svviz Documentation

Release 1.6.2

Noah Spies

Aug 17, 2017

Contents

CHAPTER 1

Introduction to svviz

The program takes as input one or several bam files with sequencing data, a genome fasta file, information about a putative structural variant (that has been identified using other methods), and (optionally) a bed file with genomic annotations. From these inputs, `svviz` uses a realignment process to identify reads supporting the reference allele, reads supporting the structural variant (or alternate allele), and reads that are not informative one way or the other (ambiguous).

Reads are then plotted, as in a standard genome browser, but along the sequence of either the alternate allele or the reference allele. The user can thus assess the support for the putative structural variant visually, determine if the breakpoints appear accurate, or estimate a likely genotype for each sample at the given structural variant.

`svviz` differs from existing genome browsers in being able to display arbitrary types of structural variants, such as large insertions, deletions, inversions and translocations. Genome browsers such as IGV are poorly suited to displaying the read data supporting these types of structural variants, which can differ dramatically from the reference genome sequence.

CHAPTER 2

Features

- **built-in support for:**
  - translocations (new feature!)
  - deletions
  - **insertions**
    - including mobile element insertions
    - complex insertions (where some sequence is deleted)
  - inversions
  - `svviz` can easily be extended to analyze translocations and complex variants, but these types are not yet implemented.
- builds reference and alternate allele sequences from genome fasta file and structural variant annotation
- identifies, from the input bam file(s), which reads (both read ends for paired-end sequencing) are likely to be relevant for the given structural variant
- performs Smith-Waterman realignment of all read segments against both alternate and reference allele sequences
- uses alignment score to determine reads supporting reference or alternate allele
- additionally, uses empirical insert-size distribution (rather than mean and stddev) to assign reads as likely derived from alternate or reference allele
- provides a (locally-running) browser-based front-end for inspecting visualizations
- **visualizes reads for multiple samples in SVG format (an open-source, web-standard vector graphics format)**
  - shows mismatched bases indicative of sequence polymorphisms (eg SNPs) or mapping errors
  - can visualize BED or GTF format annotations, for example the locations of genes or repeats
- optionally visualizes repetitiveness in the breakpoint regions using dotplots
- options to export to PDF or PNG

- batch mode to calculate summary statistics and create visualizations for many events from a VCF file

CHAPTER 3

Installation

A single command should typically suffice to install `svviz`:

```
sudo pip install svviz
```

More detailed directions follow for linux and OS X. Try installing into a *virtual environment (see below)* if you have difficulty with any of these directions.

svviz is not currently supported on Windows, although it should probably work there.

Installation on Linux

1. Ensure that the following prerequisites are installed: python and pip. If you are running a Debian-based Linux (for example Ubuntu 12.04 or later), you can use the following commands to ensure they are installed correctly:

   ```
   sudo apt-get install python-dev
   sudo apt-get install python-pip
   ```

2. Use pip to install svviz:

   ```
   sudo pip install svviz
   ```

   This should automatically install a number of required python packages (see *below*). If you prefer to install the latest development version from github, install git and then use this command instead (warning: bleeding edge! may contain bugs!):

   ```
   sudo pip install -U git+git://github.com/svviz/svviz.git
   ```

3. To enable PDF export, you will need to install the libRsvg package:

   ```
   sudo apt-get install librsvg2-dev
   ```

   or the Inkscape application:

```
sudo apt-get install inkscape
```

See the *Troubleshooting* section if you have difficulties after following the above directions.

Installation on Mac OS X

1. Ensure that you have a working compiler by following these instructions. If you already have gcc or clang installed, you can skip this step.
2. Ensure that you have the python package `pip` installed. If it is not already installed, follow the directions on the pypa website or use the following command (requires `easy_install`, which should ship with most versions of OS X):

   ```
   sudo easy_install pip
   ```

2. Use pip to install svviz:

   ```
   sudo pip install svviz
   ```

   This should automatically install a number of required python packages (see *below*). If you prefer to install the latest development version from github, install git and then use this command instead (warning: bleeding edge! may contain bugs!):

   ```
   sudo pip install -U git+git://github.com/svviz/svviz.git
   ```

3. To enable PDF export, you have two options.
   - The first, and recommended option, is to use webkitToPDF, a simple homegrown command-line program that uses OS X's built-in web rendering engine to convert SVGs (`svviz`'s native format) into PDF. As its name implies, `webkitToPDF` does not support PNG support. To use `webkitToPDF` with `svviz`, simply download the OS X app and add it to your PATH.
   - The second option is to use libRsvg package. First install and update homebrew and then run `brew install librsvg`. Export using `libRsvg` supports both PNG and PDF formats.

See the *Troubleshooting* section if you have difficulties after following the above directions.

Required python packages

`svviz` requires several python packages in order to run properly. During a normal installation, these packages should be installed automatically:

- flask
- joblib
- numpy
- pyfaidx
- pysam
- requests

Some additional functionality is provided by the following optional python packages (not installed automatically; use `sudo pip install pandas`, etc):

- pandas
- rpy2

Running the demos

Several example datasets can be downloaded and run directly from `svviz`. This is a good step to perform in order to make sure everything is installed correctly:

```
svviz demo
```

(Additional demos can be run as `svviz demo 2`, `svviz demo 3`, etc.)

This will prompt you if you would like to download the example datasets into the current working directory. If you type `y` to indicate yes, the data will be downloaded, then automatically analyzed and visualized in your web browser. The first line of output (following the download) shows the command line arguments used to analyze the demo; if you wish to play around with parameters (for example adding or removing datasets, or refining the breakpoints), you can copy and edit this command.

Subsequent lines of output will indicate progress of `svviz` as it processes the data. Once processing is complete (should typically take ~1min), `svviz` will create a local web-server (accessible only from within your computer) and open your default web browser to a page displaying the example structural variant.

