SCalable Metagenomics Pipeline (SCaMP) User Guide

VERSION 0.10

JAMES ABBOTT (J.ABBOTT@IMPERIAL.AC.UK)

Contents

1 Introduction

The SCalable Metagenomics Pipeline (SCaMP) is a system for high-throughput analysis of shotgun metagenome samples. It combines many tools (selected as being the most effective in our evaluations) to determine community composition, gene representation and abundance through metagenome assembly and annotation and pathway representation.

2 Installation

SCaMP requires a number of prerequisite packages to be installed, in addition to the SCaMP software itself. This can either be achieved by installing all the required packages and Perl modules manually, or through the use of conda, which is the recommended approach since it greatly simplifies installation.

2.1 Quick installation with Conda (Recommended)

You will need to first install both git and miniconda, and setup conda channels as described on the bioconda installation page: `https://bioconda.github.io/#install-conda`.

2.1.1 SCaMP installation

SCaMP can be then be downloaded from the git repository using the command:

```
git clone https://github.com/jamesabbott/SCaMP.git
```

Once the repository is cloned, the prerequisite packages can be installed from conda using the `setup.sh` script:

```
SCaMP/bin/setup.sh
```

The setup script will create a new conda environment name 'SCaMP' to avoid conflicts with any existing conda installations, and use this to install the prerequisite packages. Once completed, the script will give the option of appending the SCaMP bin directory to your path. This will ensure the SCaMP commands are readily available from the command line.

2.1.2 Customising the SCaMP Environment

If your system requires specific configuration such as manually setting a path, or loading an environment module to access conda, then the `bin/SCaMP` script can be edited to include any necessary settings. See the comments at the top of the script for examples of the kind of configuration which may be configured. The `setup.sh` script will have uncommented a line in this section to activate the SCaMP conda environment each time the software is run.

2.2 Manual Installation (The Slow Way...)

2.2.1 Perl Modules

A number of non-standard perl modules need to be installed prior to installing SCaMP:

1. BioPerl
2. Bio::DB::EUtilities
3. DateTime
4. File::Copy::Recursive
5. File::Find::Rule
6. HTML::Entities
7. LWP
8. LWP::Protocol::https
9. Net::FTP::Recursive
10. Parallel::ForkManager
11. XML::Simple
12. YAML::XS

Most of these will be available through your systems package management system (using `yum` or `apt`), however others may require installation from CPAN.

2.2.2 Prerequisite packages

A number of software packages also need to installed, and made available on the system path prior to running SCaMP. These are listed in table 1, along with tested version numbers.

Package	Tested Version	Download Site
bwa	0.7.15	http://bio-bwa.sourceforge.net
cutadapt	1.15	https://github.com/marcelm/cutadapt
FastQC	0.11.5	https://www.bioinformatics.babraham.ac.uk/projects/fastqc
picard	2.15.0	http://broadinstitute.github.io/picard
R	3.3.2	http://r-project.org
samtools	1.3.1	http://www.htslib.org
trim-galore	0.45	https://www.bioinformatics.babraham.ac.uk/projects/trim_galore

Table 1: Required software packages

2.2.3 SCaMP installation

SCaMP can be now be downloaded from the git repository using the command:

```
git clone https://github.com/jamesabbott/SCaMP.git
```

2.2.4 Customising the SCaMP Environment

If your system requires specific configuration such as manually setting a path, or loading an environment module to access conda, then the `bin/SCaMP` script can be edited to include any necessary settings. See the comments at the top of the script for examples of the kind of configuration which may be configured.

3 About SCaMP

3.1 The Pipeline

The SCaMP workflow consists of a number of independent stages (Figure 1), and is designed to be run under a batch-queuing environment (PBSPro or Sun Grid Engine). A work directory needs to be created on shared

storage accessible by the compute nodes under the control of the queueing system, and fastq files copied into a `reads` directory within this work directory. Depending upon the compute configuration, data can either be copied to local storage on compute nodes, or analysed directly within the work directory. The path to the work directory needs to be defined within a SCaMP.yaml configuration file. Output files will be written within the work directory as the various stages of the pipeline are run. The stages of the SCaMP workflow need to be executed in the correct order - stages will look for the outputs files of preceding stages in the workflow as their input files, consequently executing the stages out of sequence will result in failed jobs. A summary of the various stages and execution order is shown in table 2.

