# Users' Manual of PoolHapX (Version 1.0)

## Preamble

The PoolHapX program reconstructs haplotypes within-host from pooled-sequencing data by integrating population genetic models (statistical linkage disequilibrium) with genomics reads (physical linkage). It approximate the resolution of single-cell sequencing using only pooled sequencing data, enabling within-host evolution analyses.

The workflow of PoolHapX is briefed as follows: (A) PoolHapX first determines locations of physical linkage uncertainty using sequencing reads, and then divides the full genome into smaller regions. (B) Regional haplotypes are solved for and joined together using a statistical model for a parsimonious global distribution of haplotypes. C) The within-pool frequency of each haplotype is estimated by regularized regression to solve for each within-pool haplotype distribution.

Installation and a simple example are described below. Users can get the final within-host (or within-pool) frequencies of each haplotype by running the functions step by step: "script", "format", "gc", "aem", "l0l1". ScriptForPHX.jar generates all commands required by PoolHapX in a script so users can run the script easily. More description and benchmarking of PoolHapX can be found in our publication:

URL of the bioarchive.

## Installation

PoolHapX is a batteries-included JAR executable. All needed external jar packages are included in the downloadable, poolhapx.jar. However, as we used an R package L0Learn, the users have to install R and L0Learn (https://cran.r-project.org/web/packages/L0Learn/index.html). The versions of R and R package L0Learn that we have used on our platform are: version 1.2.0 for L0Learn and version 3.6.1 for R. Other versions are not tested, although they may work. Users are also expected to have java (version: 1.8) on their platform. Longranger (version: 2.2.2) should be installed if processing 10x linked-reads.

Several other tools are prerequisites for running. PoolHapX. Users can download and install them from the websites:

bwa (if using paired-end reads): https://github.com/lh3/bwa
samtools 0.1.19+: http://www.htslib.org/download/
GATK 4.1: https://software.broadinstitute.org/gatk/download/index
longranger (if using 10x linked-reads): https://support.10xgenomics.com/genome-exome/software/pipelines/latest/installation

## Functions

script: the “script” function generates a script which contains all commands. Users can run the project_name.cmd to get the final results calculated by PoolHapX from the initial FASTQ files.
format: the “format” function generates the vef file (vef file contains variant sites linking information extracted from BAM files) and calculates the variant frequency at different sites for different pools from the VCF/SAM files.

gc: the “gc” function generates the graph coloring result from the vef file.
aem: the “aem” function first divides the variation sites into several regions from the graph coloring result; after that the aem function infers haplotypes in local regions for different levels using hierarchical AEM algorithm.

l0l1: the “l0l1” function finalizes the identity and frequency of the global haplotypes using L0L1 regulated regression.

## Quick start with included example data

Example data is provided. After decompressing the downloadable, users can see the reference folder, fastq files folder, fastq_file.txt, and config.properties under the “Example” folder. After updating absolute paths of executable (such as bwa etc) and parent folder in the config.properties file, users can run PoolHapX by a simple commands:

Usage:

```
> java -jar  PoolHapX.jar script config.properties
```

Then go to the folder of /PATH/TO/Working_dir/cmd and run:

```
> ./<project_name>.cmd
```

Users will then generate the final haplotype results for each pool at the “output” folder under their working directory.

## Full Manual

### Data Preparation

### Check fastq_name file format

Please put all FASTQ files under the same directory. All paired-end fastq files have to be named as: sample_id.read1.fastq and sample_id.read2.fastq. Write the name of all your fastq files into a single file following the format below:

```
# header information such as project name
<sample1>.read1.fastq     <sample1>..read2.fastq
<sample2>.read1.fastq     <sample2>.read2.fastq
<sample3>.read1.fastq     <sample3>.read2.fastq
<sample4>.read1.fastq     <sample4>.read2.fastq
<sample5>.read1.fastq     <sample5>.read2.fastq
<sample6>.read1.fastq     <sample6>.read2.fastq
<sample7>.read1.fastq     <sample7>.read2.fastq
<sample8>.read1.fastq     <sample8>.read2.fastq
<sample9>.read1.fastq     <sample9>.read2.fastq
<sample10>.read1.fastq    <sample10>.read2.fastq
```

*Each row is an observation(sample), and each name is separated by tab

For 10x linked reads, the folder name for each pool should be listed in the file:

```
# header information such as project name
sample1
sample2
sample3
sample4
sample5
sample6
sample7
sample8
sample9
sample10
```

