SIDR Documentation

Release 0.0.1

Duncan Murdock

Aug 10, 2018

Contents

SIDR (Sequence Identification with Decision tRees, pronounced *cider*) is a tool to filter Next Generation Sequencing (NGS) data based on a chosen target organism. SIDR uses data fron BLAST (or similar classifiers) to train a Decision Tree model to classify sequence data as either belonging to the target organism, or belonging to something else. This classification can be used to filter the data for later assembly.

There are two ways to run SIDR. The first, or default, method takes a number of bioinformatics files as input and calculates relevant statistics from them. The second, or custom, method takes a specifically formatted tab-delimited file with your chosen statistics and uses that directly to train the model.

CHAPTER 1

Documentation

1.1 Installing SIDR

SIDR can be installed either using pip–the python package manager–or manually using the included setup.py file.

1.1.1 Dependencies

SIDR is able to install all of its dependencies from PyPI automatically. This should work in most cases. If you are installing under Python 3, you may need to manually install Cython with:

```
pip install cython
```

If you have a locally installed version of HTSLib, you can include it by using the commands:

```
export HTSLIB_LIBRARY_DIR=/usr/local/lib
export HTSLIB_INCLUDE_DIR=/usr/local/include
```

before installing SIDR.

1.1.2 OSX

Users running a recent version of OSX may need to install an alternative python distribution like that provided by Homebrew. If SIDR installation fails with permission errors, this is the most likely solution.

1.1.3 Using a Virtualenv

Some cluster users may need to setup a Python virtualenv due to the nature of working with a cluster environment. A virtualenv can be setup with the commands:

```
virtualenv venv
. venv/bin/activate
```

If necessary, virtualenv can be installed in the user's home directory (~/.local/bin must be in $PATH) with the following command:

```
pip install --user virtualenv
```

1.1.4 Installing from PyPI

Installing from PyPI is the easiest method, and thus the recommended one. To install SIDR:

```
pip install sidr
```

1.1.5 Installing from Source with pip

Note: When installing from source, setuptools will attempt to contact PyPI to install dependencies. If this is not an option then dependencies will need to be manually installed.

If PyPI is not an option or if you'd like to run the latest development version, SIDR can be installed by running the following command:

```
pip install git+https://github.com/damurdock/SIDR.git
```

If you're installing SIDR in order to develop it, download the source from GitHub and install it by running the following command in the unzipped source directory:

```
pip install --editable .
```

1.1.6 Installing from Source with Setup.py

If for some reason pip is completely unavailable, SIDR can be installed by downloading the source from GitHub and running the following command command in the unzipped source directory:

```
python setup.py install
```

1.2 Data Preparation

In order to run SIDR, you will need to perform several analyses of your data. For the default analysis, you will need:

- A preliminary assembly
- An alignment back to that preliminary assembly
- A BLAST classification of that assembly
- A copy of the NCBI Taxonomy Dump

Alternatively, you can precalculate the data you wish to use to train the model, and save it in a specific format for input. This is explained here: *Running With a Runfile*.

1.2.1 Assembly

SIDR requires a preliminary assembly of your data built with standard *de novo* assembly techniques. The scaffolds from this assembly will be used as input for the machine learning model. During testing, the ABySS assembler was used to generate preliminary assemblies, however at this time no testing has been done as to the effect the preliminary assembler has on downstream assembly.

Regardless of the tools used, the final scaffold FASTA file will be used for input into SIDR.