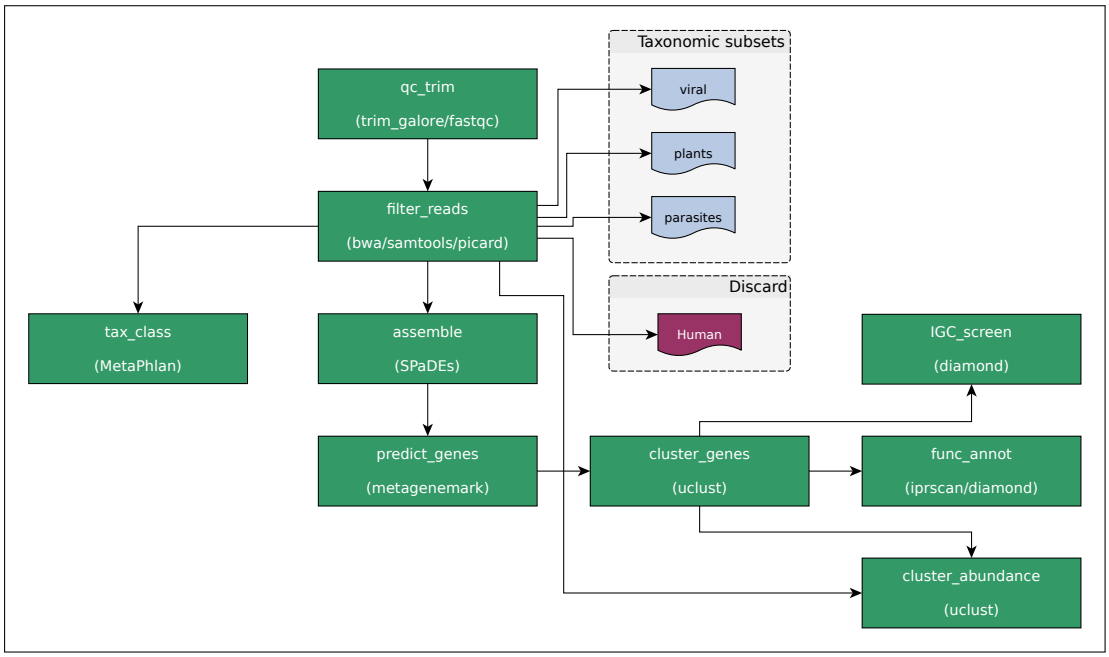

Figure 1: SCaMP Workflow

Stage	Order	Summary
qc_trim	1	Carries out QC analysis of sequence reads, and trims low-quality sequence and adapters
filter_reads	2	Filters reads against databases of known sequences to remove e.g. host contaminants

Table 2: Stages of the SCaMP workflow, including execution order.

3.2 Software Layout

The SCaMP software is laid out across a few directories, the contents of which are hopefully self-explanatory:

`bin` - Executable scripts:

- **build_ref_db**: Downloads and indexes reference databases.
- **filter_reads**: Filters sequence reads against reference database.
- **qc_trim**: Runs QC on sequence reads and applies quality and adapter trimming.
- **SCaMP**: Main script for launching analysis runs.
- **setup.sh**: Installs prerequisite packages and configures system.

`doc` - Documentation

- **SCaMP_User_Guide.tex**: LaTeX source for user guide.

- **SCaMP_User_Guide.pdf**: PDF format user guide.

`etc` - Configuration data

- **barcodes.txt**: Example barcode sequences for adapter trimming.
- **conda_packages.txt**: List of conda packages to be installed by `bin/setup.sh`.
- **SCaMP.yaml**: YAML format configuration file.

