Neptune

Version 1.2.4

Eric Marinier

February 27, 2017

Contents

1 Introduction

Neptune locates genomic signatures using an exact k-mer matching strategy while accommodating k-mer mismatches. The software identifies sequences that are sufficiently represented within inclusion targets and sufficiently absent from exclusion targets. The signature discovery process is guided by probabilistic models instead of heuristic strategies. Neptune identifies general-purpose signatures are agnostic of application-specific requirements, such as physical and chemical properties, and may require further investigation to determine their appropriateness for application.

2 Installation

This installation guide assumes the use of the BASH Unix shell. Neptune may be installed on most 64-bit Unix environments. However, the specifics of installations on all environments is beyond the scope of this manual. Neptune may either be run on a single machine or a computing cluster. Neptune achieves maximum parallelization when submitting jobs through a DRMAA-compliant cluster computing scheduler. The installation and configuration of a DRMAA-compliant scheduler will require a significant understanding of Unix. However, it is possible to run Neptune in parallel on a single machine without DRMAA. Neptune is known to be compatible with the SGE and Slurm schedulers.

2.1 Python

Neptune requires Python 2.7. Note that Python 2.7 is provided with many major distributions of Linux. The following may check your Python version:

```
$ python --version
```

2.2 Dependencies

2.2.1 Debian-Based Installation

This section assumes the user has the APT package manager. This is common to the Ubuntu operating system. However, this section should be compatible with any 64-bit Debian distribution. The following operation will automatically install Neptune's dependencies and require security privileges (sudo) to install the dependencies:

```
$ sudo neptune/install/debian_dependencies.sh
```

2.2.2 Manual Installation

If you cannot install the dependencies using the above script, the following dependencies must be manually installed, if necessary, by the user:

- pip
- virtualenv
- build-essential
- python-dev
- NCBI BLAST+

2.3 Neptune

Neptune will be installed using pip into its own Python virtual environment. The following will install Neptune locally into the source directory and will not require security privileges:

```
$ neptune/INSTALL.sh
```

Alternatively, you may specify an install location, PREFIX, such as /usr/local/. Neptune will create the directories PREFIX/lib and PREFIX/bin. This may require security privileges:

```
$ neptune/INSTALL.sh PREFIX
```

2.4 DRMAA Requirements

The following is only necessary for execution of Neptune in DRMAA mode on a cluster computing environment. These instructions require a strong understanding of Unix and cluster computing configuration. The user will need to manually install and configure a DRMAA-compliant scheduler, such as SGE or Slurm, on either a single machine or on a computing cluster. The user will additionally need to install and configure Python DRMAA bindings with considerations for the DRMAA-compliant scheduler. The following are required to operate Neptune in DRMAA mode:

1. DRMAA-compliant scheduler
2. Python DRMAA bindings

2.4.1 DRMAA-Compliant Scheduler

Neptune has been tested using SGE installed on a single machine with the following instructions:

https://scidom.wordpress.com/2012/01/18/sge-on-single-pc/

However, any DRMAA-compliant scheduler is expected to work. The instructions for installing and configuring such scheduling environments are beyond the scope of this resource.

2.4.2 Python DRMAA Bindings

Neptune uses a Python DRMAA binding to schedule DRMAA jobs and communicate with the scheduler. The information necessary for installing and configuring the Python DRMAA bindings is available the following location:

https://github.com/pygridtools/drmaa-python

2.5 DRMAA Installation

It may be helpful to create a submission wrapper script for Neptune to avoid entering the same DRMAA native specification parameters for every submission. The following SGE and Slurm submission wrapper scripts automatically include native specification parameters, appropriate for the scheduling environment, which may be overwritten by the submitting user as necessary.

