Haploflow: strain-resolved de novo assembly of viral genomes

About Haploflow:

- Haploflow is a strain-aware viral genome assembler for short read sequence data
- Uses a flow algorithm on a deBruijn graph data structure to resolve viral strains
- Written in C++ and works on UNIX systems
- **If using Haploflow, cite:**
  A., Bremges, A., Deng, ZL. et al. Haploflow: strain-resolved de novo assembly of viral genomes. Genome Biol 22, 212 (2021), https://doi.org/10.1186/s13059-021-02426-8

- Haploflow reconstructs full-length haplotypes (distinct viral genomes) from short-read sequencing data
- Designed for low-divergence viruses, meaning it's good at resolving strains that are very similar to one another (like different variants of the same virus)
- Uses a flow network approach on an assembly graph (de Bruijn) to extract high-quality haplotypes
- Can be used without a reference genome (de novo), although reference-based modes can also be applied according to one's pipeline

Installation: The easiest way to install haploflow is using bioconda and `conda install -c bioconda` in your UNIX environment.

- If this doesn't work, you can build haploflow from source:
  - First, clone this repository using `git clone <address>`, entering the directory from which you cloned Haploflow to
  - Create a build folder, `mkdir build`
  - Enter the new directory with `cd build; cmake ..`
    - This will create a Makefile which you can then run to create the Haploflow executable: `make`
    - This should create a `haploflow` executable in your build directory

Running Haploflow: The Haploflow executable can be directly executed

- Can show the help and parameters using `./haploflow --help` as follows:

```
HaploFlow parameters:                    Produce this help message
     -- help                             read file (fastq)
     - [ --read-file ] arg               deBruijn graph dump file
     - [ --dump-file ] arg               produced by HaploFlow
                                         log file (default: standard
     --log arg                           out)
     - [ --k ] arg (=41)                 k-mer size, default 41, please
                                         use an odd number
     - [ --out ] arg                     Folder for output, will be
                                         created if not present.
                                         WARNING: Old results will get
     - [ --error-rate ] arg              overwritten
(=0.0199999996)                          percentage filter for erroneous
                                         kmers - kmers appearing less
                                         than relatively e% will be
     --create-dump arg                   ignored
                                         Create dump of the deBruijn
                                         graph WARNING: This file may be
     --from-dump arg                     huge
                                         run from a Haploflow dump of
     - [ --two-strain ] arg              the deBruijn graph
(=0)                                     mode for known two-strain
                                         mixtures
     - [ --strict ] arg (=1)             more strict error correction,
                                         show be set to 5 in first run
                                         on new data set to reduce run
                                         time. Set to 0 if low abundant
                                         strains are expected to be
     - [ --filter ] arg (=500)           present
                                         filter contigs shorter than
     - [ --thresh ] arg (=-1)            value
                                         Provide a custom threshold for
     - [ --debug ] arg (=0)              complex/bad data
                                         Report all temporary graphs and
                                         coverage histogram
```

- The input reads are given with the `--read-file` option and the output directory with `--out`, which are the only required options
    - Haploflow will then run with default parameters
- The most important parameter is `k`, the *k*-mer size of the deBruijn graph
    - This is 41 by default
    - Increasing the value may improve assembly for large read lengths or very deep sequencing runs

- The `error-rate` parameter determines a lower bound of coverage or detection limit of different strains and is a percentage value
    - The default value is `0.02`, because Illumina data is expected to have less than 2% errors
    - Setting this value too low can cause Haploflow to run far slower
    - Setting this value too high will prevent Haploflow from finding lower abundant strains
- The `strict` parameter is complementary in the sense that it determines an overall lower bound for read coverage
    - `-1` imposes no constraints
    - `0` will use the inflection point of the coverage histogram
    - `≥ 1` will result in use of a sliding window over the coverage histogram to determine the lower bound
- The `thresh` parameter is mutually exclusive with the `strict` parameter and will overwrite its value if set
    - It sets a fixed threshold below which *k*-mers are ignored
    - Haploflow by default filters contigs shorter than 500 bp
        - This can be changed using the `filter` option
- The parameters `create-dump`, `from-dump`, and `dump-file` are just needed if the deBruijn graph is supposed to be written to a file to be reused in another run
    - This file is possibly huge (b/c it's uncompressed), so use with caution

Step 1: Install Haploflow

1) # Bioconda
```
conda create -n haploflow_env -c bioconda -c conda-forge
conda activate haploflow_env
```

Step 2: Prepare Files

2) # Make sure the trimmed reads are ready to go
   # either FASTQ or gzipped FASTQ (i.e. `sample_R1.fastq.gz`, `sample_R2.fastq.gz`)

Step 3: Run Haploflow

3) Run as `bash`
```
#!/bin/bash
#SBATCH --job_name=haploflow
#SBATCH --output=haploflow_%j.out
#SBATCH --error=haploflow_%j.err
#SBATCH --time=12:00:00
#SBATCH --cpus-per-task=8
#SBATCH --mem=100G

module load haploflow
# or activate conda env
# conda activate haploflow_env
```

4) Basic command

```
haploflow \
      --left sample_R1.fastq.gz \
      --right sample_R2.fastq.gz \
      --outdir haploflow_output \
      --threads 4
```

this will: assemble haplotypes de novo, output results to the defined directory, and use 4 CPU threads (this can be changed)