`lib` - Common software components used by scripts in `bin`

- **SCaMP.pm**: Class of common methods used by scripts in `bin`.

3.3 Preparing to Run SCaMP

3.3.1 Configuration File

The `etc/SCaMP.yaml` defines the SCaMP configuration. In shared installations, each user can copy this file to a file named `.SCaMP.yaml` in their home directory. If this file is found, then the configuration will be read from this file in preference to the `etc/SCaMP.yaml` file.

The configuration file should be edited to define the following attributes:

- **work_dir**: Path to directory where SCaMP analysis will be carried out
- **database_dir**: Path to directory for storing reference databases

A series of directories will be created in `work_dir` as the analysis proceeds, storing various intermediate files and analysis results.

In a shared software installation, a single set of reference databases can be downloaded and stored in a centralised `database_dir` to reduce disk space usage.

3.3.2 Sequence Reads

A directory should be created in `work_dir` named `reads`, and fastq files (preferably compressed using gzip) for the project should be copied into this directory.

SCaMP is designed to be run using paired reads from an Illumina sequencer, typically a MiSeq or HiSeq. In order for the reads to be identified correctly they should be named in accordance with one of the following convention:

samplename(_barcode)?(_xxxxx)?_R[12].f(ast)?q.gz

The filename should start with the name of the sample, followed by an optional barcode (typically six or eight bases). Additional details may optionally be included in the next section of the filename, following which the filename must include either _1/_2, or _R1/_R2 (indicating read 1 or read 2 of the paired reads). The filename should be suffixed with either .fastq.gz or fq.gz.

Examples of valid filenames include:

- *Sample-001_R1.fastq.gz*
- *Sample-002_R2.fq.gz*
- *Sample-003_12.fq.gz*
- *Sample-004_TAATCG_L005.R1.fastq.gz*

3.3.3 Reference Databases

SCaMP allows samples to be filtered against various taxonomic databases to remove e.g. host contamination. The databases should be located within the directory specified by the `database_dir` parameter in the `SCaMP.yaml` configuration file. A separate subdirectory should be created for each database (named with the database name), and within this multiple copies of the database can exist in e.g. time-stamped directories, with a symlink named 'latest' to the most recent copy of the database, which will be used for analysis.

Reference Database Format

Each database used by for filtering by these scripts requires the following:

1. A fasta formatted sequence of sequences, named 'database.fa'
2. A fasta index, generated with 'samtools faidx' ('database.fa.fai')
3. BWA indices appropriate for searching with BWA MEM
4. A tab-delimited file ('database.tax.dat') relating taxonomy data to each sequence ('database.taxid.dat')

The tab-delimited file should be formatted as follows:

```
##Accession   Scientific name  Strain  NCBI TaxId
chr1    Homo sapiens            9606
578454_CABE01000000     Candida parapsilosis    CDC317  578454
```

In the Homo sapiens example there is no strain defined, so this field is simply left blank

Downloading Databases

The `bin/build_ref_db` script allows preconfigured reference databases to be downloaded and appropriately formatted, with the resulting databases being made available in `database_dir`. A list of the databases available for downloaded can be obtained by running

```
build_ref_db -avail
```

Downloading and formatting a database can be achieved by running

```
build_ref_db -db dbname
```

Database Configuration

Configurations for the available databases are defined in the `databases` block of the SCaMP configuration file. These can be modified as required to create custom databases for screening sequence reads. The following parameters may be defined to configure a database:

- **format**: Format of reference database. Valid values: embl/genbank
- **type**: Download type. Valid values: ftp/entrez/entrez_accession.
  - *ftp*: Defines a database to be downloaded from an ftp server. Requires *ftphost*, *ftpdir* and *filepattern* parameters to be defined.