Troubleshooting

svviz won't install

1. Do you have a C compiler installed? You will need to install the command-line tools if you are on OS X.
2. **Have you tried installing svviz in a virtual environment? This helps rule out problems with incorrect dependencies:**

    (a) Install virtualenv: `sudo pip install -U virtualenv`

    (b) Create a virtual environment: `virtualenv svviz_env`

    (c) Activate the environment: `source svviz_env/bin/activate`

    (d) Install svviz: `pip install -U svviz` (note that when installing to a virtualenv, you should not need to be superuser)

I can't access the web view

1. Are you running the web browser on the same computer as svviz? For security reasons, the web server is only available within the same computer. To safely get around this, you will need to set up an ssh tunnel from one computer to the other (see *here* for directions)
2. Are you accessing the correct URL? The server always runs on localhost, but the port is chosen randomly and may change between runs.
3. Have you tried reloading the page? The server can take a moment to start, and this may take longer than it takes for your web browser to open.

The web view opens but only shows summary statistics, no track data

It may take a minute or so to load the data tracks into the browser, depending on how many reads are present in the region of the variant being visualized. The alt tracks will load first, but the view will not become fully interactive until the ambiguous tracks finish loading. If you are having trouble with the loading taking too long, please submit a bug report.

I can run the demo, but I can't load my own data

1. Is your input bam file coordinate-sorted and indexed properly? Try removing your sample.bam.bai file and recreating it using `samtools index sample.bam`.
2. Have you checked that you've specified the correct command-line options? The first line of output after you run svviz, if there is not an error parsing the input, shows how svviz interpreted the command line arguments you provided. This can help you track down a potential misspelling or other error in specifying command line arguments.
3. Have you properly specified the variant coordinates on the command-line? Running `svviz` without any arguments will output the help, including how to specify the variant coordinates.

Other problems

See the *FAQs* for answers to other questions.

Please report any other problems on the issues page of the github project site.

CHAPTER 4

Visual output

The visualizations produced by svviz are split into several sections. The top shows some information about the structural variant that is being visualized, including type and genomic coordinates, as well as summaries of read counts in each category.

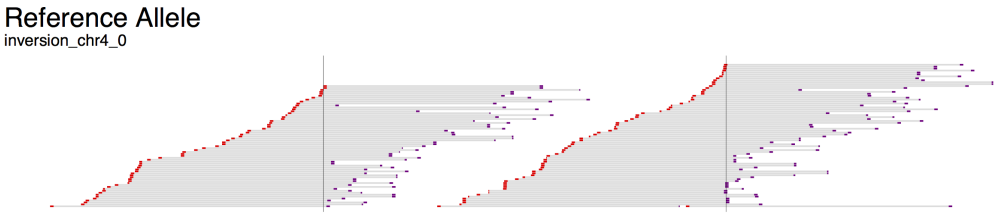


Fig. 4.1: Example view of the reference allele. Breakpoints are indicated by thin vertical lines.

Below the overview information, the visualization is split into three parts: "Alt", "Ref" and "Amb" (the ambiguous track is only shown in the web view and not in the exported visualizations). Reads aligning better to the alternate allele than the reference allele will be shown in the first set of tracks, whereas those aligning better to the reference allele will be shown in the middle set of tracks.

At the bottom (web view only), in the "Amb" section, are reads that align poorly to both alleles, or equally well to both alleles, and hence do not provide evidence for or against the structural variant being analyzed. These reads are shown aligned against the reference sequence, and typically include reads that map near the structural variant breakpoints but do not map across the breakpoints.

An axis is shown below each section, showing the scale in basepairs, as well as indicating the positioning of the breakpoints and the orientation of each region between breakpoints (large arrows). Coloring of the segments is consistent between Alt, Ref and Amb sections.

Fig. 4.2: Axis for inversion on chromosome 4 (Alt allele).

For example, the first demonstration (`svviz demo`), which analyzes an inversion on chromosome 4, displays three colored bars: red, blue and then grey. The blue segment in the middle indicates the inverted region, and hence the arrows point to the left in the Alt section but to the right in the Ref section. Breakpoints are indicated as vertical lines through all the tracks.

The read alignments displayed in svviz are the result of a Smith-Waterman realignment of the full read sequence against the reference allele or the inferred alternate allele. Reads aligning to the minus strand are colored red and those aligning to the plus strand are colored purple. If the sample contains paired-end sequencing reads, the unsequenced space between read pairs is shown by light gray bars. Only reads matching the expected orientation (typically inwards-facing for standard Illumina libraries) will be assigned to one of the alleles (those aligning with unexpected orientations will be counted as ambiguous).

Overlapping portions of read pairs are shown in a light green color (for example, if the insert size is 250 and 2x150bp sequencing was performed, the middle 50bp would be sequenced from both reads).

Mismatches to the sequence of the allele are shown as colored vertical bars with the width of a nucleotide (you may need to zoom in to see them). Insertions or deletions, relative to the sequence of the assigned allele, are displayed as gray and cyan bars, respectively. Sequencing errors will thus show up as isolated vertical bars within individual read sequences; polymorphisms (SNPs and indels) will show up as vertical bars spanning multiple reads at the same position.

Web interface

In the interactive web view, zooming can be accomplished by clicking the plus or minus buttons, or by holding down the option (Mac) or alt (linux) key and spinning the scroll wheel while hovering over a track. If you hover your mouse over a read, the nucleotide-level alignment(s) will be shown for the read (or read-pair), along with some information about the length of the aligned read (or read pairs) and the reason the read (pair) was assigned to a given allele.