2.5.1 Slurm Wrapper

neptune-slurm

```
#!/usr/bin/env bash

DRMAA_LIBRARY_PATH=/usr/local/lib/libdrmaa.so.1

neptune --drmaa --default-specification "-n 1 --nodes=1 --ntasks-per-node=1 --mem
    ↪ =10240" $@
```

Slurm Example

```
$ neptune-slurm -i /path/to/inclusion/ -e /path/to/exclusion/ -o /path/to/output/
```

2.5.2 SGE Wrapper

neptune-sge

```
#!/usr/bin/env bash

DRMAA_LIBRARY_PATH=/opt/gridengine/lib/linux-x64/libdrmaa.so

neptune --drmaa --default-specification "-l h_vmem=8G -pe smp 4" $@
```

SGE Example

```
$ neptune-sge -i /path/to/inclusion/ -e /path/to/exclusion/ -o /path/to/output/
```

3 Parameters

A help message may be viewed by running:

```
$ neptune --help
```

3.1 Required Parameters

Neptune requires the location of the inclusion, exclusion, and output directories. The remaining parameters will be estimated based on the input sequence or revert to default settings. The following is the minimum number of command line parameters required to run Neptune:

```
$ neptune
   --inclusion /path/to/inclusion/
   --exclusion /path/to/exclusion/
   --output /path/to/output/
```

The following parameters are required by Neptune:

inclusion
: **-i [LOCATION ...] // --inclusion [LOCATION ...]**
A list of inclusion targets in FASTA format. You may list multiple file or directory locations following the **--inclusion** parameter. Neptune will automatically include all root-level files within directories.

exclusion
: **-e [LOCATION ...] // --exclusion [LOCATION ...]**
A list of exclusion targets in FASTA format. You may list multiple file or directory locations following the **--exclusion** parameter. Neptune will automatically include all root-level files within directories.

output
: **-o [LOCATION] // --output [LOCATION]**
The location of the output directory. If this directory exists, any files produced with existing names will be overwritten. If this directory does not exist, then it will be created.

3.2 *k*-mer Parameters

The following parameters relate to k-mer generation and aggregation:

***k*-mer**
: **-k [INT] // --kmer [INT]**
The size of the k-mers. This must be a positive integer and should be large enough such that random *intra*-genome k-mer matches, within the largest genome, are unexpected. The size of k-mers cannot be larger than the smallest sequence record. This will be automatically calculated if not specified.

organization
: **--organization [INT]**
The degree of organization of k-mer counting and aggregation. This parameter determines the number nucleotide bases used in parallelized k-mer counting and, in turn, the number of parallel instances of k-mer aggregation. The number of parallel instances is determined by 4^o, where o is the specified organization argument (**--organization**). This value must be a non-negative integer smaller than k. If the parameter is not specified, then $o = 0$ and there will be no parallel k-mer aggregation. This will likely require a much longer computation time to complete k-mer aggregation.

3.3 Filtering Parameters

The following command-line parameters relate to signature filtering:

filter length

--filter-length [FLOAT]

The minimum percent length of a signature candidate against a exclusion target required to filter out the candidate. This value is a percentage expressed as a floating point number [0.0, 1.0]. If the any exclusion hit exceeds the percent length **and** percent identity of any candidate, the candidate is removed. The default value is 0.5.

filter percent

--filter-percent [FLOAT]

The minimum percent identity of a signature candidate against a exclusion target required to filter out the candidate. The percent identity is calculated as identities divided by the alignment length. This value is a percentage expressed as a floating point number [0.0, 1.0]. If the any exclusion hit exceeds the percent length **and** percent identity of any candidate, the candidate is removed. The default value is 0.5.

seed size

--seed-size [INT]

The seed size used for alignments. This value must be no smaller than 4. The default value is 11.

3.4 Extraction Parameters

The following command-line parameters relate to signature extraction:

reference

-r [FILE] [FILE ...] // --reference [FILE] [FILE ...]

A list of references from which to extract signatures. If this parameter is not specified, signatures will be extracted from **all** inclusion targets. You may list multiple file locations following the **--reference** parameter.

rate

--rate [FLOAT]

This is the probability (0.0, 1.0) that any two homologous bases are different from each other. This should incorporate mutation rates, sequencing error rates, and assembly error rates. The rate is used to calculate the maximum allowable gap size in a signature and the minimum expected number of exact k-mer matches in a signature. If this value is not specified, the rate is assumed to be 0.01.

gc-content

--gc-content [FLOAT]

The expected GC-content of the environment. The GC-content is used to calculate the maximum allowable gap size in a signature and the minimum expected number of exact k-mer matches in a signature. If this value is not specified, it is calculated by observing the GC-content of each target during signature extraction. The value must be between (0.0, 1.0).

confidence

--confidence [FLOAT]

The statistical confidence of decision making in the software. The confidence affects the automatic calculation of both the maximum gap size and minimum number of inclusion hits. If this value is not specified, a default of 0.95 is used. The value must be between (0.0, 1.0).