**Check config file format (configure to your setting)**

```
#config_file
Main_Dir = /PATH/TO/PoolHapX_work_dir
Project_Name = Test
Java = /PATH/TO/java
samtools = /PATH/TO/samtools
gatk = /PATH/TO/gatk
PHX_JAR = /PATH/TO/PoolHapX.jar
#If Sequencing_Technology is 10x_linked-reads, users may leave the parameter "bwa" blank.
```

```
bwa = /PATH/TO/bwa
#If Sequencing_Technology is paired-end_reads, users may leave the parameter "longranger" blank.
longranger = /PATH/TO/longranger
#/PATH/TO/longranger_ref_folde is generated by the "mkref" function of longranger. If Sequencing_Technology is paired-end_reads, users may leave the parameter "Longranger_Ref_Folder" blank.
Longranger_Ref_Folder = /PATH/TO/longranger_ref_folder
Fastq_Path = /PATH/TO/fastq_file_dir
Fastq_File = /PATH/TO/fastq_file.txt
Ref_Path = /PATH/TO/HIV_HXB2.fa
Rscript_Path = /PATH/TO/Rscript
Sequencing_Technology= 10x_linked-reads or paired-end_reads
```

* config file included in the package is configured to sample set

**Generate the script**

Command:

```
> java -jar  PoolHapX.jar script <config file>
```

The script for running PoolHapX will be generated under “/PATH/TO/Working_dir/cmd/”. The name of the file will be “<project_name>.cmd”

**Run PoolHapX**

```
> ./<project_name>.cmd
```

**Output format:**

The output for each step, i.e., graph coloring, aem and l0l1 regression, is generated under “/PATH/TO/Working_dir/<project_name>/intermediate/”. The final output is generated under “/PATH/TO/Working_dir/<project_name>/output/”. Under the folder “output”, each pool (host) has a folder. Within its folder, one can find the haplotype frequencies in this pool the file named as “final_freq_haps.txt”.
The format of the frequencies is exampled below:

| Hap_ID | h0 | h1 | h2 | h3 | h4 | … |
|---|---|---|---|---|---|---|
| Freq 0.02870493 | 0.02202937 | 0.03137516 | 0.06108144 | 0.12449933 | … | |
| 0;211;211;0:1 | 0 | 1 | 1 | 1 | 0 | … |
| 0;231;231;0:1 | 1 | 0 | 0 | 0 | 0 | … |
| 0;310;310;0:1 | 0 | 1 | 1 | 1 | 0 | … |
| … … | | | | | | |

The first row lists the haplotype IDs. The second row lists the frequencies of each haplotype. The first column denotes the ID of the genetic variants in the format of chromosome-ID; start-position; end-position; alleles. In the event of some viruses that have not chromosome number, PoolHapX will use 0 to denote the chromosome ID. For SNPs, the start-position and end-position are the same. In the rest of the file, each column represents the composition of the haplotypes, i.e., the alleles at each location.

## PoolHapX properties file

We provide default parameters for users ("/PATH/TO/Working_dir/<project_name>/input/PHX.properties"). Under most circumstances, the default parameters work well when compared with other existing tools. However, in the event that users may want to make change to the parameters themselves, the properties file is located under input directory, named as "PHX.properties". We list the meaning for all the parameters below (explanation of each parameters as well as their ranges). Users can modify these parameters according to their needs.

# PoolHapX Parameters

##########
## The name of the project, will be the prefix of names of cross-pool files.
**Proj_Name** = project_name

## File locations: input directory; output files directory; intermediate files directory; gold standard files directory.
**Input_Dir** = /PATH/TO/Input_Dir
**Intermediate_Dir** = /PATH/TO/Intermediate_Dir
**Output_Dir** = /PATH/TO/Output_Dir
## If users do not have the gold standard files, please just leave the parameter blank
**Gold_Dir** = /home/chencao/Desktop/PoolHap/freq_0_pool_25_dep_100/gold_standard

##########
### Graph-Colouring: link all reads to generate candidate global haplotypes based on physical linkage.

## Maximum number of positions in a window. [Default 20: Range: 1 - 100]
**Num_Pos_Window** = 20

## Maximum(?) number of gaps in a window. [Default 2: Range: 1 – 20]
**Num_Gap_Window** = 2

##########

### Divide-and-Conquer: divide the genome into multiple regions based on linkage uncertainty.

## Proportion of raw GC-haplotypes that contain the gap in the pool. [Default: 0.6, Range: 0 - 1]
**In-pool_Gap_Support_Min** = 1

## Proportion of raw GC-haplotypes that contain the gap across all pools. [Default: 0.1, Range: 0 - 1]
**All-pool_Gap_Support_Min** = 1

## Minimum number of SNPs in a Level 1 region. [Default: 10, Range: 8 - 12]
**Level_1_Region_Size_Min** = 10

## Maximum number of SNPs in a Level 1 region. [Default: 14, Range: 10 - 14]
**Level_1_Region_Size_Max** = 12

## Minimum number of SNPs in a Level 1 tiling region. [Default: 10, Range: 8 - 12]
**Level_1T_Region_Size_Min** = 10

## Maximum number of SNPs in a Level 1 tiling region. [Default: 14, Range: 10 - 14]
**Level_1T_Region_Size_Max** = 12

## Estimated number of individuals in a pool. [Default: 1000000, Range: 1000 - 1000000]
**Est_Ind_PerPool** = 1000000

## Number of maximum mismatch positions in constructing Level 2. [Default: 1: Range: 0 - 2]
**Level_1_2_Region_Mismatch_Tolerance** = 1

