
GASVDIR=/home/gasv

#OPTIONAL (set to NULL if not being used):
UNIQUEFILE=NULL
MAXUNIQUEVAL=NULL                     #MUST SPECIFY IF UNIQUEFILE IS GIVEN
MINSCALEDUNIQUE=NULL                  #MUST SPECIFY IF UNIQUEFILE IS GIVEN
LRTHRESHOLD=NULL                      #default 0
MINCLUSTER=NULL                       #default 4
```

- **BAMFILE:** The .bam file that will go through the pipeline. It is required.

- **GASVDIR:** The directory that contains all of the gasv files.

- **UNIQUEFILE:** The file containing a uniqueness track, used in GASVPro-CC. It is optional.

- **MAXUNIQUEVAL:** The highest numerical uniqueness value given to a fragment in the unique file. REQUIRED if a unique file is given.

- **MINSCALEDUNIQUE:** The minimum value to which the uniqueness values can be scaled. REQUIRED if a unique file is given

- **LRTHRESHOLD:** The minimum liklihood threshold past which variants will be reported from GASVPro-CC. It is optional.

- **MINCLUSTER:** The minimum cluster size. It is optional.

After the desired parameters are chosen, be sure to save the file. Since this file needs to be executable, run

```
chmod 777 GASVPro-HQ.sh
```

Now it is possible to run GASVPro-HQ.sh. Initiate the pipeline by running

```
./GASVPro-HQ.sh
```

3.1.1 Output of GASVPro-HQ.sh

GASVPro-HQ.sh produces quite a few output files, only a few of which are the "final products" of the entire pipeline. The following files are produced by running GASVPro-HQ:

```
date_time.gasv
date_time.gasv.in.clusters
date_time.gasv.in.clusters.GASVPro.clusters
date_time.gasv.in.clusters.GASVPro.clusters.pruned.clusters
date_time.gasv.in.clusters.GASVPro.coverage
date_time.gasvpro.in
date_time.info
date_time_all.concordant
date_time_all.deletion
date_time_all.divergent
date_time_all.insertion
date_time_all.translocation
```

The final output of the pipeline is *date_time*.`gasv.in.clusters.GASVPro.clusters`, containing SV clusters in regions format, *date_time*.`gasv.in.clusters.GASVPro.clusters.pruned.clusters`, containing a pruned subset of those clusters, and *date_time*.`gasv.in.clusters.GASVPro.coverage`, containing extra information about each cluster (such as concordant coverage) from GASVPro-cc.

3.2 GASVPro

For users with both a high-quality BAM file of unique mappings and a BAM file of lower-quality and possibly multiple mappings. The GASVPro pipeline will incorporate both types of data.

We have created a script called `GASVPro.sh` which will run this pipeline completely given two BAM files. As above, it is necessary to change the options at the top of the script file to match the desired parameters. The `GASVPro.sh` parameters appear as follows:

```
#REQUIRED:
BAMFILEHQ=/path/to/HQ.bam
BAMFILELQ=/path/to/Ambiguous.bam
WORKINGDIR=/FULL/path/to/output/directory   #GIVE FULL PATH!
GASVDIR=/path/to/gasv/

#OPTIONAL (set to NULL if not being used):
UNIQUEFILE=NULL
MAXUNIQUEVAL=NULL               #MUST SPECIFY IF UNIQUEFILE IS GIVEN
MINSCALEDUNIQUE=NULL            #MUST SPECIFY IF UNIQUEFILE IS GIVEN
LRTHRESHOLD=NULL    #default 0
MINCLUSTER=1        #default 4
MAXIMAL=FALSE       #use GASV's --maximal flag. (use TRUE or FALSE)
```

After the desired parameters are chosen, be sure to save the file. To ensure the file is executable, run

GASVPro Workflow

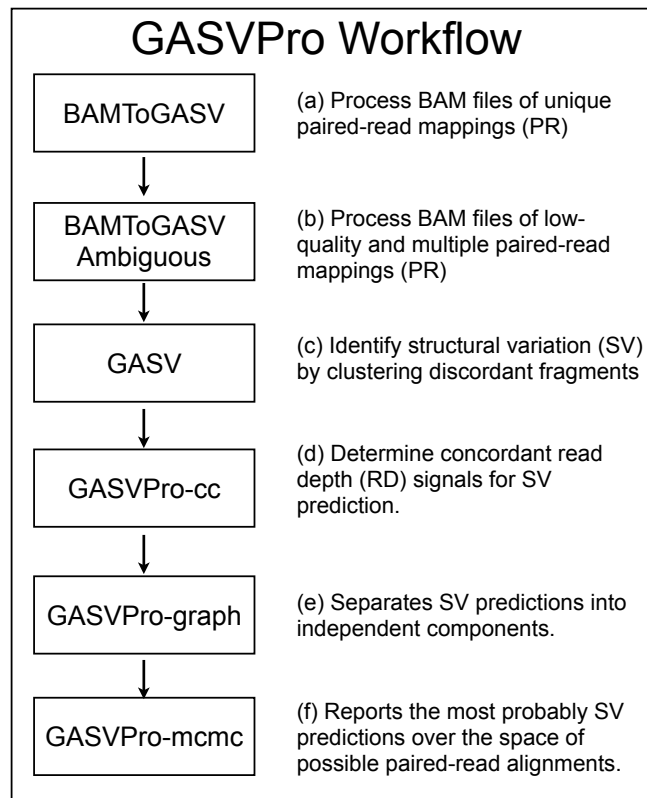|	
BAMToGASV	(a) Process BAM files of unique paired-read mappings (PR)
BAMToGASV Ambiguous	(b) Process BAM files of low-quality and multiple paired-read mappings (PR)
GASV	(c) Identify structural variation (SV) by clustering discordant fragments
GASVPro-cc	(d) Determine concordant read depth (RD) signals for SV prediction.
GASVPro-graph	(e) Separates SV predictions into independent components.
GASVPro-mcmc	(f) Reports the most probably SV predictions over the space of possible paired-read alignments.

Figure 3: **GASVPro Workflow** (Refer to Figure 1 for GASV and GASVPro-HQ workflow.)

