proatac Documentation

Release 0.2.1

Caleb Lareau

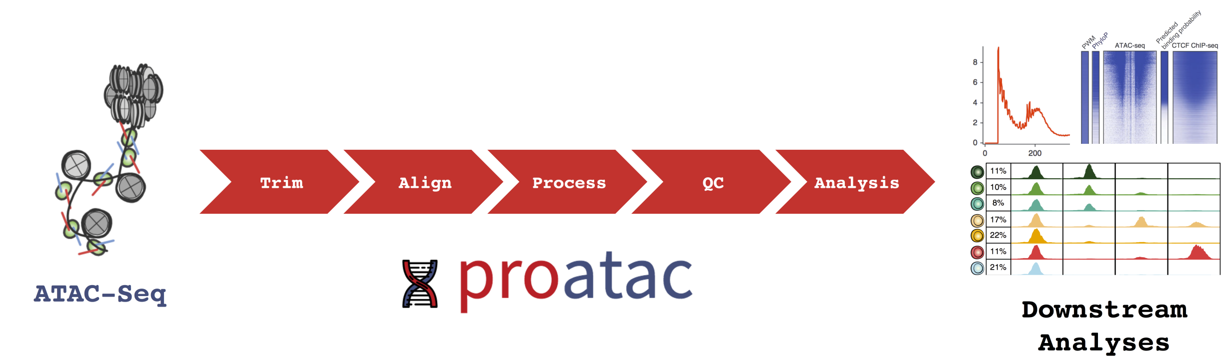
Feb 22, 2022

Contents

CHAPTER 1

About

proatac is an open-source command-line toolkit that performs robust and scalable preprocessing of ATAC-Seq data. Specifically, we've implemented our workflow using Snakemake, a robust, scalable computational workflow platform. Various snakefiles are wrapped alongside meta-data annotations and various object-oriented constructs and distributed as a *Python 3* package. The figure below provides a brief overview of the functionality of the **proatac** pipeline.



CHAPTER 2

Installation

2.1 Install stable version through PyPi

There are a few dependencies needed to get **proatac** to run. All are very common bioinformatics tools / languages and should be readily available in most systems. However, **note that the current implementation of proatac is not supported on Windows platforms**.

Depending on your python environment, we generally recommend using a virtual environment to keep python dependencies tidy. An example of installing **proatac** inside a new python virtual environment called `venv3` using the following sequence of commands–

```
python3 -m venv venv3
source venv3/bin/activate
```

The installation can then be specified using the following:

```
pip3 install proatac
```

2.2 Install via GitHub

Though **not recommended**, a bleeding-edge (development) version can be installed directly from Git. Again using a virtual environment–

```
python3 -m venv venv3
source venv3/bin/activate
pip3 install git+ssh://git@github.com/buenrostrolab/search/tree/master/proatac
```

While installing **proatac** is obviously a great first step, make sure that all of the dependencies are met. Check out the next page for more detail.

2.3 Installing proatac as an environment module.

A common use case of **proatac** will be processing ATAC-seq data in a high-performance computing cluster environment. As each computing cluster is different, you're probably best off inquiring with you sysadmin how to install **proatac** Here are a few general tips though (modified from the MultiQC installation guide):

A typical installation procedure with an environment module Python install might look like this: (Note that `$PYTHONPATH` must be defined before pip installation; this can be specified by creating a virtual environment and loading it as shown above)

```
# Create the proatac version (replace 0.3 with whatever the current version is)
VERSION=0.3
INST=/path/to/software/proatac/$VERSION
module load python/3.6.1
mkdir $INST
export PYTHONPATH=$INST/lib/python3.6/site-packages
pip3 install --install-option="--prefix=$INST" multiqc
```

Once **proatac** is installed, we need to create a new module file. While these vary between systems, but here's an example that also imports the necessary dependencies:

```
#%Module1.0######################################################################
##
## proatac
##

set components [ file split [ module-info name ] ]
set version [ lindex $components 1 ]
set modroot /path/to/software/proatac/$version

proc ModulesHelp { } {
    global version modroot
    puts stderr "\proatac - use proatac $version"
    puts stderr "\n\tVersion $version\n"
}
module-whatis   "Loads proatac environment."

# load required modules
module load python/3.6.1
module load samtools/1.5.0
module load R/3.4.0
module load java/1.6.0
module load macs2/2.1.1.20160309
module load bowtie2/2.3.1

# only one version at a time
conflict proatac

# Make the directories available
prepend-path    PYTHONPATH  $modroot/lib/python3.6/site-packages
```

CHAPTER 3

Dependencies

3.1 Getting proatac running

proatac has a few dependencies that are listed below with relevant hyperlinks for installation instructions from the source. To quickly determine what may be lacking in your system, try running **proatac** with the default.yaml file (more on that *here*) using the `--check` flag. To do this, we'll first clone the repository

```
git clone https://github.com/buenrostrolab/proatac.git
proatac yaml/default --check
```