## Number of maximum mismatch positions in constructing Level 3. [Default 2: Range: 1 - 3]
**Level_2_3_Region_Mismatch_Tolerance** = 2

## Number of maximum mismatch positions in constructing Level 4. [Default 2: Range: 3 - 7]
**Level_3_4_Region_Mismatch_Tolerance** = 5

## Number of AEM levels. [Default 4: Range: 1 - 4]
**AEM_Maximum_Level** = 4

## Number of maximum mismatch positions in BFS. [Default 6: Range: 4 - 8]
**BFS_Mismatch_Tolerance** = 6

##########
### Approximate Expectation-Maximization: generate regional haplotype sets and their frequencies.

## Maximum number of iterations regardless of convergence. [Default: 200, Range: 50 - 1000]
**AEM_Iterations_Max** = 200

## The epsilon that controls the stop criteria of AEM (i.e. convergence). [Default: 0.00001, Range: 0 - 0.000001]
**AEM_Convergence_Cutoff** = 0.00001

## For each iteration of AEM, some very rare haplotypes with frequencies below this parameter will be set to a frequency of zero. [Default: 0.00001, Range: 0.0 - 0.000001]
**AEM_Zero_Cutoff** = 0.00001

## Initial value for regional cross-pool frequency cutoff immediately after AEM. [Default: 0.01, Range: 0.0 - 0.05]
**AEM_Regional_Cross_Pool_Freq_Cutoff** = 0.01

## Maximum number of regional haplotypes in a region for AEM. [Default: 50, Range: 1-200]
**AEM_Regional_HapSetSize_Max** = 50

## Minimum number of regional haplotypes in a region for AEM. [Default: 5, Range: 1-20]
**AEM_Regional_HapSetSize_Min** = 3

## If both denominator and numerator are very close to zero, the Importance Factor (IF) value. [Default: 5.0, Range: 1.0-10.0]
**IF_0_0** = 0.1

## if the denominator is close to zero but the numerator is not, the IF value. [Default: 50.0, Range: 10.0-1000.0]
**IF_Denominator_0** = 10.0

##########
### L0L1 Regulated Regression: reduce the number of full-length haplotypes and estimate their frequency using L0L1 regulated regression.

## Path for rscript binary file
**Rscript_path** = /PATH/TO/Rscript

## The maximum weight for the longest distance between two SNPs. [Default: 2.0, Range: >= 1]

**Regression_Distance_Max_Weight** = 2.0

## The max weight for the highest coverage. [Default: 2.0, Range: >= 1]
**Regression_Coverage_Weight** = 2.0

## The weight of the constraint Sigma freq_i = 1, where freq_i is in the in-pool frequency for haplotype_i. [Default: 5.0, Range: >= 1]
**Regression_One_Vector_Weight** = 5.0

## The weight of the constraints Sigma freq_i * h_ij = MAF_j (j is the SNP index and i is the haplotype index). [Default: 2.0, Range: >= 1]
**Regression_Hap_MAF_Weight** = 2.0

## The weight for LD (specifically, the probability for both SNP_k and SNP_j being the alternate allele). [Default: 1.0, Range: >= 1]
**Regression_Hap_LD_Weight** = 1.0

## Number of maximum mismtach positions for linking regression regional regions using AEM level. [Default 7: Range: 2 - 12]
**Regression_Mismatch_Tolerance**= 7

## Number of maximum selected haplotypes to generate higher level potential haplotypes for following regression. [Default 25, Range: 10 - 50]
**Maximum_Selected_HapSetSize** = 25

## The minimum regularization gamma penalty for L0L1 regression. [Default: 0.0001, Range: 0 - 1]
**Regression_Gamma_Min** = 0.0001

## The maximum regularization gamma penalty for L0L1 regression. [Default: 0.1, Range: 0 - 1]
**Regression_Gamma_Max** = 0.1

## The number of gamma values beween Regression_Gamma_Min and Regression_Gamma_Max for L0L1 regression. [Default 10: Range: 2 - 20]
**Regression_n_Gamma** = 10

## Number of maximum regions for each step of L0L1 regulated regression divide and conquer. [Default 3, Range: 2, 3]
**Regression_Maximum_Regions** = 3

## Sequencing Technology [10x_linked_reads / paired-end_reads]
**Sequencing_Technology** = paired-end_reads

## Number of threads used for parallelly running L0L1 regulated regression. [Default 3: Range: >=1]

**Number_Threads** = 3

## Copyright License (MIT Open Source)