minimum inclusion hits

--inhits [INT]

The minimum number of inclusion hits required to start and continue signature extraction. If this value is not specified, it will be automatically calculated using the number of inclusion targets, the GC-content, the rate, and the k-mer size. The calculation can be found in the *Mathematics* documentation. This value must be a positive integer.

minimum exclusion hits

--exhits [INT]

The minimum number of exclusion hits necessary to stop extraction of a signature. If this value is not specified, it is assumed to be 1. This value must be a positive integer.

maximum gap size

--gap [INT]

The maximum allowable number of base positions shifted before seeing an exact k-mer match. If this value is not specified, it will be automatically calculated using the rate, GC-content, and the k-mer size. The calculation can be found in the *Mathematics* documentation. This value must be a positive integer.

minimum signature size

--size [INT]

The minimum size for a signature. Signatures which are shorter than this length will not be reported. If this value is not specified, the minimum signature size will be four times the length of the k-mer size ($4k$). It is not recommended to locate signatures smaller than this size, unless application-specific. This value must be a positive integer.

3.5 Parallelization Parameters

The following parameters relate to the parallelization of Neptune:

parallelization

-p [INT] // --parallelization [INT]

The number of parallel working processes to create when Neptune is operating in a non-DRMAA mode (default). This parameter will directly increase the speed of many stages of the software, provided there are sufficient resources available to run the worker process simultaneously. This value must be a positive integer. The default value is 8.

3.6 DRMAA Parameters

It may be necessary to specify job submission parameters that are required by your cluster-computing environment. If you require DRM-specific command line arguments, they may be provided to Neptune using one of several arguments. The **--default-specification** parameter will provide the DRM-specific arguments to all jobs which are created. Additional command line arguments allow precise specifications for each type of job.

drmaa

--drmaa

This flag enables DRMAA-based Neptune execution. This will require a DRMAA-compatible cluster computing environment to be installed and configured. However, Neptune will likely operate significantly faster in this environment.

default specification

--default-specification [STRING]

DRMAA-specific command line arguments for all jobs. These arguments must be provided as a quoted string. The default specification will be applied to all job types and overwritten when specified.

count specification

--count-specification [STRING]

DRMAA-specific command line arguments for k-mer counting. These arguments must be provided as a quoted string. These arguments will overwrite the default specification, if specified, for this job type.

aggregate specification

--aggregate-specification [STRING]

DRMAA-specific command line arguments for k-mer aggregation. These arguments must be provided as a quoted string. These arguments will overwrite the default specification, if specified, for this job type.

extract specification

--extract-specification [STRING]

DRMAA-specific command line arguments for signature extraction. These arguments must be provided as a quoted string. These arguments will overwrite the default specification, if specified, for this job type.

database specification

--database-specification [STRING]

DRMAA-specific command line arguments for database construction. These arguments must be provided as a quoted string. These arguments will overwrite the default specification, if specified, for this job type.

filter specification

--filter-specification [STRING]

DRMAA-specific command line arguments for candidate signature filtering. These arguments must be provided as a quoted string. These arguments will overwrite the default specification, if specified, for this job type.

consolidate specification

--consolidate-specification [STRING]

DRMAA-specific command line arguments for signature consolidation. These arguments must be provided as a quoted string. These arguments will overwrite the default specification, if specified, for this job type.

4 Examples

4.1 Basic

The following basic example will report all of the signatures that are sufficiently shared by the (FASTA) sequences in the inclusion directory and sufficiently absent from the (FASTA) sequences in the exclusion directory. Neptune will automatically calculate many of the parameters used in this execution.

```
$ neptune
   --inclusion inclusion_directory/
   --exclusion exclusion_directory/
   --output output_directory/
```

The output of immediate interest will be located in the follow file:

```
output_directory/consolidated/consolidated.fasta
```

This file will contain a consolidated list of signatures, sorted by their Neptune score, which is a combined estimate of sensitivity and specificity. The signatures with higher scores, near the top of the file, are considered the most discriminatory signatures.

