Pindel Implementation Overview

Akash Singh - 5156416

Hello! This document contains an overview of the way this pindel implementation works and instructions to run this on python.

There are 2 python files in this package – *PindelAdmin* and *PindelFunctions*. *PindelFunctions* contains the function which finds minimum and maximum unique substrings within the reference genome for a given sequence. *PindelAdmin* sequentially goes through all read pairs to look for insertion/deletion events. For more details, please refer to the documentation in the files themselves.

Instructions to run this implementation:

- Unzip the Pindel.zip file in the folder *"folder_location/folder_name"*
- Save the input files (a file for the reference genome and a SAM file, both in .txt format) in the above folder. Format should be the same as for the input files present in the folder. In other words, the reference genome file should have a single string of nucleotides, all pasted in one line (no gaps) and the SAM file should have the one row for each read, where different columns in a given row are tab-separated. Directly copying the text from the files present on github page of the course and pasting them in a text file will achieve this format.
- Alternatively, the folder contains the files for both the small and large test cases. The file names are self-explanatory.
- Import the contents of this folder as a python project in an IDE.
- Open PindelAdmin.py from the project files.
- Change the name of reference genome file in the line: ***'referenceGenome: TextIO = open("PindelReferenceLarge.txt")'***
- Change the name of SAM file in the line: ***'mainFile: TextIO = open("SamFileLarge.txt")'***
- Enter the number of read pairs, the read length, the maximum insertion size, and maximum deletion size in totalReadPairs, readLength, insertSize, and maxDeletionSize respectively (the values corresponding to the larger test files are pre-entered).

The output will report the deletion events (breakpoints and number of reads supporting the event), followed by insertion events (breakpoints, insertion length, and number of reads supporting the event).