  - *entrez*: Defines a database to be retrieved from the NCBI using an entrez query. Requires *entrez_query* parameter to be defined.
  - *entrez_accession*: Defines a database to be retrieved from the NCBI using entrez from a list of accessions. Requires *accession* parameter to be defined.
- **ftphost**: URI of FTP server i.e. `ftp.ensemblgenomes.org`
- **ftpdir**: Path to directory on FTP server where database is found. i.e. `/pub/plants/current/embl`
- **filepattern**: Regular expression defining list of files to be downloaded. i.e. `.dat.gz$`
- **entrez_query**: Entrez query to select records for inclusion in database i.e. `"Viruses"[Organism]`
- **accessions**: List of accession to be included in database

4 Running SCaMP

SCaMP is designed to be run under either a PBSPro or Sun Grid Engine batch-queueing environment. While all the components of the pipeline can be submitted manually, the `SCaMP` script will take care of submitting the correct number of array tasks to the available queueing system.

Each component (or stage) or the pipeline is run by a separate script located in the `bin` directory. SCaMP requires the stage of the pipeline to run to be defined, and will pass through additional command-line arguments to the individual stage script. Provided arguments will be verified before the jobs is submitted to the queueing system.

Running `SCaMP` without any arguments provides usage information, along with a list of the available stages which can be run. **N.B. stages must be executed in the correct order, as described below.**

Update output...

```
[jamesa@wssb-james ]$ SCaMP

SCaMP: SCalable Metagenomics Pipeline
usage: /home/jamesa/src/SCaMP/bin/SCaMP -r run_stage [-l|--local] [-- script arguments]

Configured stages: qc_trim filter_reads

[jamesa@wssb-james doc]$
```

SCaMP is designed to be run either on cluster environments with local storage available on each compute node, or on systems with high-performance parallel filesystems i.e. GPFS/Lustre. In a typical cluster environment, it is preferable for data to be copied to a temporary directory on the execution node to ensure heavy IO loads take place on faster local storage and reduce network load on the cluster. In contrast, systems with parallel file systems or large shared memory machines with locally attached storage can manage the IO loads from shared storage adequately, hence it is optimal not to move data to a temporary directory for computation.

By default, SCaMP will copy databases and sample data to a temporary directory on each cluster node. If your environment benefits from fast shared storage, then SCaMP can be run using the `-l` or `-local` arguments, in which case the databases and sample data will not be copied to local temporary directories. For example, the command:

```
SCaMP -r qc_trim --local
```

will execute the 'qc_trim' stage using local storage. Consult your local system maintainers if you are unsure of whether local storage or temporary storage on the compute nodes should be used in your environment.

Executing the 'SCaMP' script results in the command-line parameters being validated by the stage script, to ensure that all necessary arguments have been provided, and that the directories/files required for the stage to run are available prior to submitting the job. Once these have been validated successfully, the job will be submitted to the PBS or SGE system and the job id of the stage will be displayed on the screen. Progress of the job can then be monitored using the `qstat` command.

Once the job completes, any standard output or standard error generated during it's execution will be writen to a file named `SCaMP:stage.oJOBID.TASKID` in the directory from which the job was submitted.

4.1 SCaMP Stages

The following section descriptions obtained from the embedded help documentation within each script. Please refer to 3.1 for the correct ordering in which the stages should be run.

4.1.1 qc_trim

SYNOPSIS qc_trim [--qual 20] [--length 50]

DESCRIPTION qc_trim carried out initial quality and adapter trimming of sequence reads, using trim_galore.

Gzip compressed sequence reads should be placed in a 'reads' subdirectory of the work_dir defined in SCaMP.yaml.

Read files should be named according to the following scheme:

sample_(barcode)?(_xxxxx)?_(R)?[12].[fastq|fq].gz

i.e. SP-128_CATGGC_L005_R1.fastq.gz

See the ScaMP User Guide for further examples of valid filenames.

By default, a quality threshold of 20 is used, along with a minimum read length to retain of 50 bases. N's at the end of reqds are also removed.

trim_galore will automatically search for Illumina universal and Nextera transposase adapters. Should other primer sequences need to be trimmed, these can be provided in a tab-delimited file using the **adapters** argument. An example file is included in the etc directory, named **barcodes.txt**.

