Neptune
Version 1.2.0

Eric Marinier

March 16, 2016

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 user is using the APT package manager and the BASH Unix shell. These are common to the Ubuntu Linux operating system. Neptune may be installed on most 64-bit Unix environments. However, the specifics of installations on other environments is beyond the scope of this manual.

Neptune may either be run on a single machine or on 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 substantial understanding of Unix. However, it is possible to run Neptune on a single machine without DRMAA. Neptune is known to be compatible with the SGE and Slurm schedulers.

2.1 Neptune Requirements

Neptune should be installed and run on a standard 64-bit Unix environment. The installation process will attempt to automatically install many dependencies. However, the following dependencies must be manually installed, if necessary, by the user:

1. Python 2.7

2. NumPy

3. SciPy

4. BLAST+

5. pipsi

2.1.1 Python 2.7

Please note that Python 2.7 is provided with many major distributions of Linux. However, if this is not the case with your distribution, then the following should install Python 2.7:

```
# Install requirements:
$ sudo apt-get install build-essential checkinstall
$ sudo apt-get install libreadline-gplv2-dev libncursesw5-dev libssl-dev libsqlite3
    ↪ -dev tk-dev libgdbm-dev libc6-dev libbz2-dev

# Download Python 2.7.10:
$ cd /usr/src
$ wget https://www.python.org/ftp/python/2.7.10/Python-2.7.10.tgz

# Extract:
$ tar xzf Python-2.7.10.tgz

# Compile Python source:
$ cd Python-2.7.10
$ sudo ./configure
$ sudo make altinstall

# Check the Python version:
$ python2.7 -V
```

2.1.2 NumPy and SciPy

Please note that NumPy and SciPy are provided with many major distributions of Linux. However, if this is not the case your distribution, then either refer to the official NumPy installation instructions and official SciPy installation instructions, or the following should install NumPy and SciPy:

```
# Install NumPy requirements:
$ sudo apt-get install gcc gfortran python-dev libblas-dev liblapack-dev cython

# Install SciPy requirements:
$ sudo apt-get install python-matplotlib ipython ipython-notebook python-pandas
    ↪ python-sympy python-nose

# Install NumPy and SciPy:
$ sudo apt-get install python-numpy python-scipy
```

2.1.3 BLAST

BLAST may be installed using APT:

```
# Install BLAST+:
$ sudo apt-get install ncbi-blast+
```

2.1.4 pipsi

The pipsi tool is used to install Python packages using pip into shielded virtual environments. The software attempts to avoid the problem of software installations conflicting with other installations using differing dependencies. The pipsi tool may be installed using curl and Python:

```
# Install pipsi:
$ curl https://raw.githubusercontent.com/mitsuhiko/pipsi/master/get-pipsi.py |
    ↪ python
```

Furthermore, you may need to add the default pipsi install location to your PATH variable.

```
# The following adds a line to your bashrc file:
$ echo "export PATH=$PATH:~/.local/bin" | cat >> ~/.bashrc

# Source the updated bashrc file:
$ source ~/.bashrc

# Ensure the PATH was updated correctly:
$ echo $PATH | grep local/bin
```

2.2 Neptune Installation

Neptune executes in a Python environment and therefore does not need to be explicitly compiled. Neptune is best installed using pipsi, as it will further install several Python dependencies required by Neptune automatically. The installation is performed by running the following:

```
# Install Neptune:
$ pipsi install /path/to/neptune/download/location/

# Ensure installation was successful:
$ neptune --version
> neptune 1.2.0
```

2.3 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

2.3.1 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.

2.3.2 Python DRMAA Bindings

Neptune uses a Python DRMAA binding to schedule parallel 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

2.4 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.

2.4.1 Slurm Wrapper

<div align="center">neptune-slurm</div>

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

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

/home/user/.local/bin/neptune --drmaa --default-specification "-n 1 --nodes=1 --
    ↪ ntasks-per-node=1 --mem=10240" $@
```

<div align="center">Slurm Example</div>

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

2.4.2 SGE Wrapper

neptune-sge

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

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

/home/user/.local/bin/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] [LOCATION ...] // --inclusion [LOCATION] [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] [LOCATION ...] // --exclusion [LOCATION] [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.

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°, 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 example will allow Neptune to automatically calculate many of the parameters used in execution. However, Neptune will make assumptions about the probability two homologous bases are different and use fixed thresholds for signature filtering.

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

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**]: a 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.