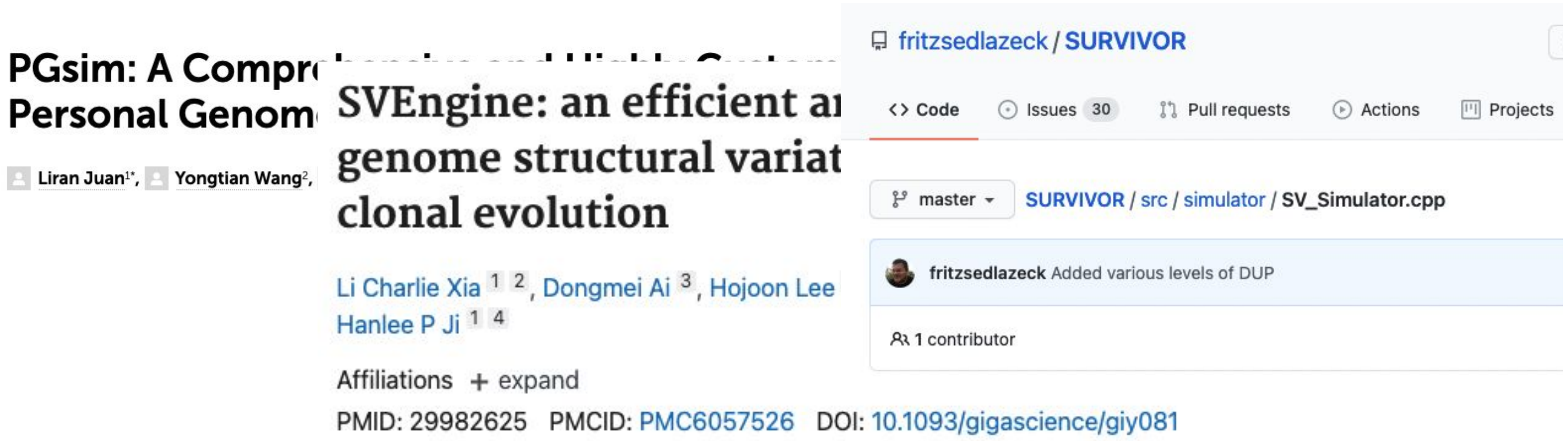
insilicoSV

Nick Jiang & Victoria Popic

Introduction

- insilicoSV is a software to design and simulate complex structural variants, both novel and known.
- Motivated out of a lack of available, flexible SV simulators



PGsim: A Compr
Personal Genom

Liran Juan[1*], Yongtian Wang[2],

SVEngine: an efficient a
genome structural variat
clonal evolution

Li Charlie Xia[1 2], Dongmei Ai[3], Hojoon Lee
Hanlee P Ji[1 4]

