# HAPFLOW

## Visualising haplotypes in sequencing data


### Abstract

HapFlow is a python application for visualising haplotypes present in sequencing data. It identifies variant profiles present and reads and creates an abstract visual representation of these profiles to make haplotypes easier to identify.



Prepared by Mitchell Sullivan and Nathan Bachmann

Mitchell Sullivan
mjsull@gmail.com

# Table of Contents

# 1. General Information

## 1.1 Overview

HapFlow is a python application for visualising haplotypes present in sequencing data. It identifies variant profiles present and reads and creates an abstract visual representation of these profiles to make haplotypes easier to identify.

## 1.2 Description

HapFlow is an application for visualizing hapolotypes present in reads. It is written in python and uses the Tkinter windows system. It is divided into 2 parts.

HapFlow-generator creates a profile of variants present in each read. If the variant profile is unique, a flow (profile of variants in a read) is created. If the flow already exists in another read, the count of the flow is incremented by one.

HapFlow-viewer displays the created flow file on the canvas of the GUI. An orange rectangle within the blue rectangle represents the portion of the genome currently being displayed. Underneath, an orange rectangle with vertical lines represents where the variants are located within the displayed section of the genome, these lines are extended below and spaced an equal distance apart in the area where the flows are viewed. Each flow consisting of one or more reads is represented as one or more arrows overlapping each variant line that the reads of the flow align to. Width of the arrow represents the number of reads within that flow. Variants on the same read of a pair are joined by a solid line, variants on different reads of a pair are joined by a dotted line. Arrows grouped at the top of the canvas represent the most common variant, the second group of flows represents the second most common variant and so on. The last group represents potential sequencing, alignment or variant calling errors as they have sequence that has not been called as a variant. Information about the sequence of each variant is represented underneath the flows. The canvas is scrollable and zoomable allowing the user to easily navigate through whole genomes

HapFlow attempts to group flows according to consensus (groups indicated by hue), this function is still experimental.

## 1.3 Acronyms

SAM – Sequence Alignment Map (format)
BAM – Binary Alignment Map

## 1.4 Citing HapFlow

Sullivan MJ, Bachmann NL, Timms P, Polkinghorne A. (2015)
HapFlow: Visualising haplotypes in sequencing data
PeerJ PrePrints 3:e1105
https://dx.doi.org/10.7287/peerj.preprints.895v1

# 2. Installation

## 2.1 Binaries

Windows, OSX and GNU/Linux executables are available from the HapFlow website (mjsull.github.io/HapFlow). If these do not work on your computer please let us know. HapFlow can also be run as a python application.

## 2.2 Windows

There is limited windows functionality due to pysam compatibility issues with Windows. If you have managed to get pysam working with windows please let us know how and we will update this document.

If you have created a flow file they can be viewed on windows by running HapFlow.exe.

## 2.2 Running from source / Command-line

Ensure pysam version 0.8.1 and Tkinter are installed. If you wish to create SVG images of hapflow canvassvg must also be installed. Both pysam and canvassvg are pip installable. Then download HapFlow.py and run as a python application.

# 3. Getting Started

## 3.1 Workflow

Hapflow uses a graphical user interface (GUI) for inputting and outputting files. The input files that Hapflow requires are a BAM file and a VCF file.

This section describes the workflow for using Hapflow with paired-end Illumina reads by creating a sorted BAM file and VCF file.

**1) The reads first need to be aligned against a reference genome to create a sorted BAM file. This can done using the read mapping BWA to create a SAM file:**

% bwa mem reference.fasta reads.fq > aln.sam

**2) Samtools can be used convert the SAM file to a sorted and indexed BAM file:**

% samtools faidx reference.fasta

% samtools import reference.fasta.fai aln.sam aln.bam

% samtools sort aln.bam aln.sorted

% samtools index aln.sorted.bam

**3) Freebayes is the recommended software for creating a VCF file to be used with HapFlow**

% freebayes –F 0.05 –p 10 –f reference.fasta aln.sorted.bam > aln.vcf

**4) Filtering of low quality variant calls may be needed to reduce noise.**

% vcffilter –f “QUAL > 20” aln.vcf > aln.filt.vcf

**4) Launch Hapflow from the UNIX command line or using an executable.**

% python Hapflow.py

**5) Create the flow file using the sorted BAM and the filtered VCF file.**

File -> Create flow file

**6) Load the flow file.**

File -> Load flow file

# 4. Overview of the GUI

## 4.1 File Menu

### 4.1.1 Create Flows

Selecting “create flow file” will open a menu where the sorted BAM file and VCF file can be selected as well as designating a location to save the Flow file.