Scrolling using the scroll-wheel or panning by clicking and dragging pans all tracks simultaneously, whereas using the vertical scrollbars only scrolls the track of interest. Horizontal scrolling can also be performed using the scroll wheel by holding down shift.

To stop running the locally-hosted server and quit python, press ctrl-c in the terminal window. Any web views you have open will still show the reads, but information will no longer be updated when you hover your mouse over a read.

Complex variants

For some structural variant types, the breakpoints are distant in the genome (in the extreme, they fall on entirely different chromosomes). An example might be a translocation between chromosome 6 and chromosome 19. To visualize these types of events, svviz visualizes each breakpoint region in a separate "chromosome part", which are then separated visually by a thick gray line.

Here, in the reference allele, you can see the chromosome 19 part colored red (upstream of breakpoint) and blue (downstream of breakpoint). The chromosome 6 part is shown in reverse orientation (indicated by axis arrows pointing left), with the segments colored gray (upstream) and yellow (downstream). Reads are shown tiling nicely across both breakpoints in both the cancer and matched normal sample.

The alternate allele shows the fusion of the red upstream chromosome 19 segment with the gray upstream segment of chromosome 6. Reads tile across this translocation breakpoint in the cancer but not the normal sample (here, this is a fusion of two genic regions in the intronic regions; nearby genes can be shown using bed/gff format gene annotations). Note also that the reciprocal event (the yellow/blue fusion) is shown but with zero supporting reads.

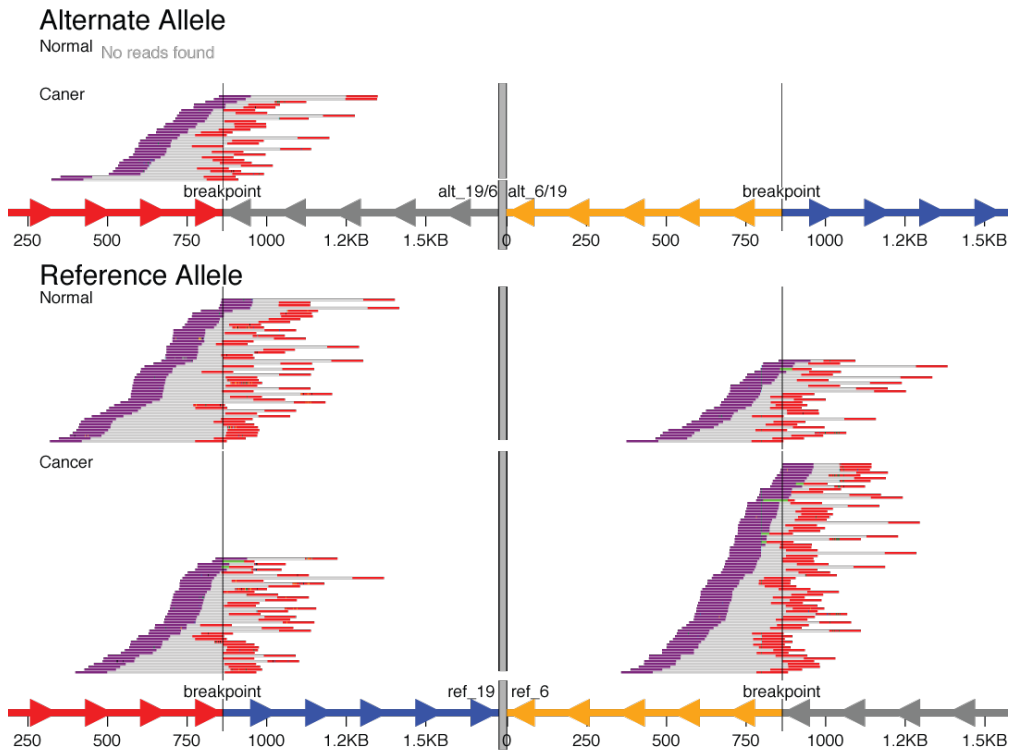


Fig. 4.3: A translocation between chromosomes 6 and 19 that was found only in the cancer sample but not the matched normal sample.

CHAPTER 5

Command line interface

Loading and visualizing your structural variant

To visualize your structural variant of interest, you will need at least the following three pieces of data:

1. Input bam file(s). This bam file must be coordinate-sorted and have an index file (sample.bam.bai) in the same directory. You can use samtools to sort and index your bam file. Bam files are specified using the `-b` command line argument, which can be provided multiple times to load and visualize multiple samples simultaneously.
2. Input genome fasta file. If a .fai index file does not already exist in the directory containing the fasta file, it will be created. This means that, if the .fai file does not already exist, the fasta file needs to be in a directory for which you have write permission. This fasta file is the first required command line argument.
3. The coordinates of the structural variant. The type of event is specified by the `-t` command line option. The following four event types are currently supported:

   Deletions The format for specifying deletion breakpoints is `chrom start end`.

   Inversions To specify an inverted region, use `chrom start end`.

   Insertions The format for specifying insertions is `chrom breakpoint <inserted sequence>`.

   Mobile elements Mobile element insertions can be specified by `<mobile_elements.fasta> <chrom> <pos> <ME name> [ME strand [start [end]]]`, where `<ME name>` must match the header line from the mobile_elements.fasta file, and strand, start and end are optional coordinates of the relevant portion from the mobile element sequence.

   Translocations Translocations can be specified using the following format: `chrom1 pos1 chrom2 pos2 orientation`, where `orientation` is either + or −, and specifies whether region1 and region2 are both on the plus strand of the genome, or are on opposite genomic strands; and pos1 and pos2 are the chromosomal coordinates of the breakpoints. See *below* for more info.

Breakend Additional types of structural variant can be specified using breakend format: `chrom1 pos1 strand1 chrom2 pos2 strand2`. Note that, due to limitations of the Smith-Waterman alignment library used by svviz, breakend breakpoints must be distant from one another, relative to the insert size/read length. See *below* for more info.