OUTPUT FILES Following trimming, the outputs will be written to a 'trimmed' subdirectory in work_dir. A separate subdirectory will be created for each sample, containing the following files:

sample_[12].fastq.gz.trimming_report.txt

TrimGalore report on trimmed bases

sample_[12]_val_[12]_fastqc.html

HTML FastQC results

sample_[12]_val_[12].fq.gz

Trimmed sequence reads, in gzipped fastq format

OPTIONAL ARGUMENTS

qual

Set quality threshold (default: 20)

Bases with a quality score below this value will be trimmed from the end of sequences

length

Seq minimum read length (default: 50)

Reads shorter than this value will be discarded following quality and adapter trimming

adapters

Path to tab-delimited file of bar-coded adapter sequences to remove. A tab-delimited file can be specified containing specific adapter sequences for individual barcodes. This should contain 3 columns:

Barcode

The distinct barcode for this sample, which must be included in the fastq file names.

Adapter1

The first (5') adapter sequence

Adapter2

The second (3') adapter sequence

SCAMP ARGUMENTS The following arguments should only be provided if running the command directly for a command line, and *not* when submitting jobs through the SCaMP script, which will take care of setting these appropriately.

stage

Stage data to local storage

If this argument is provided, sample data will be copied to a local temporary directory on the compute node running the job, otherwise the data will be processed *in-situ*

check

Check command syntax

Provides a check that the provided syntax/file-naming is correct to ensure jobs submitted to the batch system are valid.

help

Display help text

man

Display manual page

4.1.2 filter_reads

SYNOPSIS filter_reads --db [db_name]

DESCRIPTION Filters sequence reads against a database and removes any aligned reads from the fastq files. Alignments are carried out using the BWA mem algorithm, and the resulting alignments split into separate bam files for aligned and unaligned reads. Reads which are successfully aligned are taxonomically classified based on the hits, while new fastq files consisting of reads which do not align against the database are produced.

INPUT AND OUTPUT FILES The outputs from the filtering will be written to a *filtered/databasename* directory within work_dir. A symbolic link (*latest*) will indicate the outputs of the most recent filtering operation, since multiple databases filtering outputs may be present in this directory. The fastq files in the *latest* linked directory will be used as input for the next filter_reads run, such that filtering operations are carried out successively. If no *filtered/latest* linked directory exists then fastq files in the *trimmed* directory will be used as inputs to the filtering process instead.

Output files are split into separate subdirectories for each sample, with each subdirectory containing the following output files:

sample.database.bam

BAM format file containing all reads following alignment agsint the selected database.

sample.database.aligned.bam

BAM format alignment of reads which align against the selected database.

sample.database.aligned.bam

BAM format file containing reads which do not align against the selected database.

sample.database.filtered_reads.txt

Text file containing list of reads ID's which align against the database and have therefore been filtered from the output fastq files.

sample.database.mapping_report.txt

Text format report containing breakdown of number of reads aligned to different target sequences in the selected database, and the total number of reads aligned to each taxon represented in the database

sample.database.filtered_[12].fq.gz

Fastq files containing reads following removal of sequences matching database sequences.

REQUIRED ARGUMENTS

db

Database to align reads against

SCAMP ARGUMENTS The following arguments should only be provided if running the command directly for a command line, and *not* when submitting jobs through the SCaMP script, which will take care of setting these appropriately.

stage

Stage data to local storage

If this argument is provided, sample data will be copied to a local temporary directory on the compute node running the job, otherwise the data will be processed *in-situ*

check

Check command syntax

Provides a check that the provided syntax/file-naming is correct to ensure jobs submitted to the batch system are valid.

help

Display help text

man

Display manual page

4.2 Example Analysis

The following commands illustrate a typical run of the SCaMP pipeline in the correct order of execution.

```
SCaMP -r qc_trim
SCaMP -r filter_reads --db human
SCaMP -r filter_reads --db plants
SCaMP -r filter_reads --db fungi
SCaMP -r filter_reads --db parasites
```