1.2.2 Alignment

The second piece of data required is an alignment of your raw reads to the preliminary assembly. The alignment can be constructed using any standard alignment tools, during testing GSNAP was used. Regardless of the tools used, the alignment must be in a sorted and indexed BAM file. These can be created from a SAM alignment using the following samtools commands:

```
samtools view -Sb /path/to/alignment.sam -o /path/to/alignment.bam
samtools sort /path/to/alignment.bam /path/to/alignment_sorted
samtools index /path/to/alignment_sorted.bam
```

1.2.3 BLAST

The last piece of data that must be precalculated is a BLAST classification of the preliminary assembly. This may be constructed with a tool besides command-line BLAST, so long as it is properly formatted. To make a properly-formatted BLAST result file, you can use the command:

```
blastn \
    -task megablast \
    -query /path/to/FASTA \
    -db nt \
    -outfmt '6 qseqid staxids' \
    -culling_limit 5 \
    -evalue 1e-25 \
    -out /path/to/output
```

Currently SIDR assumes that BLAST input will have the sequence ID in the first column, and the NCBI Taxonomy ID in the second column. Any alternative classification tool may be used so long as it can produce this output. Any additional columns in the BLAST output will be ignored.

1.2.4 Taxonomy Dump

SIDR uses the NCBI Taxonomy to translate the BLAST results into the desired classification. The Taxonomy dump can be downloaded from:

```
ftp://ftp.ncbi.nlm.nih.gov/pub/taxonomy/taxdump.tar.gz
```

After downloading, extract it and note it's location. By default, SIDR checks the directory listed in $BLASTDB, however this can be changed at runtime.

1.3 Running With Raw Input

By default, SIDR will analyse your data and construct a Decision Tree model based on the GC content and average coverage of your contigs. Before running SIDR, you will need to prepare some data based on your input. This is described here: *Data Preparation*

To run SIDR with the default settings on raw data, enter a command like:

```
sidr default -d [taxdump path] \
-b [bamfile] \
-f [assembly FASTA] \
-r [BLAST results] \
-k tokeep.contigids \
-x toremove.contigids \
-t [target phylum]
```

1.4 Running With a Runfile

SIDR can take a "runfile" with pre-computed variables as input. The runfile should be a comma delimited file starting with a header row. A column named "ID" which contains the contigID must exist, along with an "Origin" column with the name of the organism identified by BLAST for contigs where one was found. All other columns are used as variables for the decision tree, except any titled "Covered_bases", "Plus_reads", or "Minus_reads" as those are present in BBMap default output yet should not contribute to model construction.

To run SIDR in runfile mode, enter a command like:

```
sidr runfile -d [taxdump path] \
-i [runfile path] \
-k tokeep.contigids \
-x toremove.contigids \
-t [target phylum]
```

1.5 Runfile Example

SIDR's runfile mode accepts any comma-delimited file with at least an "ID" column with the contigID, an "Origin" column with the species name as identified by an outside classification tool, and one or more variable columns with which to construct the model. One way to construct a runfile with GC content and coverage variables calcluated with BBTools is described below. For this example, you will need:

- A preliminary assembly
- An alignment back to that preliminary assembly
- A copy of the NCBI Taxonomy Dump
- A local BLAST database

1. **BLAST the assembled data**:

```
blastn \
-task megablast \
-query [assembly FASTA] \
-db nt \
-outfmt '6 qseqid qlen staxids bitscore std sscinames sskingdoms stitle' \
```

(continues on next page)

(continued from previous page)

```
-evalue 1e-25 \
-max_target_seqs 2 \
-out blast.out
```

2. **Select the best BLAST hits**:

```
cat blast.out | awk '!_[$1]++' | cut -f 1,2,15 | sed 's/scaffold_//g' | sort -k1n␣
↪> scaffold_identities.txt
```

3. **Use BBTools to calculate GC content and coverage**:

```
pileup.sh countgc=t out=[organism].out in=[assembly BAM] ref=[assembly FASTA]
```

4. **Format the output from BBTools**:

```
cat [organism].out | sed '1d' | sed 's/scaffold_//g' | sort -k1n > [organism].
↪sorted
```

5. **Combine the BBTools and BLAST outputs**:

```
paste [organism].sorted scaffold_identities.txt | cut -f 1-9,12 | sed 1i"ID      ␣
↪   Avg_fold                Length   Ref_GC  Covered_percent                ␣
↪Covered_bases    Plus_reads           Minus_reads        Read_GC             ␣
↪Origin" | tr '\t' ',' > [organsim].csv
```