### 4.1.2 Change max. variants

Change the maximum number of variants shown. Flow file must be reloaded after selecting this option. Default is 2 max. variants (3 rows – 2 variants + 1 error).

### 4.1.3 Load Flows

Loads a flow file and visualizes the haplotype profile in the main window.

## 4.2 Tools menu

### 4.2.1 Goto Base

Go to a user selected base position in the haplotype profile.

### 4.2.2 Create Image

Create a Scalar Vector Graphics (SVG) file of the current window.

### 4.2.3 Select OTU

Select which operational taxonomic unit (OTU) to mark. This can be used to mark out a haplotype profile to separate reads.

### 4.2.4 Write OTUs

Write OTUs to BAM files. Select BAM file from which the flow file was created and a prefix for output bam files. Output files will be named prefix.1.bam for the alignments from the first OTU, prefix.2.bam for alignments from the second OTU etc.

## 4.3 View Menu

### 4.3.1 Hide gapped

Hides the lines that represent gaps in the reads

### 4.3.2 Show gapped

Shows the lines that represent gaps in the reads

### 4.3.3 Stretch X

Extends x-axis of the display window.

### 4.3.4 Shrink X

Shrinks the x-axis of the display window.

### 4.3.5 Stretch Y

Extends y-axis of the display window.

### 4.3.6 Shrink Y

Shrinks the y-axis of the display window.

## 4.4 HELP

### 4.4.1 About

Provides version details about HapFlow.

### 4.4.1 Help

Links to the HapFlow Github page.

### 4.4.2 Citing HapFlow

Citation information.

## 4.5 CONTEXT MENUS

Right clicking on a flow will bring up a variety of options.

### 4.5.1 Details

Provides the position in the alignment of the highlighted variant, the direction and number of the reads.

### 4.5.2 Get flow readnames

Outputs the name of the reads in the selected flow as a textfile.

### 4.5.3 Write flow to BAM

Outputs the read of the selected flow as a BAM file.

### 4.5.4 Get group readnames

Outputs the name of the reads in the selected group as a textfile.

### 4.5.5 Write group to BAM

Outputs the read of the selected group as a BAM file.

## 4.6 Key bindings

### 4.6.1 (d) Stretch X

Extends x-axis of the display window.

### 4.6.2 (a) Shrink X

Shrinks the x-axis of the display window.

### 4.6.3 (w) Stretch Y

Extends y-axis of the display window.

### 4.6.4 (s) Shrink Y

Shrinks the y-axis of the display window.

