cgat single cell Documentation

Release 1.0

CGAT Developers

Jul 05, 2023

Getting started

CGAT-sc is a collection of single cell pipelines that allow users to perform some of the standard workflows associated wit single cell analysis. This repository is still in development but the aim is to collate a series of single cell workflows that allow the

The workflow management systems that underpins all of our pipelines is **'CGAT-core <>https://github.com/cgat-developers/cgat-core>'_**. We demonstrate the functionality of CGAT-core using a simple RNA-seq pipeline in cgat-showcase. Therefore, if you are not familiar with how we build our workflows I suggest that you look at these two pipelines first.

CHAPTER 1

Citation

Continued development, will most likely be cited as part of a scientific publication. Watch this space. . . .

CHAPTER 2

Support

- Please refer to our FAQ section
- For bugs and issues, please raise an issue on github
- For contributions, please refer to our contributor section and github source code.

CHAPTER 3

Selected publications using CGAT-sc

These publications will appear here. Watch this space as this is a new repository.

3.1 Installation

The following sections describe how to install the scflow framework.

3.1.1 Conda Installation

The preferred method for installation is through conda. Currently this installation is still in working progress. Preferably the installation should be in a separate environment.:

```
# The conda environment is currently not ready but we are working on this
#conda create -n scflow -c cgat scflow
#conda activate scflow
#scflow --help
```

3.1.2 Pip installation

You can install scflow using pip, this will only install the package without any dependancies, which will have to be installed separately.:

```
pip install scflow
```

3.1.3 Manual Installation

The repository can also be installed manually, but dependancies will need to be installed separately.:

```
git clone https://github.com/Acribbs/scflow.git
python setup.py install
scflow --help
```

3.1.4 Installing additional software

We always recommend using conda to install additional software where possible.

This can be easily performed by:

```
conda search <package>
conda install <package>
```

3.2 Cluster configuration

Currently SGE, SLURM, Torque and PBSPro workload managers are supported. The default cluster options for cgatcore are set for SunGrid Engine (SGE). Therefore, if you would like to run an alternative workload manager then you will need to configure your settings for your cluster. In order to do this you will need to create a `.cgat.yml` within the user's home directory.

This will allow you to overide the default configurations. To view the hardcoded parameters for cgatcore please see the parameters.py file.

For an example of how to configure a PBSpro workload manager see this link to this config example.

The .cgat.yml is placed in your home directory and when a pipeline is executed it will automatically prioritise the `.cgat.yml` parameters over the cgatcore hard coded parameters. For example, adding the following to the .cgat.yml file will implement cluster settings for PBSpro:

```
    memory_resource: mem

options: -l walltime=00:10:00 -l select=1:ncpus=8:mem=1gb

queue_manager: pbspro

queue: NONE

parallel_environment:  "dedicated"
```

3.3 singlecell

3.3.1 Overview

This pipeline performs alignment free based quantification of drop-seq, 10X single-cell sequencing analysis using wither kallisto or salmon. Pseudoalignment is performed on the RNA reads, using kallisto or Alevin and the resulting data is quantitatively and qualitatively analysed.

The pipeline performs the following analyses: * Alignment using kallisto or alevin (part of salmon) * QC of reads using the scater package

Input files

The pipeline is ran using fastq files that follow the naming convention Read1: Name.fastq.1.gz and read2: Name.fastq.2.gz.

- a fastq file (paired end following the naming convention below)
- a GTF geneset

The default file format assumes the following convention: fastq.1.gz and fastq.2.gz for paired data, where fastq.1.gz contains UMI/cellular barcode data and fastq.2.gz contains sequencing reads. Chromium output is of the format: samplename_R1.fastq.gz and samplename_R2.fastq.gz so will require conversion to the default file format above.

Configuring the pipeline

[describe how to set config values]

Running the pipeline

To run the pipeline you will need to set up the cluster configuration according to the cluster documentation.

However the pipeline can also be run locally without the cluster using the commandline flag *–no-cluster*.

The following command will run the pipeline:

```
scflow singlecell make full -v5
```

Report generation

The pipeline also generates Rmarkdown reports by running the following command:

```
scflow singlecell make build_report -v5
```

output

3.4 kallistobus

3.4.1 Overview

This pipeline performs alignment free based quantification using the kallisto bus tools pipeline. For further information on the code that is wrapped up in this pipeline please refer to to the following documentation

This pipeline runs kallisto bustools and runs alignment, counting and outputs either a loom or h5ad file that can be further imported into Seurat or Scanpy.

