cgat single cell Documentation

Release 1.0

CGAT Developers

Jul 13, 2023

GETTING STARTED

Droplet-based single-cell sequencing techniques have provided unprecedented insight into cellular heterogeneities within tissues. However, these approaches only allow for the measurement of the distal parts of a transcript following short-read sequencing. Therefore, splicing and sequence diversity information is lost for the majority of the transcript. The application of long-read Nanopore sequencing to droplet-based methods is challenging because of the low base-calling accuracy currently associated with Nanopore sequencing. Although several approaches that use additional short-read sequencing to error-correct the barcode and UMI sequences have been developed, these techniques are limited by the requirement to sequence a library using both short- and long-read sequencing. Here we introduce a novel approach termed single-cell Barcode UMI Correction sequencing (scBUC-seq) to efficiently error-correct barcode and UMI oligonucleotide sequences synthesized by using blocks of dimeric nucleotides.

TallyNN is a collection of single-cell workflows that allow users to perform barcode and UMI correction for oligonucleotide sequences that are synthesised using double phosphoramidites for droplet based single-cell sequencing.

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

ONE

CITATION

Watch this space....

CHAPTER

TWO

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 project_info/Contributingt source code.

2.1 Installation

The following sections describe how to install the TallyNN analysis pipelines.

First clone the Github repo and activate the UMI-tools submodule:

```
git clone https://github.com/Acribbs/TallyNN.git
git submodule init
git submodule update
```

We recommend installing [miniconda](https://docs.conda.io/en/latest/miniconda.html), then creating a new environment and install mamba:

```
conda create -n tallynn
conda install mamba -c conda-forge
```

Next install the required software:

```
mamba env upate conda/environment/tallynn.yml
```

Then, at the moment, you will need to manually install the fork of umi tools. The fork is provided as a submodule within TallyNN. You can install it as follows:

```
git clone https://github.com/Acribbs/UMI-tools.git
git checkout AC-dualoligo
python setup.py install
```

To install tallynn code, navigate to the tallynn directory and run:

```
python setup.py develop
```

2.2 Cluster configuration

Currently SGE, SLURM, Torque and PBSPro workload managers are supported. The default cluster options for cgat-core 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"
```

2.3 Tutorial guide

In the following page, we describe the steps involved in running TallyNN to error correct barcodes and UMIs. The pipeline can be executed using a test dataset that contains 100,000 Nanopore sequencing reads (https://www.cgat.org/downloads/public/adam/TallyNN/test_nanopore.fastq.gz) or full data (https://www.cgat.org/downloads/public/adam/TallyNN/E1.fastq.gz).

The fastq files are derived from an experiment in which an equal mix of JJN3, NCI-H929 and DF15 myeloma cells.

2.3.1 Download the files

The demo data is available at the following link - https://www.cgat.org/downloads/public/adam/TallyNN/:

```
wget https://www.cgat.org/downloads/public/adam/TallyNN/test_nanopore.fastq.gz
wget https://www.cgat.org/downloads/public/adam/TallyNN/Homo_sapiens-GRCh38.fa
wget https://www.cgat.org/downloads/public/adam/TallyNN/Homo_sapiens.GRCh38.101.bed
wget https://www.cgat.org/downloads/public/adam/TallyNN/hg38.fasta
wget https://www.cgat.org/downloads/public/adam/TallyNN/Homo_sapiens.GRCh38.101.gtf
mkdir data
mv test_nanopore.fastq.gz data/
```

To run the demo you will need to download the fasta file of human transcripts, the fastq file of nanopore sequenced reads, a genome fasta file and a junction bed file generated according to minimap2 [documentation](https://lh3.github.io/minimap2/minimap2.html).

Add the downloaded data to a directory called *data.dir/*

2.3.2 Generate the config file

In order to run the nanopore pipeline you will need to generate a configuration file that can be used to modify the execution of the pipeline.

This can be performed by:

```
tallynn nanopore config
```

This will generate a pipeline.yml file in the current directory. However for the purpose of this tutorial please download the pipeline.yml as follows:

```
wget https://www.cgat.org/downloads/public/adam/TallyNN/pipeline.yml
```

2.3.3 Modify the config file

Open up the pipeline.yml file. You can see a set of key value pairs that can be modified to change the running of the pipeline. For the current analysis, no changes need to be made to this file.

2.3.4 Run the pipeline

The defult behaviour is for the pipeline to execute across a cluster. This is why it is important to set up your *.cgat.yml* file correctly. In order to run the pipeline to completion run the following in the commandline:

```
tallynn nanopore make full -v5
```

Otherwise, the pipeline can also be ran locally without a cluster:

```
tallynn nanopore make full -v5 --no-cluster
```

2.4 Nanopore

2.4.1 Overview

This pipeline performs alignment free based quantification of scBUC-seq for single-cell sequencing long-read nanopore data.

The pipeline performs the following analyses: * Barcode and UMI correction * Alignment using kallisto * QC of reads and generation of a barnyard plot for species mixing experiments

Input files

The pipeline is ran using fastq files added to a directory called data/

- a fastq file
- a single cDNA of all transcripts. Downloaded from Gencode. e.g. Homo_sapiens.GRCh38.cdna.all.fa.gz. If you

have a species mixed experiment then also download Mus_musculus.GRCm38.cdna.all.fa.gz and cat both cdna files into one and modify the pipeline.yml file. * Genome fasta file, downloaded from Gencode

Configuring the pipeline

To generate a configuration file run:

```
tallynn nanopore config -v5
```

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:

```
tallynn nanopore make full -v5
```

output

The output of the pipeline is count matrix in market matrix format. This can be found within the directory mtx.dir/

2.5 Illumina

2.5.1 Overview

This pipeline performs alignment free based quantification of scBUC-seq for single-cell sequencing short-read illumina data.

The pipeline performs the following analyses: * Barcode and UMI correction * Alignment using kallisto * QC of reads and generation of a barnyard plot for species mixing experiments

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. The fastq files are added to a directory called data.dir/

- a fastq file (paired end following the naming convention below)
- a cDNA of all transcripts. Downloaded from Gencode. e.g. Homo_sapiens.GRCh38.cdna.all.fa.gz. If you

have a species mixed experiment then also download Mus_musculus.GRCm38.cdna.all.fa.gz.

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.

Configuring the pipeline

To generate a configuration file run:

```
tallynn illumina config -v5
```

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:

```
tallynn illumina make full -v5
```

output

The output of the pipeline is count matrix in market matrix format. This can be found within the directory kallistao.dir/<name_of_fastq_file>/bus/genecount/

2.6 Developers

The following people have contributed to this repository

Adam Cribbs

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

2.7 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.

2.8 Proposing a new feature/enhancement

If you wish to contribute a new feature to the TallyNN 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.

2.9 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.

2.10 Licence

TallyNN is an open-source project and we have made the repository 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:

2.10.1 MIT License