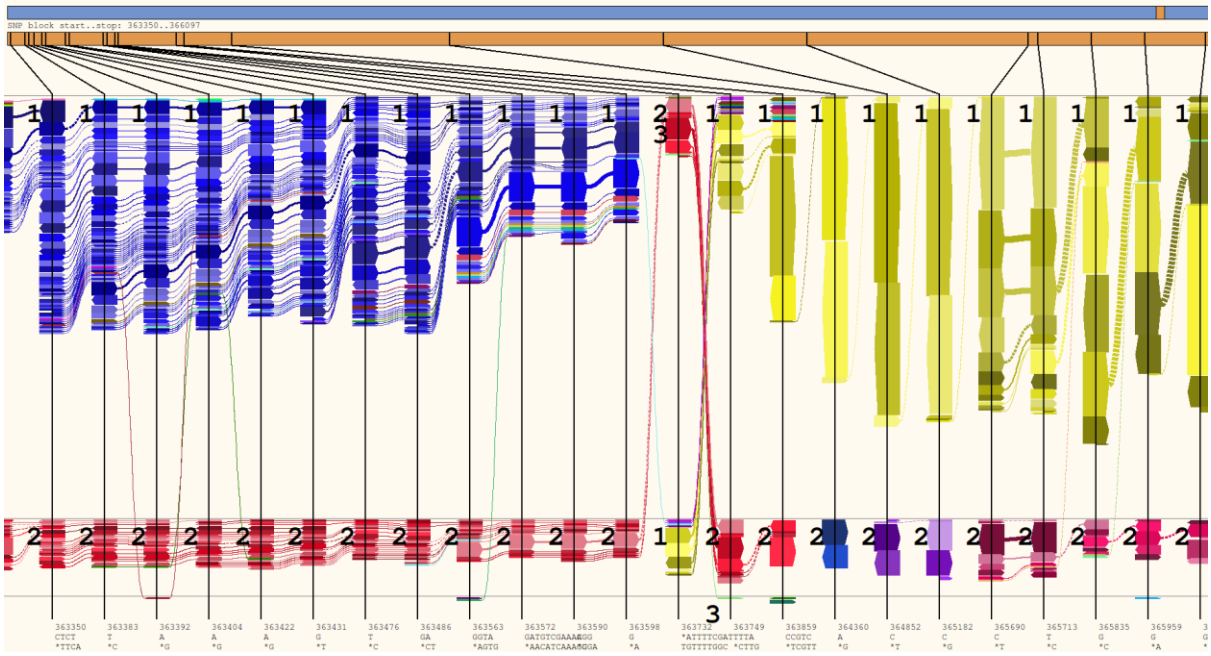
# 5. Tutorial

Tutorial files can be downloaded from http://mjsull.github.io/HapFlow/files.html.

## 5.1 TUTORIAL 1: Real data from mixed-strain infection of Koala.

This tutorial will let you create a flow file from a generated BAM and VCF file and show you how to separate reads into two operational taxonomic units. In the interest of keeping the example files small only reads aligned to bases 360,000..380,000 were used. This tutorial assumes you have downloaded and extracted tutorial1.zip from the HapFlow website.

1) Launch HapFlow by double clicking on the .exe or app file.
2) Load flow by selecting File -> Load flows and then choosing example1.flw
3) Explore the flow file by scrolling from left to right.
4) Hide lines marking flows with gaps by selecting view -> hide gapped.
5) Adjust the amount of variants displayed by pressing A and D on your keyboard.
6) Adjust the height of the flows by pressing W and S on your keyboard.
7) Left click on a Flow to highlight the flow, right click to bring up options relating to the flow.
8) Go to the variant at base 363,350 by selecting tools -> goto base and typing in 363350
9) Define OTU 1 by double clicking variants along the top and middle rows.
10) Define OTU 2 by selecting tools-> Select OTU -> 2 and then double clicking variants along the top and middle rows.
11) Define OTU 3 by selecting tools-> Select OTU -> 3 and then double clicking positions of the flow with an erroneous base.



Chlamydia sample with selected OTUs

12) Write OTUs to BAM by selecting tools -> Write OTUs and then choosing the downloaded BAM file and defining a prefix for output file. Files will be written to prefix.1.bam prefix.2.bam and prefix.3.bam

## 5.2 TUTORIAL 2: Simulated data from 3 strains

This tutorial will cover the approach for generating BAM and VCF for read data containing three *Chlamydia pecorum* strains and visualizing the haplotypes with HapFlow. In the tutorial folder are two FASTQ files containing simulated paired-end Illumina reads based on the complete genome of *C. pecorum* E58 at 20x coverage, *C. pecorum* PV3056 at 10x coverage and *C. pecorum* PV787 at 5x coverage mixed together to represent a mixed infection. The FASTQ files are called "mixed_3strains_R1.fastq" and "mixed_3strains_R2.fastq". Also included in the tutorial folder is a FASTA file of the complete genome *C. pecorum* W73, which will be used as a reference genome.