```
chmod 777 GASVPro-HQ.sh
```

Now it is possible to run GASVPro.sh. Initiate the pipeline by running

```
./GASVPro.sh
```

3.2.1   Output of GASVPro.sh

The output of GASVPro.sh includes all of the output of GASVPro-HQ.sh (*one set of SV files for each of the HQ and LQ BAMs*, with combined gasv and gasvpro clusters files), plus the following additional files contained within the specified output directory:

```
sv_*.sv (many files)
BAMToGASV.gasvpro.in_sv_*.MCMCThreshold.clusters (many files)
BAMToGASV.gasvpro.in_sv_*.PR.results (many files)
BAMToGASV.gasvpro.in_sv_*.Variant.results (many files)
p_star.summary
svFileList.summary
```

These files are the final output of the GASVPro.sh pipeline. The final set of predictions is given in the (FINAL FILE). See the GASVPro-graph and GASVPro-mcmc sections (Sections **6.4** and **6.5**, respectively) for details.

4   BAMToGASV

BAMToGASV generates GASV formatted input files from the SAM/BAM alignment format. SAM files are also accepted as input, but for clarity this manual will refer to BAM files only. Picard, a Java tool for processing BAM files, is packaged in the JAR file. Running BAMToGASV with no arguments produces basic usage information.

4.1   Requirements

BAMToGASV is currently designed to process BAM files where each read has at most one reported mapping.

Java 6 is required to run BAMtoGASV. We suggest initializing the Java heap size using the `-Xms512m -Xmx2048m` options, which requires running on a machine with more than 2G of memory. The BAM file must containing pairing information in the flags - if it does not, then Picard's `fixmate` function is called to make a new BAM file with pairing information and then the program exits.

4.2   Command Line Options