4.2 File Locations

You may wish to specify particular files used in signature discovery. This may be important when specifying references for signature extraction:

```
$ neptune
   --inclusion inclusion_dir/ in1.fasta in2.fasta
   --exclusion exclusion_dir/ ex1.fasta ex2.fasta
   --reference in1.fasta in2.fasta
   --output output/
```

4.3 DRMAA Parameters

It may be necessary to specify DRMAA native specification parameters to accommodate Neptune job scheduling. This example specifies the resources required by all jobs (**--default-specification**) and further specifies that k-mer aggregation jobs (**--aggregate-specification**) will require more memory. The remaining Neptune parameters are automatically calculated.

```
$ neptune
   --inclusion inclusion/
   --exclusion exclusion/
   --output output/
   --default-specification "-l h_vmem=6G -pe smp 4"
   --aggregate-specification "-l h_vmem=10G -pe smp 4"
```

5 Output

Neptune's output directory contains the following items:

- **candidates**: directory containing signature candidates
- **filtered**: directory containing filtered candidates in extracted order
- **sorted**: directory containing filtered signatures in sorted order
- **consolidated**: directory containing the consolidate signatures
- **database**: directory containing Neptune's constructed databases
- **aggregate.kmers**: file containing all observed k-mers
- **receipt.txt**: file containing Neptune's run receipt

A file with the same name as each reference will be placed in each output directory, corresponding to the reference file from which it was derived.

5.1 Candidate Signatures

The candidate signatures are the sequences produced from the signature extraction step. These signatures will relatively sensitive, but not necessarily specific. This is because signature extraction is done using exact k-mer matches. The candidate signatures are guaranteed to contain no more exact matches with any exclusion k-mer than specified by the **--exhits** parameter. However, there may be inexact matches with exclusion targets.

5.2 Filtered Signatures

The filtering step is designed to remove signatures which are not interesting enough to warrant further investigation, because the negative component of their score is prohibitively large. The filtering step removes signatures that align sufficiently with any exclusion target. The filtered signatures are a subset of the candidate signatures.

5.3 Sorted Signatures

The sorted signatures files are organized as FASTA records containing the same signatures as their filtered signatures counterparts. However, the signatures are listed in descending order by their signature score. Signatures are assigned a score corresponding to their highest-scoring BLAST alignments with all inclusion and exclusion targets, which is a sum of a positive inclusion component and a negative exclusion component. This score is maximized when all inclusion targets contain a region exactly matching the entire signature and there exists no exclusion targets that match the signature. The signatures have the following format:

```
>[ID] [SCORE] [IN SCORE] [EX SCORE] [LENGTH] [REF] [POS]
[SEQUENCE]
```

```
>425 score=0.86 in=0.98 ex=-0.13 len=31 ref=ecoli pos=160
TGTCATTCTCCTGTTCTGCCTGTATCACTGC
```

Where:

- [**ID**]: an arbitrary, run-unique ID assigned to the signature
- [**SCORE**]: the total signature score

- [**IN SCORE**]: positive inclusion component of signature score
- [**EX SCORE**]: negative exclusion component of signature score
- [**LENGTH**]: signature length in bases
- [**REF**]: name of the contig from which the signature was extracted
- [**POS**]: starting position of the signature in the reference
- [**SEQUENCE**]: sequence content of the signature

5.4 Consolidated Signatures

The sorted signatures from all references are combined into a single "consolidated.fasta" file, located within the "consolidated" directory. Signatures are added to the consolidated signatures file in a greedy manner by selecting the next highest scoring signature available from all references. While effort is taken to prevent signatures from overlapping entirely, it is possible for consolidate signatures to have a small amount of overlap.

5.5 Databases

The databases directory contains BLAST databases constructed from the inclusion and exclusion files.

5.6 Aggregate k-mers

The aggregated k-mers file, aggregated.kmers, contains a list of all k-mers observed in the inclusion and exclusion groups. These k-mers are sorted and followed by two integers: the number of inclusion and exclusion targets the k-mer appears in, respectively.

5.7 Run Receipt

The run receipt contains information about the Neptune execution. It contains a list of all the files in the inclusion and exclusion group, and the command line parameters used for the execution.

6 Walkthrough

6.1 Overview

The purpose of this walkthrough will be to illustrate a simple, but complete example of using Neptune to locate discriminatory sequences. We will identity signature sequences within an artificial data set containing three inclusion sequences and three exclusion sequences. The output will be a list of signatures, sorted by score, for each inclusion target, and one consolidated signatures file, sorted by signature score, containing signatures from all inclusion targets.