Affiliations + expand

PMID: 29982625 PMCID: PMC6057526 DOI: 10.1093/gigascience/giy081

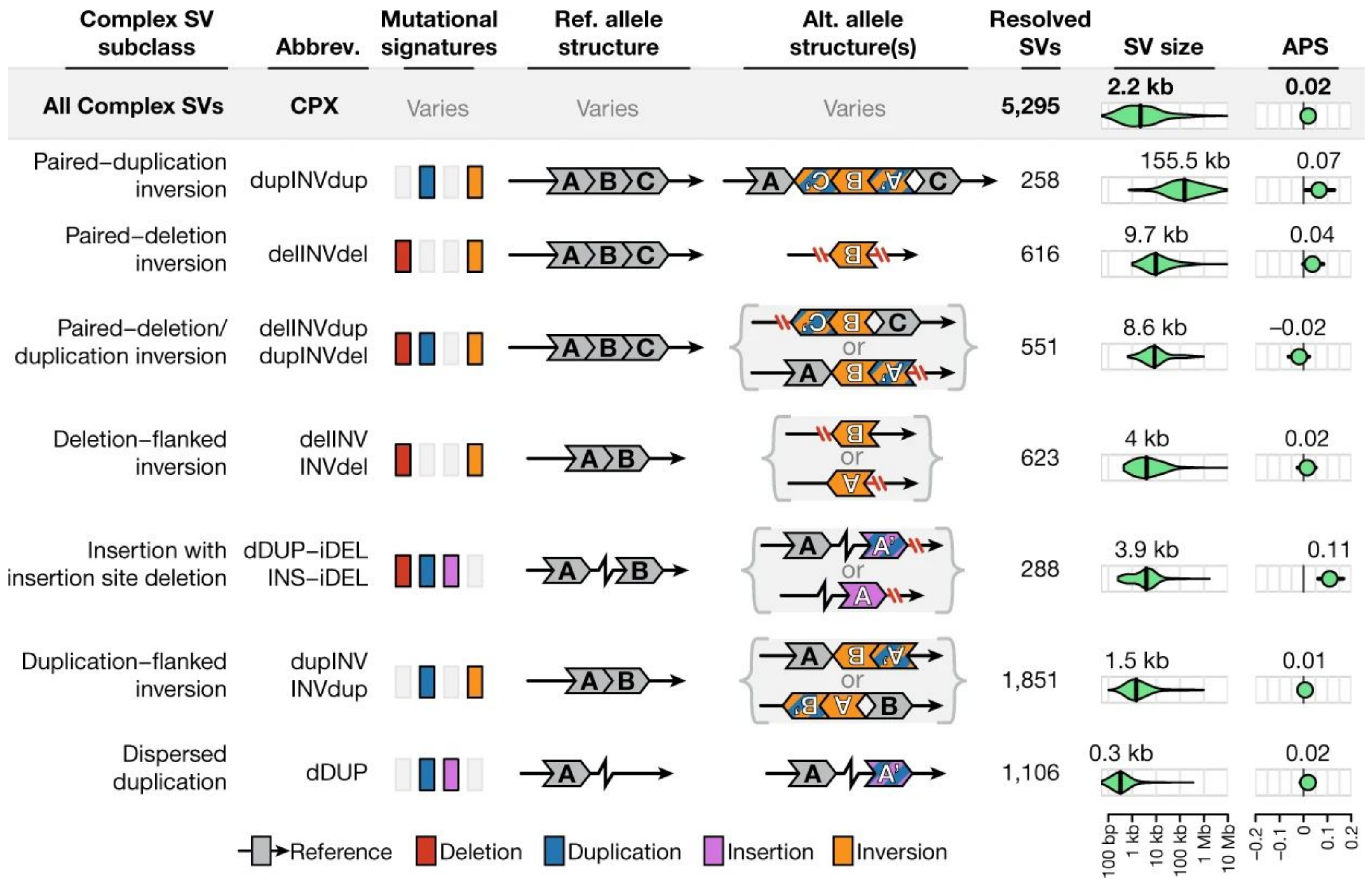
fritzsedlazeck / SURVIVOR

<> Code Issues 30 Pull requests Actions Projects

master SURVIVOR / src / simulator / SV_Simulator.cpp

fritzsedlazeck Added various levels of DUP

1 contributor



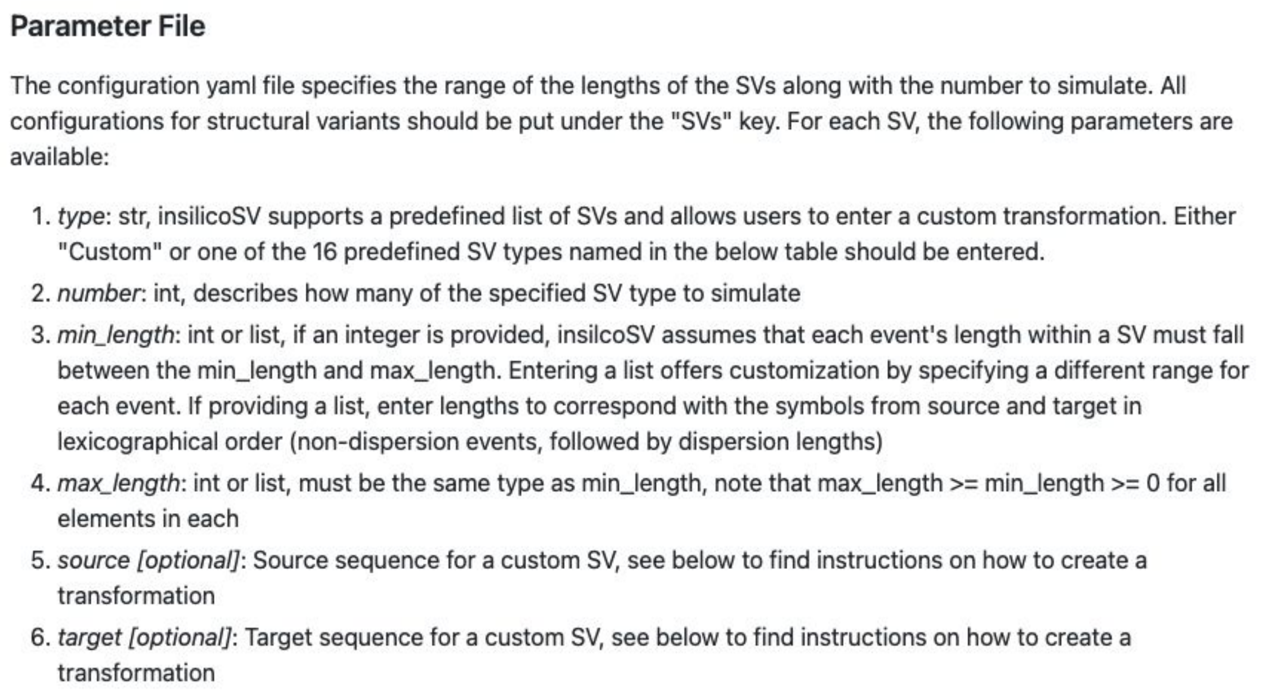
Complex SV subclass	Abbrev.	Mutational signatures	Ref. allele structure	Alt. allele structure(s)	Resolved SVs	SV size	APS
All Complex SVs	**CPX**	Varies	Varies	Varies	**5,295**	**2.2 kb**	**0.02**
Paired–duplication inversion	dupINVdup	🟦🟧	A B C	A C B A C	258	155.5 kb	0.07
Paired–deletion inversion	delINVdel	🟥🟧	A B C	B	616	9.7 kb	0.04
Paired–deletion/ duplication inversion	delINVdup dupINVdel	🟥🟦🟧	A B C	C B C or A B A	551	8.6 kb	–0.02
Deletion–flanked inversion	delINV INVdel	🟥🟧	A B	B or A	623	4 kb	0.02
Insertion with insertion site deletion	dDUP–iDEL INS–iDEL	🟥🟦🟪	A B	A A’ or A	288	3.9 kb	0.11
Duplication–flanked inversion	dupINV INVdup	🟦🟧	A B	A B A or B A B	1,851	1.5 kb	0.01
Dispersed duplication	dDUP	🟦🟪	A	A A’	1,106	0.3 kb	0.02

Reference 🟥 Deletion 🟦 Duplication 🟪 Insertion 🟧 Inversion

Basic Usage Overview

- insilicoSV takes in two input files:
  - the reference genome
  - yaml configuration file
- Following the simulation, it outputs two haplotype files, a BEDPE file, and a basic stats file.