```
java -Xms512m -Xmx2048m -jar BAMToGASV.jar <BAMFile> [Options]
```

-LIBRARY_SEPARATED <**String**> (Default: sep)
    Determines whether or not to consider multiple libraries separately

sep Produce a set of output files for each library.

all Produce a single set of output files for all libraries.

If no libary information is provided in the BAM file header, it will default to the 'all' setting. In the case where no library info is found or 'all' is passed as -LIBRARY_SEPARATED, the word "all" is used as the LIBRARY_ID in the output file names.

-OUTPUT_PREFIX <String> (Default: BAM filename)
The prefix for your output files.

-MAPPING_QUALITY <Integer> (Default: 10)
Mapping quality threshold for reads.

-CUTOFF_LMINLMAX <String> (Default: PCT=99%)
Specifies lower and upper bounds on the fragment distribution.

PCT=X% Take the quantile on the top/bottom X percent.

SD=X Take the standard deviation above/below the mean.

EXACT=X,Y Set Lmin to X and Lmax to Y.

FILE=fname File of the form '<libname> <CUTOFF_LMINLMAX>' for using different
cutoffs for individual libraries.
EXAMPLE:
library_id1 EXACT=123,456
library_id2 SD=3
library_id3 PCT=90%

In the case that the computed Lmax is smaller than twice the read length (which is the minimum fragment length), Lmax is reset to be twice the read length.

-USE_NUMBER_READS <Integer> (Default: 500000)
The number of fragments in the BAM file to use in computing Lmax and Lmin. Note that a relatively small number of fragments will be sufficient to get a good estimate of Lmin and Lmax. Depending on the size of the BAM file and running time constraints between 500,000 or 1,000,000 would be sufficient for most applications. If EXACT=X,Y is specified for the CUTOFF_LMINLMAX argument, no fragments will be used, and USE_NUMBER_READS argument is ignored.

-CHROMOSOME_NAMING_FILE <String> (Default: none)
File of the form '<ChrName>\t<Integer>' for specifying non-default chromosome namings. This is an optional parameter, you do not need this option if you are using the default chromosome naming references (a number, X, Y, chr+a number, chrX, and chrY). The second column contains unique integer IDs for each non-default chromosome name.
EXAMPLE:
Ca21chr1 1
Ca21chr2 2
Ca21-mtDNA 9

-PROPER_LENGTH <Integer> (Default: 10000)
Ignore PRs with separation larger than PROPER_LENGTH when calculating Lmin and

Lmax. Extreme outliers (PRs with mapped length >10Mb) can cause huge standard deviation values which will produce large values for Lmax in with the SD option. If you use EXACT=X,Y or PCT=X% as CUTOFF_LMINLMAX, this option will not be considered.

-PLATFORM <String> (Default: Illumina)

Paired Illumina reads are sequenced from different strands of the fragment, while paired SOLiD reads are sequenced from the same strand of the fragment.

Illumina Reads are sequenced with an Illumina-like platform.

SOLiD Reads are sequenced with a SOLiD-like platform.

If the BAM header information reports a platform different from the -PLATFORM option, a warning is emitted but the program continues with the -PLATFORM option.

-WRITE_CONCORDANT <Boolean> (Default: False)

Writes a file of concordant PRs of the form '<chr> <start> <end>' to an OUTPUT_PREFIX.LIBRARY_ID.concordant file. WARNING - this may be a very large file. In a future release this will be a compressed binary file.

-WRITE_LOWQ <Boolean> (Default: False)

Writes a file of low-quality pairs (pairs where at least on read has quality below -MAPPING_QUALITY). WARNING - this may be a very large file.

-VALIDATION_STRINGENCY <String> (Default: silent)

Picard performs internal testing of BAM records as they are read into the BAMtoGASV program. The VALIDATION_STRINGENCY option determines how stringent this internal testing is.

silent Read SAM records without any validation.

lenient Read SAM records and emit a warning when a record is not formatted properly.

strict Read SAM records and die when a record is not formatted properly.

Picard's default setting is Strict; however, this assumes a very well-formatted BAM file from Illumina sequencing. If the output files are empty or not what you expect, we suggest running with VALIDATION_STRINGENCY set to lenient or strict to determine improperly-formatted BAM records.

4.3 Output of BAMToGASV

The output of BAMToGASV is a series of discordant PR files (see description below for GASV), an info file with summary statistics on the various libraries, and a batch input file for GASV.

Discordant fragments are grouped into output files according to the type of structural variant they indicate: deletion, insertion, inversion and translocation. Divergent fragments indicate a inter-chromosomal rearrangement that is not a deletion, inversion, or insertion.

OUTPUT_PREFIX(_LIBRARY_ID).deletion

OUTPUT_PREFIX(_LIBRARY_ID).divergent

OUTPUT_PREFIX(_LIBRARY_ID).insertion

OUTPUT_PREFIX(_LIBRARY_ID).inversion

OUTPUT_PREFIX(_LIBRARY_ID).translocation

OUTPUT_PREFIX.info

OUTPUT_PREFIX.gasv.in

In addition to a GASV formatted input file, GASV requires the minimum/maximum allowable fragment length (Lmin/Lmax) for concordant pairs. The values of Lmin and Lmax are automatically computed by the BAMToGASV. The calculated values of Lmin and Lmax are output to stdout and written to OUTPUT_PREFIX.info. In the Quickstart guide, the contents of Example.bam.info are:

```
LibraryName     Lmin    Lmax
all             153     246
```

The batch input file is written to OUTPUT_PREFIX.gasv.in. It is a GASV input format that is described below. All PR files except insertions are written to this file; GASV does not currently support insertions.

Note: The output produced from BAMToGASV.jar may slightly differ from output produced by earlier versions. Refer to RELEASE_NOTES.txt for more information.

5 GASV

GASV.jar runs the GASV algorithm on a set of discordant fragment mappings and reports a set of SV predictions. GASV.jar can be run directly on the output of BAMToGASV, using a "gasv.in" file:

```
java -jar GASV.jar <InputFile>.gasv.in
```

5.1 GASV Command Line Options

There are many options to GASV which will allow users to customize their own SV analysis pipeline. Running GASV.jar with no arguments produces basic usage information.