Batch see *here*

For example, a deletion might be called as:

```
svviz -t del -b sample1.sorted.bam -b sample2.sorted.bam hg19.fasta chr7 153757067␣
↪153758235
```

Displaying annotations

Annotation tracks can be loaded and visualized in order to display the position of important nearby genomic regions such as genes or repeat sequences. These need to be provided in standard BED format (the first 6 columns are required, up to and including strand). Such annotation tracks can easily be downloaded from the UCSC Genome Browser, either from the standard annotations provided for each assembly or using their Table Browser tool. Each bed file is specified with the `--annotations` option (or `-A`).

Exporting visualizations

The visualizations can be exported to SVG, PNG or PDF from the web view by clicking the "Export" link at the top of the web view. Alternatively, these files can be created directly, without launching the web interface, using the `--export` option (and this exported image file can be opened automatically using your system-defined image viewer by additionally specifying the `--open-exported` or `-O` option).

SVG is svviz's native format and requires no additional software. Export to PNG or PDF requires either librsvg or Inkscape (see the *installation instructions* for more info). An option for PDF export on OS X only is webkitToPDF. When multiple conversion backends are installed, you can specify which one is used to convert to PDF/PNG using the `--converter` command line option. For example, add the option `--converter inkscape` to specify that conversion should use Inkscape instead of one of the other backends. If this option is not specified, svviz will automatically pick from the installed backends.

Additional options

The default settings are typically correct for Illumina data. Read orientation and insert sizes will be inferred for each input library. Sequencing platforms that have a substantially higher error rate than Illumina may need adjusting of the `--aln-quality` and the `--aln-score-delta` options.

The `--lenient` option is recommended for pacific biosciences sequencing (because PacBio sequencing is typically of lower base-quality than Illumina sequencing, this preset changes the `--aln-quality` option to retain lower quality alignments as support for the Ref and Alt alleles; and this also changes the `--aln-score-delta` to only assign a read to an allele if the difference in alignment scores exceeds a threshold relative to a fraction of the read length, rather than a small fixed value).

The `--min-mapq` option specifies the mapping quality threshold; reads with mapq (this is set during the original genome-wide mapping by bwa, bowtie, etc) below this threshold will be discarded during pre-processing. A similar argument, `--pair-min-mapq`, can be used instead to require that at least one read end out of a read pair must have a mapq exceeding this value (this read end must be near the variant being analyzed). The `--dotplots`

option will create a dotplot to visualize sequence similarity within the genomic region(s) surrounding the structural variant. This depends on the optional python package rpy2 (first make sure R is installed and then install rpy2 using the command `sudo pip install rpy2`). You will also need to install yass, which can be installed using the homebrew command `brew install homebrew/science/yass` (OS X only) or yass can be downloaded, compiled and installed according to the instructions here (linux and OS X).

The dotplot output shows regions of similarity within the reference allele as lines: blue lines indicate similarity on the same strand and direction whereas red indicates similarity on the opposite strand/direction. Because the similarity matrix is symmetrical, only same strand similarities are shown in the upper left half and only opposite strand similarities are shown in the bottom right half. The structural variant breakpoints are shown as dashed gray lines.
A related option is `--max-multimapping-similarity`, which adjust how aggressively svviz filters out reads that potentially align to multiple locations near the structural variant. The default score of 0.95 means that any read (for paired-end reads, this means any read-end) whose second-best alignment score is more than 0.95 times the best alignment score will be assigned as ambiguous. For example, if the best alignment score is 445, and the second-best alignment score is 439, the multimapping similarity would be 439/445=0.99 and the read would be marked as ambiguous. However, a read whose best alignment score is 445 but second-best alignment score is 405 would not be filtered because the multimapping similarity of 395/445=0.89 is less than 0.95.

Important note for long-read sequencing data (eg PacBio)

svviz performs a Smith-Waterman realignment of every read against both the reference and alternate allele sequences. This realignment is performed by an extremely fast implementation called ssw. As of this writing, ssw has a serious bug which manifests as a segmentation fault (ie crash) occasionally when aligning two long sequences to one another, as might occur when aligning PacBio reads. To overcome this problem, svviz has an optional wrapper mode which runs ssw in a separate process; reads which cause a crash when aligning using ssw will be skipped. Please note that this mode runs somewhat more slowly when aligning many short reads, so it is recommended to use this mode only when necessary.

To use this wrapper mode, specify `--processes -1` on the command line.

Translocations and Breakends

Complex variants (first introduced *here*) can be visualized using the translocation or breakend event types (specify event type using the command line options `--type tra` or `--type bkend`). Two possible orientations are possible for a translocation, "+" and "–":

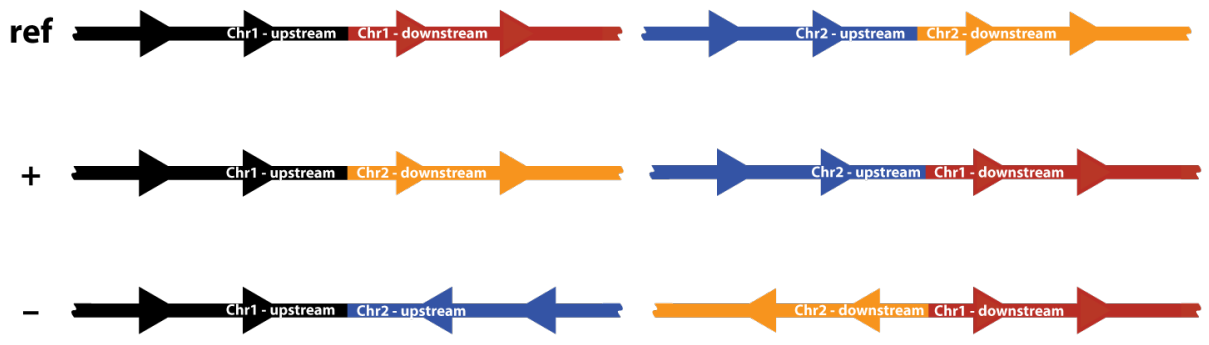


Fig. 5.1: Possible orientations for a translocation event.