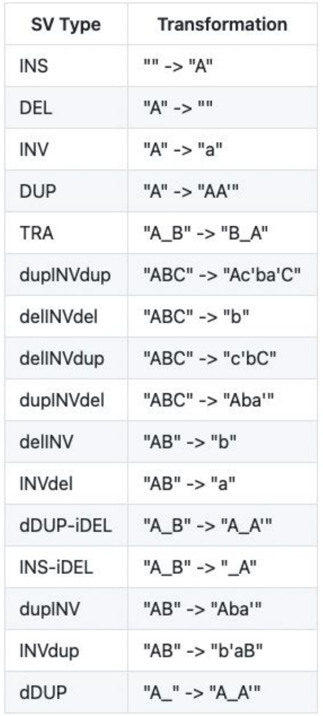
Usage Overview - YAML Config



Parameter File

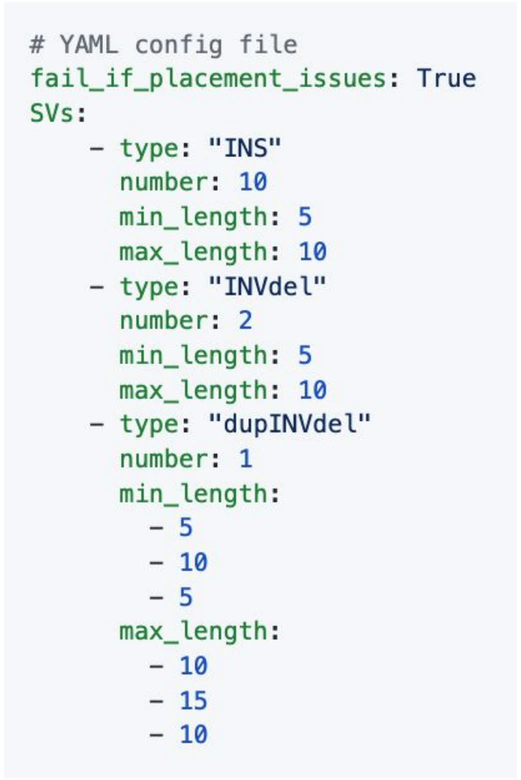
The configuration yaml file specifies the range of the lengths of the SVs along with the number to simulate. All configurations for structural variants should be put under the "SVs" key. For each SV, the following parameters are available:

1. *type*: str, insilicoSV supports a predefined list of SVs and allows users to enter a custom transformation. Either "Custom" or one of the 16 predefined SV types named in the below table should be entered.
2. *number*: int, describes how many of the specified SV type to simulate
3. *min_length*: int or list, if an integer is provided, insilcoSV assumes that each event's length within a SV must fall between the min_length and max_length. Entering a list offers customization by specifying a different range for each event. If providing a list, enter lengths to correspond with the symbols from source and target in lexicographical order (non-dispersion events, followed by dispersion lengths)
4. *max_length*: int or list, must be the same type as min_length, note that max_length >= min_length >= 0 for all elements in each
5. *source [optional]*: Source sequence for a custom SV, see below to find instructions on how to create a transformation
6. *target [optional]*: Target sequence for a custom SV, see below to find instructions on how to create a transformation



SV Type	Transformation
INS	"" -> "A"
DEL	"A" -> ""
INV	"A" -> "a"
DUP	"A" -> "AA'"
TRA	"A_B" -> "B_A"
dupINVdup	"ABC" -> "Ac'ba'C"
delINVdel	"ABC" -> "b"
delINVdup	"ABC" -> "c'bC"
dupINVdel	"ABC" -> "Aba'"
delINV	"AB" -> "b"
INVdel	"AB" -> "a"
dDUP-iDEL	"A_B" -> "A_A'"
INS-iDEL	"A_B" -> "_A'"
dupINV	"AB" -> "Aba'"
INVdup	"AB" -> "b'aB"
dDUP	"A_" -> "A_A'"

Usage Overview - YAML Config Example



```yaml
# YAML config file
fail_if_placement_issues: True
SVs:
    - type: "INS"
      number: 10
      min_length: 5
      max_length: 10
    - type: "INVdel"
      number: 2
      min_length: 5
      max_length: 10
    - type: "dupINVdel"
      number: 1
      min_length:
        - 5
        - 10
        - 5
      max_length:
        - 10
        - 15
        - 10
```

Basic entries:

- Type
- Number
- Min_length
- Max_length

Usage Overview - Custom SVs

- Custom transformation consists of source (ex. ABC or A_B_C) and target sequence (ex. a'AB or A_b'Bc'_C)
- Ex. A_B -> B_A

