SCAPP documentation

David Pellow

Updated August 2020

1 Software documentation

SCAPP is fully documented at `https://github.com/Shamir-Lab/SCAPP`. We outline the installation and usage instructions here.

1.1 Installing SCAPP

1.1.1 With Conda

SCAPP can be installed with Conda as follows:

Download the installation file `install_scapp.yaml` in a directory of your choosing, for example with the command:

```
wget https://raw.githubusercontent.com/Shamir-Lab/SCAPP/master/install_scapp.yaml
```

Then create and activate the Conda environment:

```
conda env create -f install_scapp.yaml
conda activate scapp
```

Now you can run SCAPP as described below by entering the command `scapp`.

1.1.2 From Bioconda

Alternatively, SCAPP can be installed directly from Bioconda with the command:

```
conda install -c bioconda scapp
```

1.1.3 From sources

If not using Conda, SCAPP can be built from the sources following the instructions here.

SCAPP is written in Python3 and requires a number of packages which will be installed by the setup script.

SCAPP requires BWA, the NCBI BLAST+ executables, and samtools. We also highly recommend installing PlasClass to use full functionality of the SCAPP pipeline (available from: `https://github.com/Shamir-Lab/PlasClass`).

To install SCAPP do:

```
git clone https://github.com/Shamir-Lab/SCAPP.git
cd SCAPP
python setup.py install
```

We recommend using a virtual environment.

1.2 Configuring SCAPP paths

If not installed using Conda, SCAPP requires configuration variables specifying the paths to the BWA, BLAST+, and samtools executables for the system it is being run on.

Set the paths to these executables in the file `config.json`.

This information can be configured once when first installing SCAPP and then used for all SCAPP runs provided the locations of those executables do not change.

Alternatively, the locations of these executables can be added to the `PATH` environment variable. If these executables are in the `PATH` environment variable so that they can be run from any location on the system without specifying the path, then the SCAPP configuration variables can be left blank and there is no need to alter the file `config.json`.

1.3 Testing your installation

Once you have installed and configured SCAPP you can test your installation by running the shell script `run_test.sh`.

In the event that the test fails for any reason, first ensure that you have correctly set up the environment, installed, and configured SCAPP as described in the documentation. Any remaining problems should be reported through the Issues page of SCAPP's GitHub repository so that we can provide support (`https://github.com/Shamir-Lab/SCAPP/issues`).

1.4 Running SCAPP: Basic Usage

SCAPP is run using the command `scapp` as follows:

```
scapp -g <assembly graph> -o <output directory> -k <max k value> -r1 <reads 1> -r2 <reads 2>
```

If a BAM file aligning the reads to assembly graph nodes already exists (for example from a previous run of SCAPP), then the BAM file can be used instead of the reads files rather than having the pipeline redo the alignment:

```
scapp -g <assembly graph> -o <output directory> -k <max k value> -b <BAM>
```

The basic command line options are:

- `-g/--graph`: Assembly graph fastg file.
- `-o/--output_dir`: Output directory.
- `-k/--max_k`: Maximum k value used by the assembler. Default: 55.
- `-p/--num_processes`: Number of processes to use. Default: 16.
- `-r1/--reads1`: Paired-end reads file 1.
- `-r2/--reads2`: Paired-end reads file 2.
- `-b/--bam`: BAM alignment file aligning reads to graph nodes. `-b` and `-r1,-r2` are mutually exclusive.

The `-g`, `-o` and either `-r1`,`-r2` or `-b` options are required for every run of SCAPP.

1.5 SCAPP output files

SCAPP creates a number of subdirectories and outputs files in the output directory specified by the user. Key output files are highlighted:

output directory	specified by the user
|- <prefix>.confident_cycs.fasta	output plasmids fasta file (<prefix> is the base name of the assembly graph)
|- logs	log files output by the SCAPP pipeline
|- scapp.log	log file of the main SCAPP algorithm
|- bwa_std.log	output of running BWA read alignment
|- blast_std.log	output of running BLAST to find plasmid genes
|- plasclass_std.log	output of running PlasClass to assign plasmid scores
|- intermediate_files	files output by the SCAPP pipeline
|- reads_pe_primary.sort.bam(.bai)	alignment files of reads to assembly graph, created by BWA
|- plasclass.out	plasmid scores assigned to the assembly graph nodes by PlasClass
|- hit_seqs.out	list of assembly graph nodes with plasmid gene hits
|- <prefix>.cycs.fasta	fasta file of *all* cycles passing the cycle criteria (<prefix> is the base name of the assembly graph)
|- <prefix>.cycs.paths.txt	the edges that make up each cycle in <prefix>.cycs.fasta
|- <prefix>.self_loops.fasta	fasta of the cycles in <prefix>.cycs.fasta that consist of self-loops
|- <prefix>.gene_filtered_cycs.fasta	fasta of the cycles in <prefix>.cycs.fasta that have plasmid gene hits
|- <prefix>.classified_cycs.fasta	fasta of the cycles in <prefix>.cycs.fasta that are classified as plasmids by PlasClass

The primary output is the file <prefix>.confident_cycs.fasta which contains the confident plasmid predictions (prefix is the name of the input assembly graph file less the suffix).

The scapp.log file contains a log of the SCAPP run.

<prefix>.cycs.fasta contains all the potential plasmids – cycles that pass the cycle criteria. The confident cycles that are predicted as plasmids are a subset of these. Users may want to examine these potential plasmids.

<prefix>.cycs.paths.txt contains the paths for each potential plasmid. The name of the plasmid is listed. On the next line, the names of the set of edges that are contained in the cycle are listed in order. The third line for each entry lists the numbers of those edges (for easier use with visualisation tools).

The files reads_pe_primary.sort.bam, reads_pe_primary.sort.bam.bai, and plasclass.out contain the outputs of preprocessing steps in the SCAPP pipeline. These steps are the most time consuming, and the files can be passed into the pipeline (using the -b and -pc parameters) in order to skip these steps in case SCAPP is run multiple times on the same sample.

1.6 Advanced options

The SCAPP pipeline is configurable and there are many advanced options to set different thresholds and inputs in the algorithm. In most cases, we advise using the default settings and pipeline configurations.

There are a number of ways the stages of the SCAPP can be modified with the following parameters:

- -sc/--use_scores: Flag to determine whether to use plasmid scores. Use value False to turn off plasmid score use. Default True.
- -gh/--use_gene_hits: Flag to determine whether to use plasmid specific genes. Use value False to turn off plasmid gene use. Default True.
- -pc/--plasclass: PlasClass score file. If PlasClass classification of the assembly graph nodes has already been performed, provide the name of the PlasClass output file.

- `-pf/--plasflow`: PlasFlow score file. To use PlasFlow scores for the nodes instead of PlasClass, provide the name of the PlasFlow output file. `-pf`, `-pc` are mutually exclusive.

The following parameters change thresholds used in the algorithm:

- `-m/--max_CV`: Maximum allowed coefficient of variation for coverage. Default: 0.5.
- `-l/--min_length`: Minimum allowed length for potential plasmid. Default: 1000.
- `-clft/--classification_thresh`: Threshold for classifying a potential plasmid as a plasmid. Default: 0.5.
- `-gm/--gene_match_thresh`: Threshold for % identity and fraction of length covered to determine plasmid gene matches. Default: 0.75.
- `-sls/selfloop_score_thresh`: Threshold plasmid score above which a self-loop is considered a potential plasmid. Default: 0.9.
- `-slm/--selfloop_mate_thresh`: Threshold fraction of off-loop mate-pairs, below which a self-loop is considered a potential plasmid. Default: 0.1.
- `-cst/--chromosome_score_thresh`: Threshold score, below which a long node is considered a chromosome node. Default: 0.2.
- `-clt/--chromosome_length_thresh`: Threshold length, above which a low scoring node is considered a chromosome node. Default: 10000.
- `-pst/--plasmid_score_thresh`: Threshold score, above which a long node is considered a plasmid node. Default: 0.9.
- `-plt/--plasmid_length_thresh`: Threshold length, above which a high scoring node is considered a plasmid node. Default: 10000.
- `-cd/--good_cyc_dominated_thresh`: Threshold for the maximum fraction of nodes with most mate-pairs off the cycle allowed for the cycle to be considered a potential plasmid. Default: 0.5.

Note that instead of setting each of these parameters on the command line, they can instead be set using the file `params.json`. Simply set each variable in this file to the desired value and it will be used in SCAPP. Any value passed as a command-line parameter will override the values set in this file.