You have the option of running the following quantification: * standard - will generate a standard pseudoaligned analysis output * lamanno - for RNA- velocity analysis based on la Manno et al 2018 * kite - for feature barcoding * kite:10xFB - feature barcoding for 10X genomics

A number of technologies are supported by kallisto bustools: * 10XV1 * 10XV2 * 10XV3 * CELSEQ * CELSEQ2 * DROPSEQ * INDROPSV1 * INDROPSV2 * INDROPSV3 * SCRUBSEQ * SURECELL

All of these options are set within the pipeline.yml configuration file. Please see more info below.

Input files

The pipeline is ran using fastq files that follow the naming convention Read1: Name.fastq.1.gz and read2: Name.fastq.2.gz.

- a fastq file (paired end following the naming convention below)
- a GTF geneset

The default file format assumes the following convention: fastq.1.gz and fastq.2.gz for paired data, where fastq.1.gz contains UMI/cellular barcode data and fastq.2.gz contains sequencing reads. Chromium output is of the format: samplename_R1.fastq.gz and samplename_R2.fastq.gz so will require conversion to the default file format above.

Configuring the pipeline

To set the input values for the pipeline you need to modify a configuration file. To generate this yml file run the following:

```
scflow kallistobus config -v5
```

Then open up the pipeline.yml file and modify the default values before running the pipeline.

Running the pipeline

To run the pipeline you will need to set up the cluster configuration according to the cluster documentation.

However the pipeline can also be run locally without the cluster using the commandline flag *–no-cluster*.

The following command will run the pipeline:

```
scflow kallistobus make full -v5
```

output

The output of the piepline is a bus record for each sample than can be combined using downstream tools such as Seurat or Scanpy.

3.5 Developers

The following people have contributed to this repository

Adam Cribbs

Jakub Scaber

Anna James-Bott

Carla Cohen

You can contribute to the development of our software in a number of different ways:

3.6 Reporting bug fixes

Bugs are annoying and reporting them will help us to fix your issue.

Bugs can be reported using the issue section in github

When reporting issues, please include:

- Steps in your code/command that led to the bug so it can be reproduced.
- The error message from the log message.
- Any other helpful info, such as the system/cluster engine or version information.

3.7 Proposing a new feature/enhancement

If you wish to contribute a new feature to the CGAT-sc repository then the best way is to raise this as an issue and label it as an enhancement in github

If you propose a new feature then please:

- Explain how your enhancement will work
- Describe as best as you can how you plan to implement this.
- If you dont think you have the necessary skills to implement this on your own then please say and we will try our best to help (or implement this for you). However, please be aware that this is a community developed software and our volunteers have other jobs. Therefore, we may not be able to work as fast as you hoped.

3.8 Pull Request Guidelines

Why not contribute to our project, its a great way of making the project better, your help is always welcome. We follow the fork/pull request model. To update our documentation, fix bugs or add extra enhancements you will need to create a pull request through github.

To create a pull request perform these steps:

1. Create a github account.
2. Create a personal fork of the project on github.
3. Clone the fork onto your local machine. Your remote repo on github is called `origin`.
4. Add the orginal repository as a remote called `upstream`.
5. If you made the fork a while ago then please make sure you `git pull upstream` to keep your repository up to date
6. Create a new branch to work on! We usually name our branches with capital first and last followed by a dash and something unique. For example: `git checkout -b AC-new_doc`.
7. Impliment your fix/enhancement and make sure your code is effectively documented.
8. Our code has tests and these will be ran when a pull request is submitted, however you can run our tests before you make the pull request, we have a number written in the `tests/` directory. For example: to run our import tests please run `nosetests tests/test_import.py`.
9. Add or change our documentation in the `docs/` directory.
10. Squash all of your commits into a single commit with gits interactive rebase.

11. Push your branch to your fork on github `git push origin`
12. From your fork in github.com, open a pull request in the correct branch.
13. ... This is where someone will review your changes and modify them or approve them ...
14. Once the pull request is approved and merged you can pull the changes from the `upstream` to your local repo and delete your branch.

Note: Always write your commit messages in the present tense. Your commit messages should describe what the commit does to the code and not what you did to the code.

3.9 Licence

CGAT-sc is an open-source project and we have made the cgat-developers repositor available under the open source permissive free MIT software licence, allowing free and full use of the code for both commercial and non-commercial purposes. A copy of the licence is shown below:

3.9.1 MIT License