```
java -jar GASV.jar

Program: GASV (Geometric Analysis for Structural Variation)
Version: 2.0

Usage:    java -jar GASV.jar [options] <InputFile>

          InputFile: PR file
            java -jar GASV.jar [options] <InputFile>

          InputFile: Batch File
            java -jar GASV.jar [options] --batch <InputFile>

Output:    <InputFile>.clusters
Default:   Reports all SVs supported by at least 4 mappings.
```

--help Outputs detailed information on options to GASV.

--nohead Output file will not have a header.

--outputdir <dir> Directory into which output file will be placed.

--verbose Prints extra info to standard output.

--debug Run program in debug mode

--fast Makes GASV run faster, but risks OutOfMemory errors

--batch Input file(s) are GASV Batch Files (*.in files listing PR files), rather than the PR file(s) themselves. The --lmin and --lmax options are ignored if --batch is specified.

--lmin <int> Specify integer value to use for Lmin variable. Default value used otherwise. Ignored if the --batch option is specified.

--lmax <int> Specify integer value to use for Lmax variable. Default value used otherwise. Ignored if the --batch option is specified.

--minClusterSize <int> Only those clusters with at least <int> supporting PRs are reported.

--maxClusterSize <int> Only those clusters with at most <int> supporting PRs are reported.

--maxCliqueSize <int> For clusters with greater than<int> paired end sequences, calculation of maximal sub clusters is performed. Also only at most <int> PR names will be output per cluster (though numPR column will still reflect real number of PRs). Localization for such clusters is always -1.

--maxPairedEndsPerWin <int> Does not apply if --fast otion used. If more than <int> paired end sequences are found in the current window, any extra paired end sequences are discarded.

--maximal For each connected component without a common intersection (localization reported as -1), the maximal sub-clusters (each with a common intersection) will be reported.

--numChrom <int> Specify number of chromosomes in the genome. Default is 24.

--output <mode> Specify the GASV cluster output mode (see Output File Formats).

1. standard: Default output mode
2. reads: Default mode and names of supporting fragments
3. regions :Detailed mode with region coordinates and fragment names

--nonreciprocal Only PRs with exact matching orientations will be clustered together. E.g. +/+ and -/- inversion PRs will be segregated into separate clusters. (See Figures 4 and 5.)

5.2 Input File Formats

GASV accepts two types of input files; either a set of discordant mappings or a batch file listing multiple files of discordant mappings. If you are using a BAM file, these files will be automatically created with BAMToGASV.jar. Although, users may supply their own files.

5.2.1 Paired-Read Mapping (PR) File Format

Paired-read (PR) files are used for input of mapped paired-end sequences. Each line in the file specifies a paired-mapping. The 9 columns are tab-separated in the following format:

Name: A string with a unique PR identifier.

Left Chromosome: The chromosome where the left read mapped. (Acceptable formats: CHR1, chr1, 1.)

Left Mapped Start: The start coordinate in the reference of the left read mapping.

Left Mapped End: The end coordinate in the reference of the left read mapping.

Left End Orientation: The orientation of the left read. (Acceptable formats: include +/-, Plus/Minus)

Right Chromosome: Chromosome where the right read mapped.

Right Mapped Start: The start coordinate in the reference of the right read mapping.

> **Right Mapped End:** The end coordinate in the reference of the right read mapping.
>
> **Right End Orientation:** The orientation of the right read.

> IMPORTANT PRE-PROCESSING STEP
>
> PR files must be in sorted order prior to running GASV. If your PR files were created with BAMToGASV.jar they will already be sorted. Otherwise before running GASV you should run the sorting script:
>
> Example: ./scripts/sortPR.bash PRFileToSort

Note: Since mapped start and end are in terms of the reference sequence, Start < End. (See Table 1 for example.)

PRA	1	146590306	146591106	+	1	146630713	146631513	+
PRB	2	46590283	46591083	+	1	146630714	146631514	+
PRC	2	46590283	46591083	+	1	156630714	156631514	+
PRD	6	146586311	146587111	+	10	146633908	146634708	+

Table 1: Example PR file format

5.2.2 Batch File Format (*.in files)

The GASV Batch input files provide a list of the PR file(s) to be processed. As mentioned above, each of the PR files must be sorted by chromosome and genomic coordinates. The BAMToGASV program does this automatically. If your data comes from elsewhere, the script ./scripts/sortPR.bash should be used to sort files prior to running GASV.

For PR files, each line gives the name of a PR file, the word "PR" to indicate the file type, and the minimum (Lmin) and maximum (Lmax) fragment length for paired-read mappings in the file. Columns are tab-separated in the following format:

<div align="center">

FILE   PR   Lmin   Lmax

</div>

5.3 Structural Variation Predictions

GASV outputs a clusters file named <InputFile>.clusters that lists all SV predictions with supporting information. There are three possible output formats for clusters files (detailed below). We first describe the different structural variant types reported by GASV, and provide illustrative figures for translocations (Figure 4) and inversions (Figure 5).

5.3.1 Structural Variant Types

GASV Structural Variant Types

D = Deletion

I = Inversion (see Figure 5)

 IR = Reciprocal Inversion(both ++ and --)

 I+ = Inversion (++ side only)

 I- = Inversion (-- side only)

V = Divergent. NOTE: Currently the GASVPro pipeline does not support divergent variants.

T = Translocation; (see below and Figure 4)

 TR = Reciprocal Translocation

 TN = Nonreciprocal

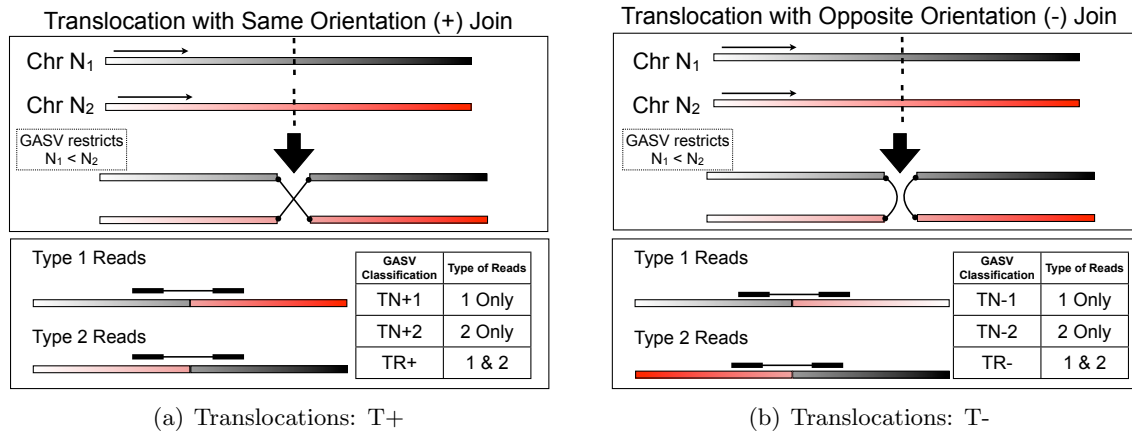
(a) Translocations: T+

(b) Translocations: T-

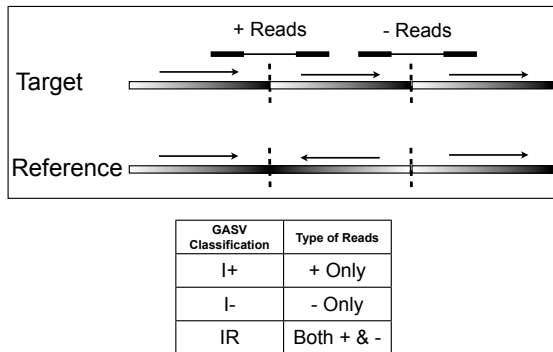Figure 4: **GASV Algorithm Translocation Definition.**

Figure 5: **GASV Algorithm Inversion Definition.**

The GASV algorithm distinguishes between several types of translocations based on the orientations of the respective chromosomes at the join (Figure 4); opposite (-) or same (+); and which resulting chromosome in the test genome was observed; type 1 or type 2. In the top panel of Figure 4, we illustrate the break and join event creating the translocation. While in the bottom panel, we show the resulting chromosomes present in the test genome, from which segments were sampled. As default mode, GASV will attempt to merge predictions from both type 1 and type 2 fragments to identify reciprocal translocation events (TR+ or TR-). If the --noreciprocal

flag has been set, clusters from type 1 and type 2 fragments will always be reported separately (TN-1, TN-2, TN+1, TN+2). A similar criteria applies to inversion structural variants (see Figure 5).

5.3.2 Standard Clusters File Format

Columns are tab-separated in the following format:

#Cluster_ID: Unique text identifier for the cluster c<cluster_number>. In --maximal mode, the sub-clusters are named as in: c[MajorClusterNumber].[MinorSubclusterNumber]

LeftChr: Chromosome containing the left breakpoint.

LeftBreakPoint: Interval containing the left breakpoint.

RightChr: Chromosome containing the right breakpoint. (Note that except for translocations LeftChr = RightChr).

RightBreakPoint: Interval containing the right breakpoint.

Num PRS: Number of PRs that support this variant.

Localization: Square root of the breakpoint region (polygon of intersection). The smaller the localization, the more precisely the breakpoint can be determined. See below for more information on localization.

Type: The type of structural variant indicated (See Section 5.3.1).

Example Clusters File:

#Cluster_ID:	LeftChr:	LeftBreakPoint:	RightChr:	RightBreakPoint:	Num PRS:	Localization:	Type:
c521	17	40566182,40566330	17	40567335,40567463	4	86.4	IR

Table 2: GASV Default Clusters Format

Note: If the localization is equal to -1, this means that while a set of PRs overlap, there is not a single structural variant that can explain the data. Running GASV with the "--maximal" flag will output all maximal structural variant predictions from such a cluster.

For examples and more information on maximality, and localization refer to Sindi, et al. (2009).

5.3.3 Additional Output Options

GASV provides two additional formatting options for the clusters file. These options are accessible by using the --output <val> option.

---

reads: The same as "standard" but has an additional column with a comma separated list of the names of names of PRs in each cluster.

---

regions: Highly detailed output that specifying the coordinates of the breakpoint polygon itself. The tab separated columns are:

> #Cluster_ID:
>
> Num PRS:
>
> Localization:
>
> Type:
>
> List of PRS:
>
> LeftChr:
>
> RightChr:
>
> Boundary Points:

The boundary points are given in the form: $x1, y1, x2, y2, \ldots$ where the points (x_i, y_i) correspond to the boundary of the breakpoint polygon.

Note: The output produced from GASV.jar may differ from output produced by earlier versions of GASV. Refer to RELEASE_NOTES.txt for more information.

6 GASVPro

6.1 Running GASVPro

GASVPro is a pipeline combining GASV predictions from discordant paired-read mappings with read depth information in a single probabilistic model. As detailed in Section 3, we provide scripts to simplify the GASVPro execution. We encourage users to adapt the scripts provided for their specific workflows, and to aid in the process we detail the specific parts of the GASVPro pipeline.

6.2 GASVPro-CC

GASVPro-CC computes the read depth information associated with each prediction. The program is run by passing a parameters file and a `.clusters` file on the command line as follows:

```
$ ./GASVPro <parametersFile> <clustersFile>
```

There are many different options available, detailed below; however, for ease of use a parameters file is created with the execution of BAMToGASV when the **-WRITE_CONCORDANT** or **-GASVPRO** option is set to **True** (see Example above). Users are encouraged to begin with this default parameters file and to add/modify options as needed by their specific application.

6.2.1 Parameters

Here we provide a sample parameters file.

```
# A Sample GASVPro parameters file
# Annotation can appear on any line beginning with a hash mark (#).
# REQUIRED PARAMETERS:
ConcordantFile: <ConcordanceFile>
Lavg: <value>
ReadLen: <value>
Lambda: <value>
Perr: <value>
Limit: <value>
Tolerance: <value>
LRThreshold: <value/all>
MaxChrNumber: <value>
# OPTIONAL PARAMETERS:
UNIQUEFile: <UniquenessFile>
MaxUniqueValue: <value>
MinScaledUniqueness: <value>
Translocations: <true/false>
TransOnly: <true/false>
Verbose: Y
```

Figure 6: **A Sample GASVPro Parameters File**

6.2.2   Input Files

Because GASVPro was designed to be run after BAMtoGASV and GASV, the input files for GASVPro will appear familiar to those who have experience with the GASV pipeline. There are also a number of parameters needed by the probabilistic model of GASVPro.

Clusters File: must be in GASV Regions Clusters Format (see section 4.2).

- **NOTE:** The `--output regions` parameter must be passed to GASV to produce a clusters file of the correct format.
- **NOTE:** The current version of GASVPro does not support divergent variants. They can be included in cluster files but will be ignored.

Concordant File: must be in BAMtoGASV default output format (<chr> <start> <end>).

- The BAMtoGASV preprocessor produces these files with either the **-WRITE_CONCORDANT** or **-GASVPRO** option set to **True**. See Section 3.2 for details.
- **NOTE:** If users provide their own concordant file it must follow the same format and **must** be sorted numerically by start coordinate.

Uniqueness File: Because GASVPro's model is based on read depth, users may want to incorporate biases in the expected coverage (such as GC content or local mapability). A uniqueness file is a **single file** of format <chr> <start> <end> <unique>. (See Section 6.3 for more information.) The uniqueness file **must** be sorted numerically by start coordinate.

6.2.3   Probability Model Parameters

Average Fragment Length (Lavg) & Average Read Length (ReadLen): *follow from sequencing libraries. In the event of multiple libraries, use the average over all datasets.*

Lambda & Perr: *parameters for probabilistic model, see manuscript.* **Perr Default Value**: `0.01`

Minimum Fragment Size (Limit): this value is the smallest variant that GASVPro will use concordant coverage on. **Default Value**: `1000`

Tolerance: Tolerance is used to remove regions that have abnormally high concordant coverage and, thus, can be eliminated from consideration as a structural variant. A tolerance of t will eliminate from further consideration as a potential structural variant a prediction with N concordant fragments provided $P(\geq N \mid \text{no variant}) < t$. **Default Value**: `0.0001`

Max Uniqueness Value: this parameter is the value assigned to the regions of the genome with optimal mapability.

Minimum Scaled Uniqueness: this parameter is the minimum value to which GASVPro will scale the uniqueness values for each variant. **NOTE:** When this parameter is set to 1, all regions will have the same uniqueness value. **Default Value**: `0.3`

6.2.4 Parameters for Formatting Output

Max Chromosome Number: this is the highest *numerical value label* of all the chromosomes in the Clusters File. **Default Value**: `100`

Translocations Mode: running GASVPro-CC in translocations mode will enable the processing of translocation clusters along with deletions and inversions (see page 18 for accepted translocation types). **WARNING: Processing translocations takes exponentially more time than processing deletions and inversions alone.** If possible, we recommend running Translocations Only mode on datasets with large numbers of translocations and possibly running two copies of GASVPro-CC in parallel (one for translocations, another for inversions and deletions). **Default Value**: `false`

Translocations Only (TransOnly): setting this value to `true` will process translocations ONLY, ignoring all other types. **WARNING: This operation can take a very long time. Default Value**: `false`

Likelihood Ratio Threshold (LRThreshold): Users can input a numerical value or the string `all`. GASVPro will only return clusters with a LR greater than or equal to the input value (See section 6.3.3 for explanation of LR). If `all` is given, all clusters will be printed. **Default Value**: `all`

Verbose: specifying `Y` will print information for each cluster processed.

6.3 GASVPro Model Overview

We provide a brief overview of the GASVPro probabilistic model to assist users in selecting meaningful parameter values. A full description of our model and algorithm is available in our manuscript (Sindi et al., 2012).

6.3.1 Poisson Coverage Model

Our probabilistic model is based on reduced coverage by concordant fragments at SV breakpoints. A deletion SV leaves a strong signal of reduced coverage, as shown in Figure 2. That is, for a true deletion coverage by concordant fragments should be reduced throughout the entire deleted segment.

Let B be a prediction deletion breakend region defined by k discordant fragments. Define $a_{\max} = \arg\max_a\{(a, b) \in B\}$, the maximum choice for a and $b_{\min} = \arg\min_b\{(a, b) \in B\}$, the minimum choice for b. For any choice of mated breakends $(a, b) \in B$, the interval $I(B) = [a_{\max}, b_{\min}]$ will be deleted. Let $n(I)$ be the number of concordant fragments whose mappings overlap the interval $I(B)$.