TTTCTTTAAACACCAGTATTTAGATGCACTATCTAGCTCCCGACAGAGCACTGGTGTC

A _ B

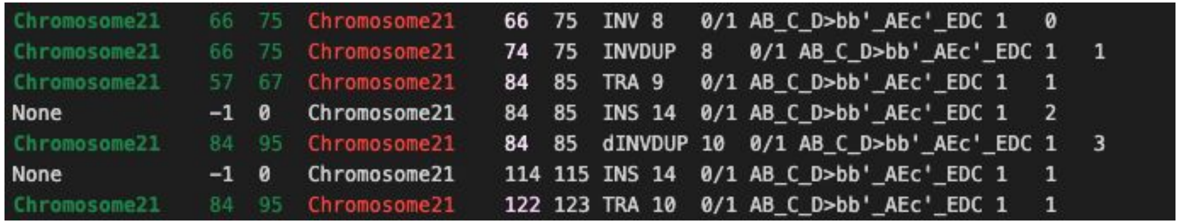
TTTCTTTATGCACTACAGTATTTAGAAACACTCTAGCTCCCGACAGAGCACTGGTGTC

Usage Overview - Syntax for Custom SVs

Name	Symbol	Description
Generic Event	Uppercase letter	The most fundamental organizing tool that maps to a reference fragment
Inversion	Lowercase letter	Ex. a transformation "ABC" -> "abc" will invert "A," "B", and "C" and organize the new fragments as denoted in the target
Duplication	Original symbol + single quotation	Ex. A transformation "ABC" -> "ABA'c" would duplicate "A" after "B" and invert the fragment C.
Dispersion	Underscore	Indicates a gap between the symbols surrounding it
Insertion	Uppercase letter	To add foreign, randomly-generated insertions, use a symbol not present in the source to the target sequence.

