**Morphling user guide**

Morphling, a rapid and accurate model-free SVs discovery framework for next generation sequencing data. The proposed framework does not rely on any models, but directly mine sequential frequently occurred combination of local enriched alignment anomalies from the raw alignment data and identify SV breakpoint regions. Essentially, Morphling converts the abnormal alignment data into a novel database through different mutational signal channels. And this database allows the model-free framework to discover SV breakpoint regions based on sequential pattern mining. Usually for a 120Gb BAM file (whole genome sequencing), it only takes 1.5h to report all predicted SV breakpoint regions on a desktop. The program is tested on Mac with 16Gb memeory, 3.3GHz Intel Core I5 processor.

**Current version:**

MorphReleaseV1.0

**Dependency:**

* Htsjdk: A Java API for high-throughput sequencing data (HTS) formats. <https://github.com/samtools/htsjdk>.
* Numpy: used in Python script for BAM parameter estimation.

**Download and install:**

* We provide an executable JAR file for command line usage, you don’t have to build the source code.
* A user-interface is provided for non-experience Linux/Unix users. It is an executable JAR file, which can be launched directly.

**Usage:**

* First run python script to generate a BAM configuration file at the same location with your BAM file. You need to specify how much standard deviation (-X) away are considered as abnormal insert size and the number of samples (-N) you would like to use for the estimation.

Example usage: samtools view file.bam | your/path/to/bamConfig.py –X 3 –N 30000

* Mode one: run with BAM.
* Mode two: run without BAM, this mode only requires Super-Item file created at mode one. Therefore, if you want to re-run your program with masked regions or with different parameters, you only need to run mode two.

**Command Line:**

* Get help information of the program:

Java –jar MorphReleaseV1.jar

* Mode one example:

Java –jar MophReleaseV1.jar bamFile=file.bam faFile=file.fa bamCfg=bam.cfg itemOut=item.txt svOut=sv.out regionMask=region.bed

* Mode two example:

Java –jar MophReleaseV1.jar faFile=file.fa bamCfg=bam.cfg itemOut=item.txt svOut=sv.out regionMask=region.bed

**Output format:**

The SV output file contains predicted SV position on the genome. Additional information includes SupType, Pattern, Region (genome region spanned by pattern), weights, ratio (allele fraction of each Super-Item), orientation (orientation of reads in Super-Item). A single SV can be supported by more than one evidence, more evidence indicates more confident calls.

* SupType=ARP\_Span: indicates SV is combined by two patterns that is able to link together through read-pair. Each pattern of the SV might be a breakpoint. Number of read pairs support such relation is provided.
* SupType=Self: a pattern is self-linked through read pairs. Then we estimate potential breakpoint based on abnormal read pairs. Number, quality and weight of these supporting read pairs is provided.
* SupType=Split: indicates SV is discovered based on split alignment. We provide additional information, such as number of split read support, split read mapping quality.
* SupType=Cross: indicates SV is discovered based on local sequence cross links. Additional information includes number of reads support the cross, the maximum cross matched sequence length.
* SupType=Realign: for region with multiple clipped Super-Items, we usually do realignment, this helps discover INDELS and small SVs. Information includes minus and plus strand support read is provided.
* SupType=OEM: one-end-unmapped reads formed cluster may indicate potential insertion breakpoint near OEM Super-Item. This is not a very confident evidence, but we report such abnormal.