6.2 Input Data

We will be using very small, artificial genomes for this walkthrough. However, these small genomes will be sufficient to illustrate the operation of Neptune. The artificial genome sequence content is derived from *Escherichia coli* and has been modified to introduce simple variation between genomes.

The example inclusion genomes are located in the following location:

```
neptune/tests/data/example/inclusion/
```

The example exclusion genomes are located in the following location:

```
neptune/tests/data/example/exclusion/
```

The inclusion and exclusion directories each contain three FASTA format genomes. The genomes all have some insertions and deletions that differentiate them from each other. However, the three inclusion genomes primarily differ from the three exclusion genomes in that they share large sequences that are absent from all exclusion genomes.

6.3 Running Neptune

Neptune will automatically calculate many of the parameters that might otherwise be specified by the user, such as the minimum number of targets signature sequence must be present within for it to be considered shared sequence. At minimum, Neptune requires the user specify the inclusion sequences, exclusion sequences, and an output directory. We will provide Neptune inclusion and exclusion sequences in the form of FASTA file genomes located within directories. The following command will run Neptune on the example data and output to the specified directory:

```
$ neptune
   --inclusion tests/data/example/inclusion/
   --exclusion tests/data/example/exclusion/
   --output output/
```

6.4 Output

6.4.1 Standard Output

After running Neptune, very similar output will be printed to standard output, indicating that Neptune is starting and completing different stages of operation:

```
Neptune v1.2.4

Estimating k-mer size ...
k = 15

k-mer Counting...
Submitted 12 jobs.
0.002063 seconds

k-mer Aggregation...
Submitted 65 jobs.
0.010473 seconds

Signature Extraction...
Submitted 6 jobs.
0.000771 seconds

Signature Filtering...
Submitted 2 jobs.
Submitted 6 jobs.
0.002498 seconds

Consolidate Signatures...
Submitted 1 jobs.
0.000411 seconds

Complete!
```

6.4.2 Consolidated Signatures

As we did not specify references from which to extract signatures, Neptune will automatically investigate all inclusion genomes for signatures and consolidate those signatures into a single consolidated signature file. The *output/consolidated/consolidated.fasta* file contains these consolidated signatures. This file may be understood as the final output of the application. The following FASTA output is from the consolidated signatures file produced from this example:

output/consolidated/consolidated.fasta

```
>1.0 score=1.0000 in=1.0000 ex=0.0000 len=103 ref=inclusion1 pos=99
TAGTCTCCAGGATTCCCGGGGCGGTTCAGATAATCTTAGCATTGACCGCCTTTATATAGAAGCTGTTATTCAAGAAGCAT...
>1.1 score=0.9979 in=0.9979 ex=0.0000 len=640 ref=inclusion1 pos=3497
CGCGGGCGATATTTTCACAGCCATTTCAGGAGTTCAGCCATGAACGCTTATTACATTCAGGATCGTCTTGAGGCTCAGAG...
>1.2 score=0.9966 in=0.9966 ex=0.0000 len=98 ref=inclusion1 pos=5209
GCGAGTTTTGCGAGATGGTGCCGGAGTTCATCGAAAAAATGGACGAGGCACTGCTGAAATTGGTTTTGTATTTGGGGAGC...
```

The FASTA header contains information relavent to the identified signature. A detailed explanation of this information is located within the output section of this manual. The *score* is the sum of the *in* (inclusion/sensitivity) and *ex* (exclusion/specificity) scores, and represents a combined measure of sensitivity and specificity. The *length* describes the length of the signature in bases. The *ref* (reference) and *pos* (position) describe the location of the signature within the reference FASTA record it was extracted from.

In this example, Neptune identified three signatures: 1.0, 1.1, and 1.2 of lengths 103, 640, and 98, respectively. We see that all of these signatures originated from the *inclusion1* reference. These signatures were located at positions 99, 3497, and 5209 within the *inclusion1* reference. These signatures are of very high quality, within the context of our data set, with scores of 1.0000, 0.9979, and 0.9969, within the possible range of score values from -1.00 to +1.00.