The top row shows the two reference chromosomes; for each chromosome, a region upstream and a region downstream of the breakpoint is shown in different colors. There are two possible fusion chromosomes for the "+" orientation. In the first, the upstream half of chromosome 1 (black) is fused to the downstream half of chromosome 2 (orange). In the second, reciprocal event, the upstream chromosome 2 half (blue) is fused to the downstream chromosome 1 half (red).

The "–" orientation works similarly, but here the plus strand of one chromosome is fused to the minus strand of the other chromosome.

For example, if the breakpoints were located at chr1:32,456,789 and chr2:12,468,579, then the "+" orientation event would be specified as `chr1 32456789 chr2 12468579 +` and the "–" orientation event would be specified as `chr1 32456789 chr2 12468579 -`. Remember to use `--type tra`.

svviz always shows both reciprocal halves of a translocation – if the translocation was not reciprocal, then reads should only support one fusion and not the other. To analyze only a single fusion, use the more general purpose breakend event type.

The breakend event type (`--type bkend`) takes two genomic locations and analyzes the structural variant formed by joining the two genomic regions. A few examples:

- a large deletion: in this case two breakpoints lie distantly on the same chromosome; the alternate allele shows the deletion allele formed by joining the upstream and downstream regions
- an inversion breakpoint: again, the two breakpoints lie on the same chromosome; the alternate allele shows the joining of two distant regions originally on the same strand but now head-to-head on opposite strands (note that the two inversion breakpoints must be analyzed separately using the breakend event type; the built-in inversion type analyzes both breakpoints simultaneously)
- a translocation: as mentioned above, the breakend event type can be used to show a single chromosomal fusion (rather than both reciprocal events)

For example, the four fusions shown in the figure above would be specified as follows in breakend format:

- `chr1 32456789 + chr2 12468579 +` (top-left)
- `chr2 12468579 + chr1 32456789 +` (top-right)
- `chr1 32456789 + chr2 12468579 -` (bottom-left)
- `chr2 12468579 - chr1 32456789 +` (bottom-right)

CHAPTER 6

Summary output

Each line describes a single summary statistics for a single allele in a single sample for one variant. For example,

```
variant                                  sample          allele  key     value
Deletion::chr1:724,921-726,121(1200)     HG002_MP_L1_L2  alt     count   4
Deletion::chr1:724,921-726,121(1200)     HG002_MP_L1_L2  ref     count   75
```

The following code illustrates one approach to analyzing this summary file from python (using the pandas library):

```
import pandas as pd
summary = pd.read_table("events_summary.tsv", sep="\t")
print summary.pivot_table(values="value", index=["variant","sample","allele"],␣
↪columns="key")
```

A partial description of the summary output follows:

- **count**: the number of reads supporting the given allele
- **alnScore_mean** and **alnScore_std**: the mean and standard deviation of the alignment scores; note that the alignment scores will vary substantially if there is heterogeneity of sequencing read lengths, as there is in, for example, PacBio data, or Illumina data when adapter sequences have been stripped
- **insertSize_mean** and **insertSize_std**: the mean and standard deviation of the insert sizes (if the data is paired-ended) or the length of the reads (if the data is single-ended); this is calculated *after* realignment, and so includes all gaps in the alignments, but does not include any clipped bases if the alignment does not include the entire read sequence
- **reason_*: these lines count how many reads were assigned to the given allele because of the given "reason":**
  - **reason_alignmentScore**: the alignment score for this allele was better than for the other
  - **reason_insertSizeScore**: the insert size for this allele was a better match to the background distribution
  - **reason_orientation**: this allele had the correct paired-end read orientation but the other allele did not

- **reason_multimapping**: these reads were assigned to ambiguous because it aligned well in two locations near the structural variant
- **reason_multiRegion**: these reads were assigned to an allele because they mapped to multiple chromosome parts in the other allele; in *this example translocation*, a read pair mapping to the red portion of chromosome 19 and the gray portion of chromosome 6 would be considered an invalid "multi-region" alignment in the reference allele, but would be considered a valid alignment across the translocation breakpoint in the alternate allele; thus, this read-pair would be assigned to the alt allele and given the "reason_multiRegion" tag

CHAPTER 7

Batch mode

To summarize a number of structural variants at once, svviz supports a batch mode.

To run batch mode, use `--type batch`, and specify (1) the reference genome (in fasta format, as above) and (2) a VCF file describing the SVs to be analyzed. These SVs must be of supported types (insertions, deletions, inversions and mobile element insertions), and specified in VCF 4.0 Format.

You will probably also wish to use the `--summary` option to specify a tab-delimited output file with the full summary statistics describing each variant and allele.

The visualizations can still be created and exported in batch mode. While in batch mode, the `--export` command-line option specifies a directory into which to place the exported visualizations. These files are named by the type and position of the event, so there will be one file per event. The default is PDF format (this can be changed by using the `--format` option).