1) The BAM file can be created by aligning the simulated Illumina reads against the reference genome, *C. pecorum* W37 using BWA and Samtools.

   % bwa index Cpecorum_W37.fasta

   % bwa aln Cpecorum_W73.fasta mixed_3strains_R1.fastq > read1.sai

   % bwa aln Cpecorum_W73.fasta mixed_3strains_R2.fastq > read2.sai

   % bwa sampe Cpecorum_W73.fasta read1.sai read2.sai mixed_3strains_R1.fastq mixed_3strains_R2.fastq > mixed_3strains_bwa.sam

   % samtools faidx Cpecorum_W73.fasta

   % samtools import Cpecorum_W73.fasta mixed_3strains_bwa.sam mixed_3strains_bwa.bam

   % samtools sort mixed_3strains_bwa.bam mixed_3strains_bwa.sorted

   % samtools index mixed_3strains_bwa.sorted.bam

2) The VCF file can be created using Freebayes

   % freebayes -f Cpecorum_W73.fasta -p 3 –F 0.03 mixed_3strains_bwa.sorted.bam > mixed_3strains_bwa.vcf

   -p 3 sets ploidy to 3

   -F 0.03 – only show variants present in more than 0.03 of the reads.

3) Launch HapFlow by double clicking on the executable or from the command-line.

   % python HapFlow.py

4) In the top menu bar, select File -> Create Flow File and load "mixed_3strains_bwa.sorted.bam" in the BAM file box and "mixed_3strains_bwa.vcf" for the VCF file box. Save the output as "mixed_3strains_bwa.ftw." This step may take a few minutes.

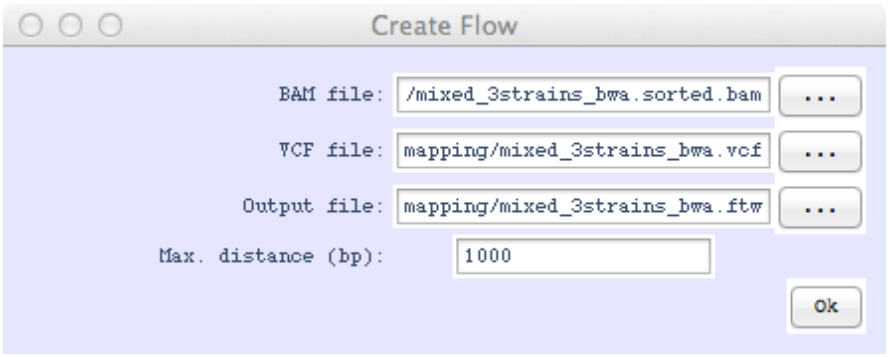



Figure 1: The Create Flow menu is where the BAM and VCF file make the flow file.

5) Select File -> Load Flow File and load "mixed_3strains_bwa.ftw". The following screen will appear in a few seconds. The x-axis can be extended by selecting View -> Stretch X or by hitting "D" on the keyboard.