We follow the Poisson model for coverage. If N fragments with average size L_{avg} have been sequenced from a genome of size G, then the number of fragments containing an arbitrary position in the genome is approximately Poisson with mean $\lambda = NL/G$. Thus, we expect the number of fragments k covering a single point to be Poisson with mean λ and overlapping an interval $n(I)$ to be Poisson with mean λ_I where

$$\lambda_I = \lambda((b_{\min} - a_{\max}) + L_{avg}).$$

GASVPro considers the probability of three events:

- Prob(B is a homozygous deletion); $P(2) = \left(p_{err}^{n(I)} \right) Pois(\lambda, k)$,

- Prob(B is a heterozygous deletion) $P(1) = Pois\left(\lambda_I/2, n(I) \right) Pois(\lambda/2, k)$,

- Prob(B is an error); $P(0) = Pois\left(\lambda_I, n(I) \right) \left(p_{err}^{k} \right)$,

where p_{err} is the probability of an erroneous mapping. The final log-likelihood ratio output is the

$$\log \Lambda(V) = \frac{\max\{\log(P(2)), \log(P(1))\}}{\log(P(0))}. \tag{1}$$

6.3.2 Uniqueness Scaling

In addition to structural variants, there are several local factors that will affect coverage by concordant fragments. Reads originating from duplications present in both the test and reference genome can not be mapped to a unique position. Thus, such regions will have low coverage due to restrictions in local mapability. Further, some technologies have biases due to sequence composition. To account for these differences, we allow scaling of $n(I)$ to more accurately reflect these biases. (In the GASVPro manuscript we used theRosetta Uniqueness Track; however, users may supply their own track.)

An uniqueness file contains genomic intervals along with a numerical value describing their uniqueness. As mentioned above, this is a single file formatted: <chr> <start> <end> <uniqueness>.

Internally, GASVPro uses this information to scale the number $n(I)$ of concordant fragments mapping to $I(B)$. The scaled concordant coverage is $\hat{n}(I)$, where

$$\hat{n}(I) = \frac{n(I)}{\alpha + (1 - \alpha)\overline{U}(I)}, \tag{2}$$

where $\overline{U}(I)$ is the average uniqueness value of a point in interval $I(B)$ divided by the maximum possible uniqueness track and by default $\alpha = 0.3$. Notice, when the interval I does not have compromised mapability, i.e., $\overline{U}(I) = 1$, we do not adjust the number of observed fragments, $\hat{n}(I) = n(I)$.

- When a uniqueness file is not specified, all regions are taken to have scaled uniqueness value 1.

- Users must specify a "maxUniqueValue" to use the track because we assume all genomic intervals not listed in a uniqueness file to have this uniqueness. (As such, users may only list regions of the genome with low-mapability.)

- The parameter α can be changed with the "minScaledUniqueness" parameter.

- As described in our manuscript, we only scale concordant coverage in the case of deletions.

6.3.3 GASVPro-CC Output

After successful completion GASVPro will output two files, a `.clusters` file and a `.coverage` file. Input parameters relevant to subsequent steps in the GASV pipeline will also be included in the filenames. The `.clusters` file contains the variance probabilities that GASVPro calculated. Here is an example GASVPro output, contained in the `.clusters` file:

#Cluster_ID:	NumPRS:	Localization:	Type:	$\cdots$	$\cdots$	$\cdots$	$\cdots$	Likelihood Ratio:
c19	1	212	TNR-	$\cdots$	$\cdots$	$\cdots$	$\cdots$	620.355

Table 3: An Example GASVPro Output Cluster (see Section 5.3.3 for all fields.)

Let the probability that a given cluster C *is* a variant be given by P_v and the probability that C *is not* a variant be given by P_{nv}. The Likelihood Ratio therefore is $P_v - P_{nv}$. The rest of the information in the `.clusters` file is identical to the input file.

Additionally, GASVPro will output another file which offers more information generated directly by the program. This file will have a `.coverage` extension and includes the following information:

> #Cluster_ID: the cluster number.
>
> Type: See page 18. NOTE: This GASVPro version currently does not support divergent variants.
>
> NumDiscordants: number of discordant fragments supporting this cluster.
>
> LogVariantProbability: the log probability that this cluster is a variant.
>
> LogNoVariantProbability: the log probability that this cluster is not a variant.
>
> Start: start coordinate.
>
> End: end coordinate.
>
> ConcordantCovLeft: concordant fragment coverage on left.
>
> ConcordantCovRight: concordant fragment coverage on right.
>
> MapabilityLeft: cluster uniqueness on left.
>
> MapabilityRight: cluster uniqueness on right.
>
> Algorithm Code: indicates which process was used to calculate these values. `0` indicates the breakend read-depth model, and `1` indicates the interval read-depth model.

6.4 GASVPro-graph

GASVPro-graph is a component of the GASVPro pipeline that separates SV predictions into independent subsets for further analysis by the MCMC algorithm (below). It takes the following parameters:

- Clusters File: produced by GASV.

- Coverage File: produced by GASVPro-CC.

- Output Directory: directory into which the output of GASVPro-graph and GASVPro-mcmc will be placed. If the specified directory does not exist, it will be created.

- (*optional*) Minimum Support: the minimum level of support required for a cluster to be considered (default: 4).

GASVPro-graph is located in the `/bin` directory. Run it with the following command:

```
./GASVPro-graph {Clusters File} {Coverage File} {Output Directory} <{Minimum Support}>
```

Output from GASVPro-graph, inside the output directory given at runtime, is a series of `sv_*.sv` files, a `p_star.summary` file, and a `svFileList.summary` file. Each `sv_*.sv` file contains the clusters and coverage information for an individual independent component. The `svFileList.summary` file is an index of these independent components containing each file and the number of clusters within it.

6.5   GASVPro-mcmc

GASVPro-mcmc is the final processing component of the GASVPro pipeline. It reports the most probable SV predictions over the space of possible paired-read alignments using a Markov chain Monte Carlo procedure. It takes the following parameters:

- Parameters File: use the same parameters file used with GASVPro-CC, created by BAMTo-GASV.

- Working Directory: use the directory into which GASVPro-graph placed its output.

- (*optional*) Start Cluster: optionally, specify a ClusterID at which to start processing.

- (*optional*) End Cluster: optionally, specify a ClusterID at which to end processing. (NOTE: these optional parameters are useful for running GASVPro-mcmc in parallel).

GASVPro-mcmc is located in the `/bin` directory. Run it with the following command:

```
./GASVPro-mcmc {Parameters File} {Working Directory} <{Start_cluster} {End_Cluster}>
```

Upon completion of each component GASVPro-mcmc outputs three files for each structural variant file in the working directory:

`*_sv_`N`.MCMCThreshold.clusters`: contains the final group of clusters associated with a given structural variant.

`*_sv_`N`.PR.results`

`*_sv_`N`.Variant.results`

The latter two files describe sampling of assignments for PR and variants themselves:

`*_sv_`N`.PR.results`
For each PR analyzed, we output the fraction of time it occupies each possible assignment (including errors) during the MCMC sampling procedure.

`*_sv_`N`.Variant.results`
For each variant analyzed, we output the fraction of time it has each possible number of supporting PRs (including 0) during the MCMC sampling procedure.

26