The following columns are required in the input VCF files:

Deletions

- chromosome (column 0)
- start coordinate (column 1)
- SVTYPE=DEL;END=<end coordinate> (column 7)

Insertions

- chromosome (column 0)
- start coordinate (column 1)
- SVTYPE=INS;END=<end coordinate> (column 7)
- **the inserted sequence must be specified either:**

  - in column 4 (alt allele)
  - or by specifying MEINFO=<seqName>, and passing the `--fasta insertionSequences.fasta` command-line argument containing seqName
  - optional coordinates within the insertionSequences.fasta file can be specified as MEINFO=<seqName,start,end,strand>
- **END=end coordinate can optionally be specified to make a compound deletion/insertion event**
  - if END is not specified, it is set to the same value as start

Inversions

- chromosome (column 0)
- start coordinate (column 1)
- SVTYPE=INV;END=<end coordinate> (column 7)

Translocations

- chromosome (column 0)
- start coordinate (column 1)
- SVTYPE=TRA;CHR2=<other chromosome>;END=<end coordinate>;STRAND=<orientation, either + or -> (column 7)

Examples

`events.vcf` (note `.` indicates a field that is ignored by svviz):

```
chr1 2827693   . .   . . . SVTYPE=DEL;END=2828322
chr3 9425916   . . ATGGCTTCGATTAGCGTCGATGCTTCGTAGAGAGTCTGCTA .   .   SVTYPE=INS
chr3 22371722   . . . . .   SVTYPE=INS;MEINFO=L1HS
chr5 46572873   . . . . .   SVTYPE=INS;MEINFO=L1HS,33,5030,-
chr6 36167622   . . TGATCGTCTTTTCTGAGAGCTGCTA .   .   SVTYPE=INS;END=36167671
chr9 458616733   . . . . .   SVTYPE=INV;END=458617412
```

Shell command:

```
svviz --type batch --summary events_summary.tsv -b sample1.sorted.bam hg19.fasta␣
↪events.vcf
```

CHAPTER 8

Frequently Asked Questions

Can I access the web interface if svviz is running on another computer (for example, a server)?

Yes, you should be able to do this by ssh tunneling:

1. Run svviz as you normally would, but define the port using the `--port n` option, where `n` is an integer from 0-65535 corresponding to an unused port on the computer running svviz.
2. From your desktop computer, start an ssh tunnel to your server. If your server is running on myserver.com, and you used port 7777 in step 1., the following command should start the tunnel from localhost:7777 on your desktop to localhost:7777 on the server:

   ```
   ssh -L 127.0.0.1:7777:127.0.0.1:7777 username@myserver.com -N
   ```

   After entering your password for myserver.com, leave ssh running in the terminal.
3. Open your browser to `http://127.0.0.1:7777`.

How do I visualize reads if the breakpoint coordinates are imprecise?

Depending on several factors, you may be able to visualize reads around imprecise breakpoints without doing anything special. This works best if the paired-end insert size is long or the degree of imprecision is small relative to the read length.

However, if you know your breakpoints are imprecise, you can use the `--lenient` command line option to retain reads with poor alignment scores. It is prudent, however, to use the output of svviz to correct the breakpoint positions and re-run svviz once more precise breakpoints have been estimated.

To refine breakpoints, look for a grey region around the alternate allele breakpoint(s), indicating a deletion, or a blue line immediately at the breakpoint, indicating an insertion. The length of the extra deleted or insert sequence can be estimated from the web view by hovering over relevant reads and inspecting the nucleotide-level alignments.

What do I do if the genomic region around the structural variant is repetitive?

If you know the genomic region including the structural variant is repetitive, or if you're getting a warning about "Found a substantial number of reads that could map to multiple locations within the same allele", you should consider doing a few things:

1. When reads are mapped to the genome using a tool such as bwa or bowtie2, they typically receive a "mapq" map-quality score that indicates how uniquely the read maps in the genome; this score can. These scores can be shown for individual reads in the web interface simply by hovering your cursor over a read, with example output being `mapq=60` or (for paired-end reads) `mapq=60,30`.
2. Make sure that you're using the `--min-mapq` or `--pair-min-mapq` options to filter out reads that map ambiguously within the entire genome.
3. Use the `--dotplots` option (see *here* for more information) to visualize sequence similarity within the genomic region.
4. Adjust the `--max-multimapping-similarity` option to filter out reads potentially aligning to multiple locations within the structural variant region (see *here* for more information).

What does the warning "LOTS OF READS IN BREAKPOINT REGION" mean?

The "LOTS OF READS IN BREAKPOINT/MATE-PAIR REGION" warning indicates that svviz will be trying to pull reads in from a genomic region with very high sequencing coverage (over 100,000 reads within the region). This warning may appear when analyzing very high coverage sequencing data or extremely large regions. The warning indicates that run-time or memory to analyze a variant may be excessive.

To skip variants that are particularly large use the `--max-size` option, for example to skip events that are larger than 1 megabase (`--max-size 1000000`). To skip variants with too many reads, use the `--max-reads` option, for example to skip events with more than 100,000 reads (`--max-reads 100000`).

For paired-end data, this warning may also appear as a result of trying to read in mate-pairs from the input BAM(s). svviz first finds all reads in the vicinity of the structural variant being analyzed. For paired-end data, svviz then tries to find the read mates for all reads identified previously. Most of the reads and their mates should both be near the breakpoints. However, in the case of a repetitively-mapping sequence, an incorrectly-called breakpoint, or a mapping error, reads can be separated from their mates in the genome. Thus, to find the mates for these types of reads, svviz must access other genomic locations from the input BAM.