If you get a message saying that the check was succesful, then you're most likely ready to begin analyzing data. However, if you run into one or more error messages, you are likely missing the necessarily software. Make sure that

- bedtools
- bowtie2 and relevant index for analysis.
- java language
- macs2

We note that macs2 though also a PyPi package is only compatible with Python 2.7 whereas **proatac** is a Python 3 package. There's a good chance that macs2 is already living in your environment if you are reading this help page, which can be tested using the following–

```
which macs2
```

and hopefully seeing a valid path. If not, one solution for macs2 install is to create a separate python2 virtual environment using the following commands –

```
python2 -m venv venv2
source venv2/bin/active

pip install numpy
pip install wheel
pip install macs2
```

- R language and package dependencies (see wiki/Rpackages for more information).
- samtools

CHAPTER 4

Trimming

4.1 Optimized adaptor trimming

Compared to most conventional tools, **proatac** offers two key features that optimize the adaptor trimming component of the workflow. First, the trimmers are written in efficient, parallelized C code for optimal performance. Secondly, the adaptor trimming makes no assumptions about the adaptor sequence *a priori* compared to most tools that require inputting the adaptor sequence ahead of time. These features ensure an improved alignment rate and optimized computational time with little user overhead. The current implementation of these features in **proatac** uses compiled linux and mac binary files from the PEAT software to perform adaptor trimming.

4.1.1 Bulk and C1-based single cell trimming.

Uses the naive trimming as described above without regard for the particular sequence.

4.1.2 Droplet-based barcode quantification and annotation

More to come but this is important. Need to split samples based on the particular barcode.

CHAPTER 5

Alignment

CHAPTER 6

Annotating peaks

6.1 Built-in annotation

For several popular genome builds, we've included a variety of useful annotation files to optimize the workflow for **proatac** users. Currently, the following reference genomes are supported–

```
hg19  hg38  mm9  mm10  hg19_mm9_c
```

6.1.1 Mitochondrial Blacklist

Due to the well-documented affinity of the Tn5 transposase for nucleosome-free DNA, a large proportion of reads in ATAC, scATAC, and related assays come from mitochondrial DNA. To mitigate the effect of the Tn5 mitochondrial affinity, we've provided a computational workflow and annotation

For the supported genomes in **proatac**, we've digested the mitochondrial genome into 20 base pair reads and mapped these against the nuclear DNA to determine areas that may be prone to false-positive signal induced by true mitochondrial reads. These blacklisted regions are combined with the ENCODE Project's Blacklist Regions. Putative peaks overlapping with these loci are filtered out automatically as part of the **proatac** workflow. A working repository to reproduce the synthetic mapping of the digested mitochondrial genome to the nuclear genome can be found in the repository here.

6.1.2 TSS Annotation

Genomic loci associated with transcription start sites (TSS) are particularly useful for quantifying the efficacy of the ATAC-Seq protocol both for insert loci distributions as well as a read enrichment. Once again, for the supported genomes, **proatac** automatically considers a set of TSS loci and uses them for quality control and annotation purposes. A simple repository to show where these files where coordinated from and how they were pre-processed is embedded.

6.1.3 Bedtools Genomes

For some operations, having the lengths of the chromsomes of the reference genomes is useful. For the supported genomes in **proatac**, we took these chromosome sizes from the bedtools git page, where these two column files are considered "bedtools genomes."

6.1.4 Missing a reference genome?

If there is a particular reference genome that you'd like to see incorporated into **proatac**, please submit an issue on GitHub or provide a pull-request for each of the files listed above as well as some internal variables (see the **proatac** python class for more information about what is required for a built-in supported genome in this tool).

CHAPTER 7

FAQ

7.1 Single end reads?

In the current implementation, **proatac** only supports paired-end reads. There's some support on Biostars that explains what one should do given single-end sequencing data.

7.2 I found a bug / error; what do I do?

Please let us know if you find any errors/inconsistencies in the documentation or code by filing a new GitHub Issue.

7.3 I ran proatac; now what do I do?

A non-exhaustive list of ideas / resources includes:

- Perform nucleosome calling with NucleoATAC
- Identify variable transcription factors using chromVAR
- Compare peaks called from **proatac** to existing datasets. CistromeDB is a particularly useful resource for this.

CHAPTER 8

Author

The primary developer is Caleb Lareau in the Buenrostro Lab.

CHAPTER 9

Citation

If you use **proatac** in your research, please cite our tool at the following URL:

`http://buenrostrolab.com/proatac`

CHAPTER 10

Bugs / Errors

Please let us know if you find any errors/inconsistencies in the documentation or code by filing a new GitHub Issue.