PIPELINE SCRIPT- DOCUMENTATION

RUNNING THE SCRIPT

The pipeline script is invoked as follows, unless the -e option is used:

```
python pipeline.py [options] <dataf> <cutf> <estsimparams>
```

If the -e option is used, the invocation looks like this:

```
python pipeline.py [options] <estf>
```

Each of the four input components will be covered below, with examples at the end of the section.

Options

The script has the following options:

-p, --peace <PARAMS>: Run peace on the simulated ESTs. Peace's parameters are specified as follows:

```
--analyzer d2
--clusterMaker mst
--estIdx 0
--frame 100 (if using D2)
--word 6 (if using D2)
```

-w, --wcd: Run wcd on the simulated ESTs. This must be immediately followed by one of two parameters:

- -c for cluster output (if you are comparing peace and wcd, you want this)
- -y for matrix output (only used for generating distance matrix)

-c, --processors: Specify the number of processors to use. Currently, this only applies to peace, because estsim and wcd do not run in parallel.

-e, --estfile: Rather than generating simulated ESTs, pass the script a predetermined EST file. When using this option, only one argument will be entered (the filename of the EST file) instead of the normal set of arguments.

dataf (.fa file)

The name of an input file in FastA format (typically with the extension .fa or .fasta) containing the sequences from which simulated ESTs are to be generated. Each sequence should be on a line by itself.

FastA format looks like this:

```
>TC301659 UP|Q5IBM5_BRARE (Q5IBM5) Esrom, complete
ATGAACAGGGGAAGGATGATGAAGCTAAGATGGCAGCGCTACTAGGGGTGCTGTCTTTACGGGGTCAAGAAAAA
CAGCAGGAATAAAC...
```

It is important that the file passed to the script be in the proper format, otherwise it will produce bizarre results. The ESTSim program does not check to make sure the file is a legitimate FastA file.

cutf (.dig file)

(the following is from the ESTsim documentation)

The name of an input file containing description of how the sequences should be cut up. Each line in the file gives one way. This can be:

random x y w

the base sequence will be split at random places to produce fragments of random length
between x and y. Note that the fragments produced are contiguous with no overlapping.
w is a floating point number between 0 and 1 (inclusive) that indicates which proportion of
fragments should be reverse-complemented.

samplerandom x y z w

fragments of random length between x and y are produced. On average, each base will be
appear in z fragments, though there is no guarantee that a base will appear at all. We would
expect that many of the fragments would overlap.
So, to produce overlapping fragments, one can either use multiple random lines or one (or
more) randomsample lines, or a combination.
w is the rc-proportion, discussed above.

pattern subsequence w

Essentially mimics a restriction site. Each basesequence will be cut at each point the restric-
tion subsequence occurs. w has the same meaning as discussed above.

In summary, each sequence in dataf is cut up as specified by each line in cutf.

estsimparams

The ESTsim parameters are given in the format: <numclones> <faultoptions>

(the following is from the ESTsim documentation)

numclones:
For each fragment we produce from dataf, we make a number of copies, mutating each copy
independently. This argument (an integer) says how many copies we want.

faultoptions:
To specify a fault option, give the name of the fault model followed by the parameters of that model. There can be as many fault models as desired (at the moment up to 3). The options are:

sbe α β γ ζ ξ κ λ μ ν
stutter η
ligate θ

Symbol	Definition	Range
α	Probability that an arbitrary base changes randomly	[0,1]
β	Margin at beginning of the EST where errors are most likely to happen	N+
γ	Probability that an error happens in the margin	[0,1]
ζ	Effect of rapid polymerase decay at end of read.	R+
ξ	Effect of gentle polymerase decay (gradually increases throughout read)	R+
κ	κ : (κ+λ+insprob+ν): proportion of errors that are substitutions.	N+
λ	Gives proportion of changes that are deletions	N+
μ	Gives proportion of changes that are insertions	N+
ν	Gives proportion of changes that are Ns	N+
η	Probability that stuttering happens after a repeat.	[0,1]
θ	Proportion of ligations that happen (as a proportion of the number of fragments).	[0,1]

(Note: stuttering and ligations are only used if the stutter and/or ligate fault models are specified. If sbe is the only fault model used, no stuttering or ligations will happen.)

EXAMPLES

```
python pipeline.py sequences.fa cutf.dig 1 sbe 0.020000 10 0.030000 1 0 10 1 1 2
```

-Generate simulated ESTs from the sequences in file sequences.fa, using the file cutf.dig, do not clone any of the ESTs, and use the sbe fault model with the specified fault options (0.020000 10 0.030000 1 0 10 1 1 2).

```
python pipeline.py -p --analyzer clu --clusterMaker mst --estIdx 0 sequences.fa cutf.dig 1 sbe 0.020000 10 0.030000 1 0 10 1 1 2
```

-Same as above, but run Peace on the resulting simulated ESTs, and analyze the results.

```
python pipeline.py -w -p --analyzer clu --clusterMaker mst --estIdx 0 sequences.fa cutf.dig 1 sbe 0.020000 10 0.030000 1 0 10 1 1 2
```

-Same as above, but also run WCD on the same set of simulated ESTs, and analyze the results.

```
python pipeline.py -c 4 -w -p --analyzer clu --clusterMaker mst --estIdx 0 sequences.fa cutf.dig 1 sbe 0.020000 10 0.030000 1 0 10 1 1 2
```

-Same as above, but also request 4 processors for running the Peace batch job.

```
python pipeline.py -w -p --analyzer clu --clusterMaker mst --estIdx 0 -e estfile.fa
```

-Run wcd and peace on the ESTs contained in the estfile.fa file (this option will not generate simulated ESTs).

SCRIPT OUTPUT

Console

The script will print the results of its analysis to the console (as well as store it in the analysis.txt file).

Directory

The script will move all intermediate and output files into a new directory. This directory is named for the EST output .fa file (minus the extension), with peace_ added at the beginning if peace was run, and wcd_ added at the beginning if wcd was run(wcd_peace_ if both were run). If the directory already exists, the script appends the PBS job ID to the end of the directory name to make it unique.

Files

The script stores all intermediate and output files related to a run (except for PBS job files, which it cleans up). This section is intended as a reference to explain the output files.

.fa file- The file that results from running ESTsim, and then running the format.py script. This will be named based on the ESTsim parameters and input files, in the following format:
estsim_faFile_digFile[estsim parameters, separated by underscores]_fmt.fa

peace output .mst file- The file that results from running peace on the ESTsim output. It is named by appending 'peace_' to the beginning of the ESTsim output file name, and instead of ending with '_fmt.fa', it ends with the '.mst' extension. If the peace command line option was not specified this file will not be created.

wcd output .txt file- The file that results from running wcd on the ESTsim output (using WCD's "compressed cluster format" option). It is named by appending 'wcd_' to the beginning of the ESTsim output file name, and instead of ending with '_fmt.fa', it ends with '.txt'. If the wcd command line option was not specified this file will not be created.

analysis.txt- A text file containing the results of the peace and WCD analysis (true positives, false positives, true negatives, false negatives). Any additional analysis we add to the script in the future (e.g. the results of some score function) would be added here.

peace_script.oxxxxxx- The output to the standard out stream of the peace job.

estsim_script.oxxxxxx- The output to the standard out stream of the estsim job.

wcd_script.oxxxxxx- The output to the standard out stream of the wcd job.

If only one of wcd or peace is run, there will only be one of these output files. If both wcd and peace are run, all three files will be created.