In most cases, svviz should take only a bit longer to find read-pairs, perhaps a few minutes. In the case that it is taking substantially longer, you can cancel svviz by hitting ctrl-c in the terminal. One potential option for skipping highly repetitive reads, thus improving performance, is to adjust the `--min-mapq` (`-q`) option. For example, if the maximum mapping-quality output by your alignment software is 60 (for example, when using bwa), you could specify `-q 60` when running svviz. The related `--min-pair-mapq` option requires that one read end be (1) close to the variant and (2) exceed this mapq threshold. For example, if both read ends have low map quality (eg, fall within a repeat), then the reads will be skipped. If instead one read end maps close to a breakpoint, but the other end maps elsewhere in the genome, to a repeat, then both reads will be retained (eg, in the case of a non-reference repeat). **Can svviz handle large structural variants?**

In theory, svviz can handle arbitrarily large structural variants, however this is limited in practice by memory and processing time. Memory scales with the number of reads being analyzed for the event. Run-time scales with the number of reads, the lengths of the reads and the size of the event. Run-time is also dependent on the I/O bandwidth, processor speed and number of processors.

In the special case of **large deletions**, svviz can visualize them in two ways: (1) visualizing all the reads in the deleted region as well as spanning the deletion breakpoints or (2) in "breakend" mode, where only reads spanning the breakpoints are shown. Because this second breakend mode only analyzes reads near the breakpoints and only aligns against those regions, this mode should be preferred for very large deletions to visualize the exact breakpoints.

To enable this mode, use the `largedeletion` (or in shorthand, `ldel`) event type when running svviz from the terminal. In batch mode, use the `--max-deletion-size` option to convert deletions above a certain size into breakend format. Note that because of the internal architecture of svviz, there is a minimum size for breakend-style events that is relative to the insert size/read length of the input bam (2 times the align distance reported at the beginning of an svviz run).

CHAPTER 9

Change log

1.6.2

fixed a regression that prevented the demo from being run from python2

1.6.1

minor tweaks

1.6.0

Adds support for python 3.x (python 2.7 will continue to be supported as well).

1.5.3

Bugfixes:

- fixed an error with running demos

1.5.2

Improvements:

- added an option `--export-insert-sizes` which exports the insert size distribution plots to png files, one per event per sample.
- added better warning and error messages around command line arguments

- output progress every 30 seconds when not in an interactive shell (eg stderr redirected to file)
- demo data is now downloaded from github

1.5.1

Improvements:

- now supports `<TRA>`-style translocations
- added a new `--sample-reads` option that can be used to downsample reads (read pairs) – use with caution!

Bugfixes:

- removed joblib dependency (joblib 0.10 introduced some API changes that were not compatible with svviz)

1.5.0

Improvements:

- dotplots can now be written to file when in export mode
- added an optional wrapper around the re-alignment module, allowing processing of long-read (eg pacbio) data despite a buggy Smith-Waterman library; specify `--processes -1` to enable this mode
- added `--aln-score-delta` option to allow specifying how different the ref and alt alignments must be in order to choose one over the other (rather than punting the read to ambiguous); this option allows specifying a fraction, which can be useful for sequencing libraries such as pacbio with reads of differing lengths

Bugfixes:

- corrected help messages for structural variant formats

1.4.0

This release includes many small improvements and bugfixes.

One change to svviz's behavior is notable:

- the `--pair-min-mapq` option now requires one read end to both exceed this mapq threshold *and* be near the variant

Improvements:

- implemented a largedeletion variant type, with an option to automatically convert deletions above a certain size to use "breakend" mode, thus only analyzing reads near the deletion breakpoints (see the *FAQ*)
- implemented a `--max-size` option which skips variants exceeding the specified number of nucleotides (see the *FAQ*)
- when the `--max-reads` option is provided in batch mode, svviz should stop analyzing a variant much sooner if that variant exceeds the max-reads threshold
- dotplots now work with multi-part variants such as breakends
- better conversion between chromosome formats (chrX<->X)
- many tweaks to the progress information provided as svviz is running

- no longer require file suffix on `--export` when in batch mode and `--format` is specified (A. Regier)
- added support for using inkscape for PDF export; the `--converter` option can be used to choose between different conversion software packages
- the test suite is now mostly included in the git repository in case anyone else wants to run the regression tests
- added demo 3, a deletion with an example annotation track

Bugfixes: - fixed handling of variant breakpoints near the ends of chromosomes (rpadmanabhan) - fixed a bug where reads mapping to both strands with similar alignment scores might not have been marked as multimapping - fixed a bug which prevented correctly analyzing translocations on the same chromosome on the same strand

1.3.2

This release includes a number of bugfixes and additions to the documentation.

- bed annotations should now work again (C. Lee)
- added support for gff files without gene or transcript IDs
- fixed support for `--processor 1` (A. Ramu)
- added more information about the "lots fo reads in region" warning (see *FAQ*)

1.3.1

This release adds substantial improvements to the handling of multi-mapping reads (ie those aligning to multiple locations near the structural variant). See *here* for more details.

1.3.0

This release adds a number of new features and fixes several bugs:

- added support for displaying gene models (exons and introns) from GFF-formatted annotation files
- added option to display reads that are in flanking genomic regions, providing context for a structural variant
- initial implementation of breakend support (note that, currently, the breakends must be distant from one another, and breakend support has not been implemented from vcf files yet)
- added checkbox to web interface to hide/show flanking reads
- added option to define the web server port, making it easier to use ssh tunneling to access svviz running on a server
- now auto-detect the number of cores available on a machine (used for the realignment step)
- added option to specify how many processes (cores) to use when performing realignment
- improved handling of paired-end reads that align to the same location
- added option to skip variants with very deep read coverage (typically indicative of a repetitive genomic region); useful in batch mode

1.2.0

This is a major feature release, implementing support for visualizing translocations.