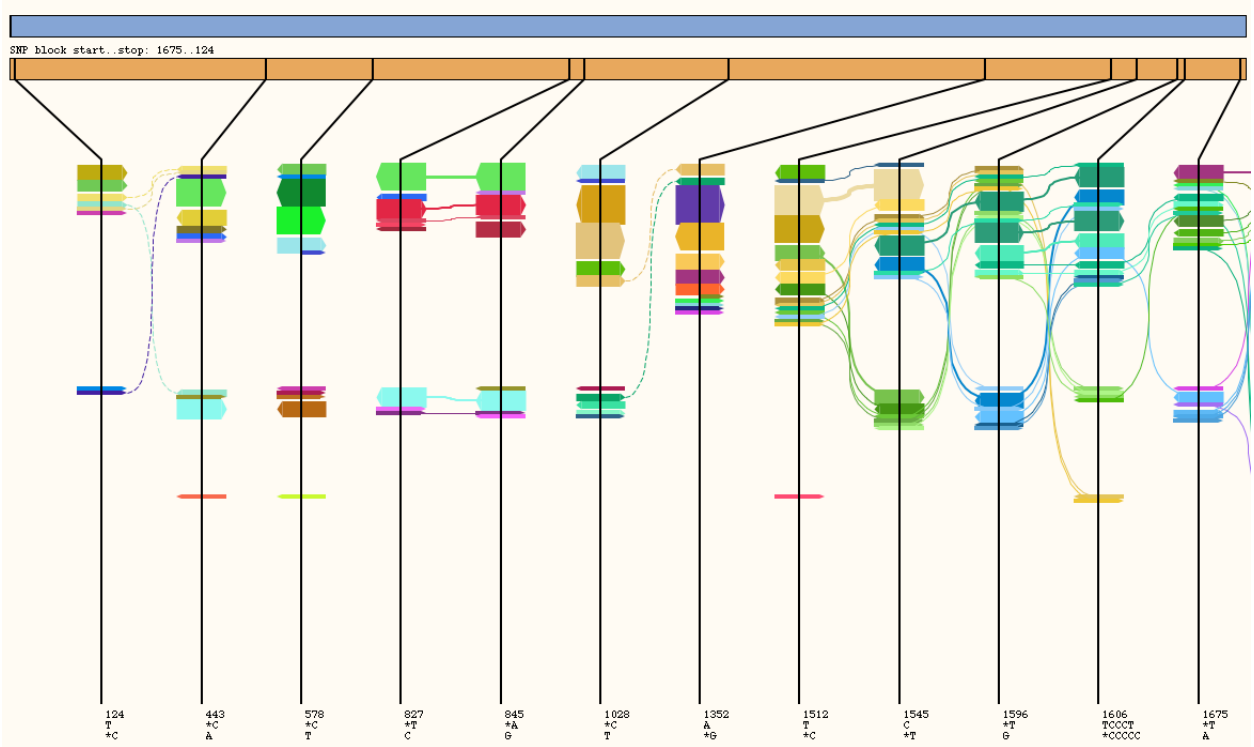



Figure 2: Hapflow diagram of simulated Illumina read data.

6) The orange rectangle with vertical lines represents where the variants are located within the displayed section of the genome, these lines are extended below and spaced an equal distance apart in the area where the flows are viewed. Each flow consisting of one or more reads is represented as one or more arrows overlapping each variant line that the reads of the flow align to. Width of the arrow represents the number of reads within that flow. A solid line joins variants on the same read of a pair.

7) Click on the arrows to highlight individual flows. Figure 3 show a section of the HapFlow profile where the three strains are visualised as flows. Figure 3A-C shows the flows containing a different combination of three single nucleotide polymorphisms (SNPs). Figure 3A represent *C. pecorum* E58 the most dominate strain (20x coverage) in the read data; hence why this flow has the thickest arrows of the three strains. Figure 3B show a flow associated with *C. pecorum* PV3056 the second most prevalent strain (10x coverage). Figure 3C marks the flow for *C. pecorum* P787, which is the least prevalent strain (5x coverage). Right click on the flow will display the options to retrieve the read names for flow or extract the reads as a BAM file.