6.4.3 Sorted Signatures

If we're interested in looking at the signatures produced from each individual inclusion target, we need to investigate the output in the *output/sorted* directory. The following are the signatures extracted exclusively from the *inclusion1.fasta* target:

output/sorted/inclusion1.fasta

```
>0 score=1.0000 in=1.0000 ex=0.0000 len=103 ref=inclusion1 pos=99
TAGTCTCCAGGATTCCCGGGGCGGTTCAGATAATCTTAGCATTGACCGCCTTTATATAGAAGCTGTTATTCAAGAAGCAT...
>1 score=0.9979 in=0.9979 ex=0.0000 len=640 ref=inclusion1 pos=3497
CGCGGGCGATATTTTCACAGCCATTTCAGGAGTTCAGCCATGAACGCTTATTACATTCAGGATCGTCTTGAGGCTCAGAG...
>2 score=0.9966 in=0.9966 ex=0.0000 len=98 ref=inclusion1 pos=5209
GCGAGTTTTGCGAGATGGTGCCGGAGTTCATCGAAAAAATGGACGAGGCACTGCTGAAATTGGTTTTGTATTTGGGGAGC...
```

The following are the signatures extracted exclusively from the *inclusion2.fasta* target:

output/sorted/inclusion2.fasta

```
>0 score=1.0000 in=1.0000 ex=0.0000 len=103 ref=inclusion2 pos=99
TAGTCTCCAGGATTCCCGGGGCGGTTCAGATAATCTTAGCATTGACCGCCTTTATATAGAAGCTGTTATTCAAGAAGCAT...
>1 score=0.9979 in=0.9979 ex=0.0000 len=640 ref=inclusion2 pos=3494
CGCGGGCGATATTTTCACAGCCATTTTCAGGAGTTCAGCCATGAACGCTTATTACATTCAGGATCGTCTTGAGGCTCAGA...
>2 score=0.9933 in=0.9933 ex=0.0000 len=99 ref=inclusion2 pos=5206
GCGAGTTTTGACGAGATGGTGCCGGAGTTCATCGAAAAAATGGACGAGGCACTGCTGAAATTGGTTTTGTATTTGGGGAG...
```

The following are the signatures extracted exclusively from the *inclusion3.fasta* target:

output/sorted/inclusion3.fasta

```
>0 score=1.0000 in=1.0000 ex=0.0000 len=103 ref=inclusion3 pos=99
TAGTCTCCAGGATTCCCGGGGCGGTTCAGATAATCTTAGCATTGACCGCCTTTATATAGAAGCTGTTATTCAAGAAGCAT...
>2 score=0.9833 in=0.9833 ex=0.0000 len=100 ref=inclusion3 pos=5203
GCGAGTTTTAACGAGATGGTGCCGGAGTTCATCGAAAAAATGGACCGAGGCACTGCTGAAATTGGTTTTGTATTTGGGGA...
>1 score=0.9792 in=0.9979 ex=0.0187 len=640 ref=inclusion3 pos=3492
CGCGGGCGATATTTTCACAGCCATTTTCAGGAGTTCAGCCATGAACGCTTATTACATTCAGGATCGTCTTGAGGCTCAGA...
```

The output from these files appears very similar, as is expected when Neptune identifies highly discriminatory signatures from a homogeneous data set. However, there are some slight differences between some of these signatures. For example, the signatures in each of these output files have corresponding ID numbers and some of these signatures have slight differences. However, because Neptune assigns signature IDs arbitrarily, this correspondence will usually never happen when using real data. Nonetheless, we see that signature ID 2 is slightly different sizes in all three inclusion targets (5209, 5206, and 5203) with slightly different scores (0.9966, 0.9933, 0.9833). Another slight difference between the signatures is the sequence similarity of signature ID 1 in *inclusion3.fasta* with exclusion sequence:

```
>1 score=0.9792 in=0.9979 ex=0.0187 len=640 ref=inclusion3 pos=3492
CGCGGGCGATATTTTCACAGCCATTTTCAGGAGTTCAGCCATGAACGCTTATTACATTCAGGATCGTCTTGAGGCTCAGA...
```

This signature had some similarity with exclusion sequence, represented by the *ex=0.0187*, and indicates a small amount of imprecision in this signature. This example also illustrates that the *score* (0.9792) is the sum of the *in* (0.9979) and *ex* (0.0187) values.

These differences in signatures from each inclusion target are a consequence of sequence differences. The user's discretion will be required in determining which of these are most appropriate. Nonetheless, as described above, Neptune will attempt to consolidate these signatures into a single output file, if a single answer is desirable.