Additional changes:

- does a better job finding reads to estimate empirical insert size distribution and read pair orientation
- checks that bam files have index and produce a more helpful error message if they do not
- annotations now also check to see if there's a mismatch between "chrX" and "X" formats, and try to automatically fix it
- wrapping pyfaidx with a pickle-able `GenomeSource` object; should make automated debugging easier
- added `--skip-cigar` option which disables visualizing mismatches and indels; this will speed up exporting and the web browser view for data with many errors (eg PacBio)

1.1.1

- no longer requires X11 if rpy2 is installed (I know, this was a weird one)

1.1.0

- code refactoring and new tests that should make it easier to modify and improve the visualizations produced by svviz
- added experimental support for webkitToPDF, a command-line tool that uses OS X's built-in SVG support (part of Safari's webpage rendering code) to convert SVGs to PDFs; this currently requires a separate install of webkitToPDF. webkitToPDF produces much better PDFs than rsvg-convert does (for example, fonts are converted properly)

1.0.9

- added link to preprint on bioRxiv
- added support for exporting one pdf per event in batch mode
- tweaks and fixes for visualizations
- changed coloring of insertions in reads to cyan

1.0.8

- filter out reads that align multiple times within the region of the structural variant ("multimapping")
- many minor bug-fixes and interface tweaks

1.0.7

- demo data now gets downloaded from Stanford webspace
- added `--version` command line option
- no longer fails if pandas is an older version
- check for librsvg before we do the analysis

1.0.6

- fixed bug that prevented `--export` option from working
- ref and alt alignment scores must differ by at least 2 in order to assign a read to an allele by alignmentScore
- minor bug fixes

1.0.5

- implemented *batch mode* to analyze multiple variants at once

CHAPTER 10

License

`svviz` is distributed under the MIT license:

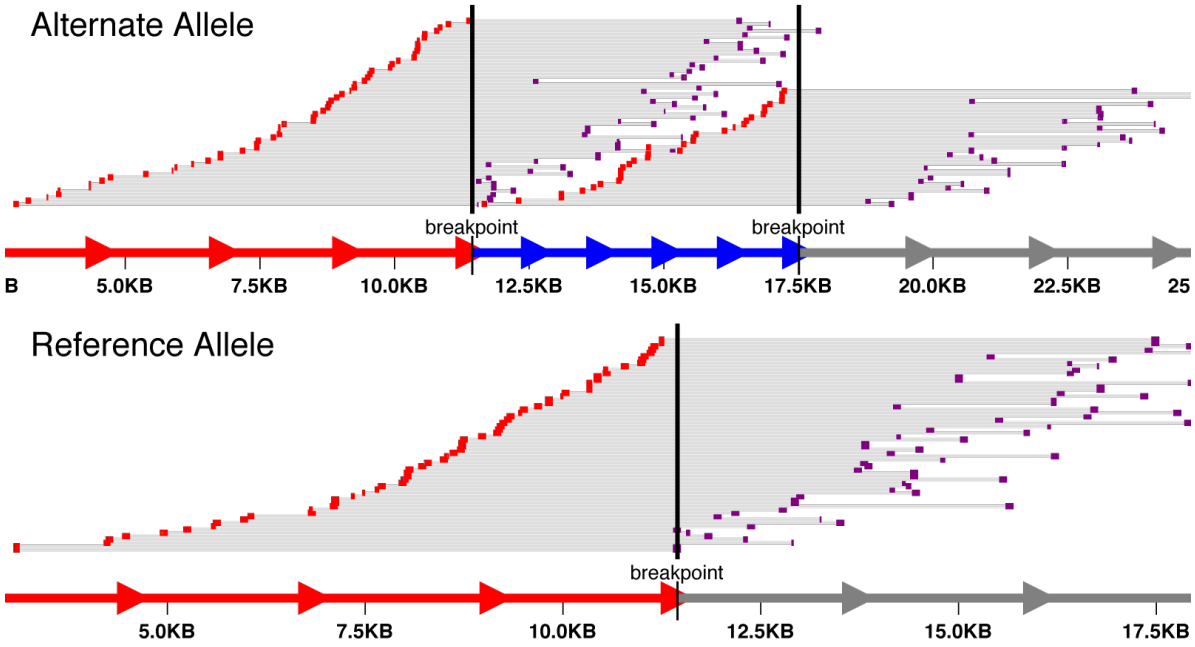
> The MIT License (MIT)
>
> Copyright (c) 2015 Noah Spies
>
> Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:
>
> The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.
>
> THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

The Smith-Waterman Alignments are performed by the Complete Striped Smith-Waterman Library, whose license requires the following statements:

> Authors: Mengyao Zhao & Wan-Ping Lee
>
> Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:
>
> The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

svviz visualizes high-throughput sequencing data that supports a structural variant. svviz is free and open source, available at https://github.com/svviz/svviz. Please submit issues, questions or feature requests using the github issue tracker.

Visit the project page for a visual tour of the features, or continue through this documentation for more detailed information about svviz.



CHAPTER 11

News

Oct, 2016 – svviz 1.6.0 as of version 1.6, svviz adds support for running under python 3.x in addition to python 2.7. I believe that the output should be identical (except for some very insignificant decimal places), but please report any anomalies or problems on the issue tracker.

Oct, 2016 – svviz 1.5.3 svviz 1.5.3 has been released. Upgrade using the following pip command: `sudo pip install --upgrade svviz`. You can see a list of the changes *here*.

CHAPTER 12

Quick-start

1. Install the latest version of svviz from github: `sudo pip install svviz`
2. Run the following command: `svviz demo`

Detailed instructions, including how to ensure that all prerequisites are installed, are available on the *Installation* page.

CHAPTER 13

Citation

svviz has been published in Bioinformatics. If you found svviz useful for your research, please cite svviz as follows:

Spies N, Zook JM, Salit M, Sidow A. 2015. svviz: a read viewer for validating structural variants. Bioinformatics doi:bioinformatics/btv478.

svviz was developed by Noah Spies, a member of the Sidow lab at Stanford and part of the Joint Initiative for Metrology in Biology (JIMB). Funding was provided by the National Institute of Standards and Technology (NIST).