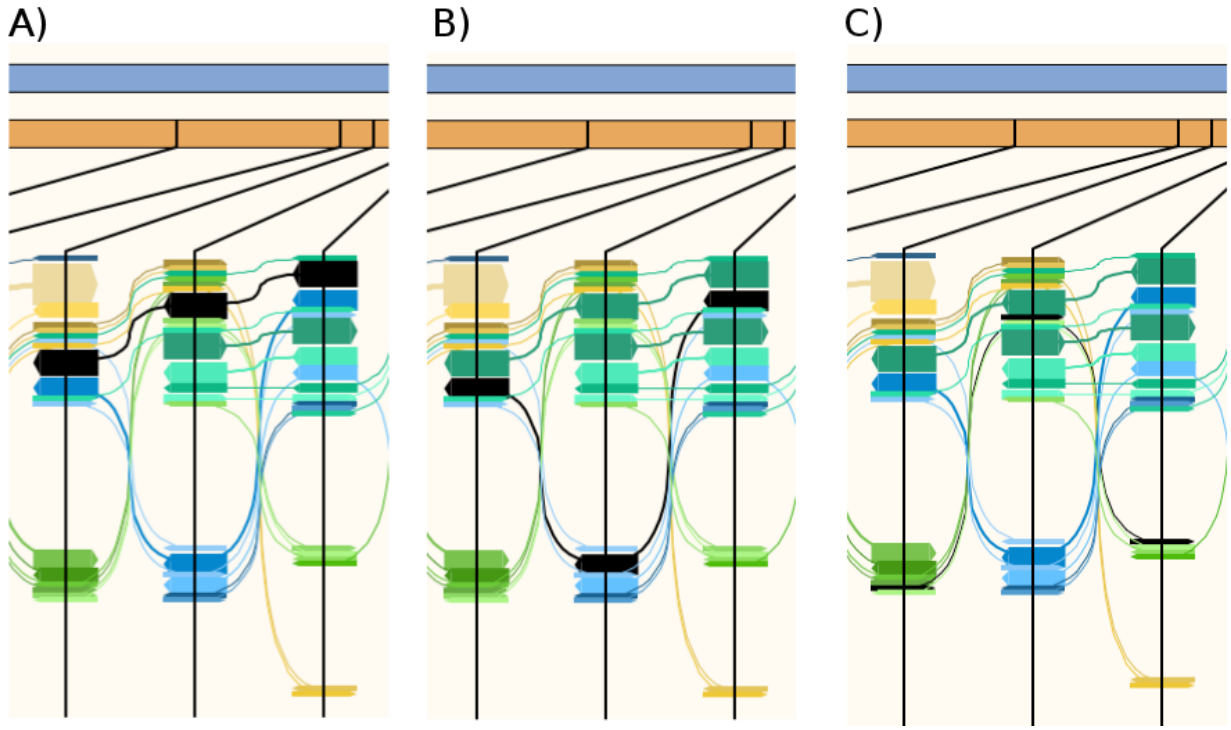



Figure 3: A section of Hapflow profile showing three SNPs (the black vertical lines in orange bar). Panel A-C shows three separate haplotypes (black arrows) representing the three *C. pecorum* strains. A) *C. pecorum* E58 flow. B) *C. pecorum* PV3056 flow. C) *C. pecorum* P787 flow.

# 6. Comand-line overview

Flow files can be created from the command-line using the following command.

$ python HapFlow.py –cl <sam_file> <vcf_file> <flow_file> <max_distance=1000>

Where

sam_File – sorted same file of aligned reads

vcf_file – vcf file of called variants

flow_file – file to write flows to

max_distance (optional) – maximum distance for to variants to be considered part of the same flow, set this to the maximum expected insert size.

# 7. Flow format specification

The flow format consists of one of each of these lines

**'C <chromosome_name>'**

chromosome_name - the name of the chromosome reads are aligned to

**'I <max_var> <depth_count_3_4> <depth_count>'**

max_var – maximum variations found at a single site

depth_count_3_4 – 75 percentile of read coverage

depth_count – maximum read coverage of a variant

**'G <max_var_1> <max_var_2> …. <max_var_err>'**

max_var_n – maximum read depth of nth most common variant

max_var_err – maximum read depth of errors

A line for each variant

**'V pos,var1,var2,….'**

pos – variant position

var1 – most common variant

var2 – second most common variant

etc.

an (*) is placed next to the variant present in the reference

Continued on next page….

Finally there is a line for each flow

**‘F pos,flow,group’**

pos – start position of the flow

flow – consists of a string of digit and symbols separated by commas

digit – 1 indicates the most common variant is found at this position, 2 indicates the second etc.

x – indicates that the bases at this position don’t match any called variant

_ – indicates that variant falls between the two reads of this pair

+ – indicates subsequent variants are on the forward strand

+s – indicates subsequent variants are on the forward strand and the read starts at the variant

- – indicates that subsequent variants are on the reverse strain

-s – indicates that subsequent variants are on the reverse strain and the read starts at the variant

e – indicates read ends on last variant