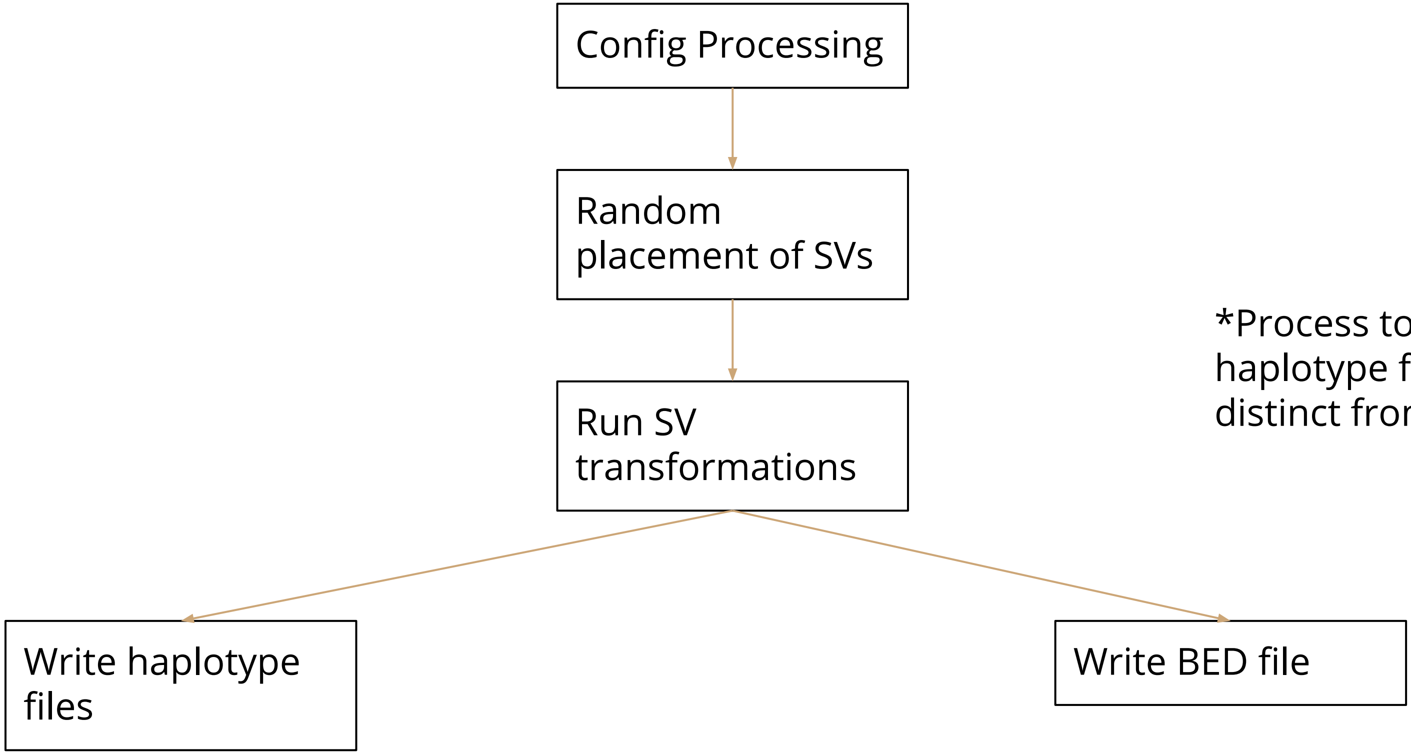
Usage Overview - BED File

1. *source_chr*: The source chromosome of the event
2. *source_start*: Start position on the source_chr [INCLUDE at pos], zero-based indexing
3. *source_end*: End position on the source_chr [EXCLUDE at pos], one-based indexing
4. *target_chr*: The target chromosome of the event
5. *target_start*: Start position on the target chr [INCLUDE at pos], zero-based indexing
6. *target_end*: End position on the target chr [EXCLUDE at pos], one-based indexing
7. *event_type*: Describes the transformation made by the event, either an INS, DEL, INV, TRA, DUP, INVDUP, or INVTRA. Dispersed duplications--those that do not occur immediately after the original--have an attached "d" at the front.
8. *event_size*: Size of the reference fragment impacted by the event
9. *parent_sv*: Describes the parent SV the event is a component of, for instance "dupINVdup." If a custom SV was provided, the name becomes "source>target"
10. *nth_sv*: int, index to count up each SV (note: not the events). All events of a SV belong in the same index.
11. *order*: int, for insertion-like operations such as TRA, INS, or DUP, the "order" index describes in which order events that target the same position were compiled. Events with INV and DEL operations have an order of 0.



```
Chromosome21    66  75  Chromosome21    66  75  INV 8   0/1 AB_C_D>bb'_AEc'_EDC 1   0
Chromosome21    66  75  Chromosome21    74  75  INVDUP  8   0/1 AB_C_D>bb'_AEc'_EDC 1   1
Chromosome21    57  67  Chromosome21    84  85  TRA 9   0/1 AB_C_D>bb'_AEc'_EDC 1   1
None            -1  0   Chromosome21    84  85  INS 14  0/1 AB_C_D>bb'_AEc'_EDC 1   2
Chromosome21    84  95  Chromosome21    84  85  dINVDUP 10  0/1 AB_C_D>bb'_AEc'_EDC 1   3
None            -1  0   Chromosome21    114 115 INS 14  0/1 AB_C_D>bb'_AEc'_EDC 1   1
Chromosome21    84  95  Chromosome21    122 123 TRA 10  0/1 AB_C_D>bb'_AEc'_EDC 1   1
```

insilicoSV - Behind the Scene



*Process to write haplotype files distinct from BED

SV random placement

- Steps
  - First choose a random chromosome and position to simulate
  - Check that non-dispersion events do not overlap with previously placed SVs
  - Store intervals of non-dispersion events to be compared later

TTTCTTTAAACACCAGTATTTAGATGCACTATCTAGCTCCCGACAGAGCACTGGTGTC

A _ B

A _ B

Running SV transformations

Steps:

- Split up target sequence into “blocks,” which are groupings of symbols between dispersion events
- Within each block, use mapping of symbol-to-event to construct a replacement fragment based on the target
- Compile a list of changes formatted as (block_start, block_end, new_frag)

BED Export Process

Steps:

- Within each block of the target sequence, iterate symbol-by-symbol to identify the sub-transform made at this moment
    - Ex. A’ indicates a duplication, while a indicates an inversion
- Keep track of an “order” index by incrementing the value for insertion-like operations and resetting to 0 when an original symbol is encountered
- Export in the format previously described