scallop

Release 1.0

Alex M. Ascensión and Olga Ibáñez Solé

Jul 28, 2022

CONTENTS

Scallop is a method for the quantification of membership single-cells have for their clusters. Membership can be thought of as a measure of transcriptional stability. The greater the membership score of a cell to its cell type cluster, the more robustly the transcriptional signature of its corresponding cell type is expressed by that cell. Check our preprint Lack of evidence for increased transcriptional noise in aged tissues in bioRxiv.

CHAPTER

ONE

HOW TO CITE

Lack of evidence for increased transcriptional noise in aged tissues Olga Ibáñez-Solé, Alex M. Ascensión, Marcos J. Araúzo-Bravo, Ander Izeta bioRxiv 2022.05.18.492432; doi: https://doi.org/10.1101/2022.05.18.492432

CHAPTER

TWO

FAQ

What is the membership score?

The membership score isthe frequency with which the most frequently assigned cluster label was assigned to a cell. That is to say, if a cell has a membership score of 0.7, that means that the cell was assigned to the same cluster in 70% of the bootstrap iterations. The greater the membership score, the more drawn a cell is to its cell type cluster.

What value should I give to the ``n_trials`` parameter?

This parameter defines the number of bootstrap iterations to run. We recommend using `n_trials` > 30. This recommendation is based on our analysis of the convergence of membership scores when gradually increasing the number of bootstrap iterations on five different sc-RNAseq datasets. The output of the analysis is shown in the Supplement 1 to Figure 1 in our preprint.

What value should I give to the ``frac_cells`` parameter?

This parameter defines the fraction of randomly selected cells to use in each bootstrap iteration. We recommend using `frac_cells` > 0.8 to ensure that rare cell types are not entirely excluded from the analysis.

How do you define equivalent clusters across bootstrap iterations?

When iteratively running a clustering algorithm, the labels given to clusters by the clustering algorithm depend on cluster size. With Leiden, the biggest cluster will be named "0", the second biggest will be named "1", and so on. In order to make cluster labels equivalent across iterations, a relabeling step is run within the scallop pipeline. Clusters are relabeled so that the number of cells in common between them is maximized. The relabeling process is explained in more detail in the Scallop subsection within the Methods section of our preprint.

Can I use a clustering algorithm other than Leiden?

Yes. There are several options: Louvain, K-means, DBscan, etc.

2.1 Installation guide

`scallop` can be installed via PyPi:

```
pip install scallop
```

2.1.1 Development version

If you want to install the latest development version you can do it by cloning the repository:

```
git clone https://gitlab.com/olgaibanez/scallop.git
```

Do `cd` there:

```
cd scallop
```

Checkout to `dev` branch:

```
git checkout dev
```

And run pip in "editable mode":

```
pip install -e .
```

2.2 Tutorial

In order to use `scallop` you need a basic knowledge of how `scanpy` and `annData` objects work. `scallop` has been developed to work with the naming convention of `pp`, `tl`, and `pl` from `scanpy`. If you are comfortable with that convention, then `scallop` will be as easy as `scanpy` is, and if you don't know the convention, then this tutorial will help you use both `scallop` and `scanpy`!

2.2.1 Creating a `Scallop` object from an `annData`

In order to work with scallop you need to first run `scanpy` and load a dataset in an `annData` object.

```
import scanpy as sc

adata = sc.read(filename)
```

Now, you have to create the `Scallop` object in which all the information from bootstrapping experiments will be saved.

```
import scallop as sl

scal = sl.Scallop(adata)
```

In this object you can run several bootstrap experiments, which will be saved in a `Bootstrap`. Each `Boostrap` object will contain a unique combination of clustering, number of trials, percentage of cells, etc. chosen. All `Bootstrap` objects will be saved within the `Scallop` object and can be accessed.

2.2.2 Running a bootstrap experiment

In order to run a bootstrap experiment, use the command `sl.tl.getScore()`:

```
sl.tl.getScore(scal, res=1.2, n_trials=30, frac_cells=0.95)
```

In this case, `scallop` will run a bootstrap experiment with `leiden` resolution parameter of 1.2, will execute 30 bootstrapping repetitions, and will select the 95% of cells to do the clustering with them in each repetition.

By default `scallop` stores all results in the `Bootstrap` object. If you need the score, running

```
score = sl.tl.getScore(scal, res=1.2, n_trials=30, frac_cells=0.95, do_return=True)
```

will return a `pd.Series` with the cells from `adata`, and the score for each cell.

If you run `sl.tl.getScore()` with the exact same parametters, `scallop` won't run the boostrap and will return the bootstrap result instead.

You can access all the run bootstraps with

```
scal.getAllBootstraps()
```

An example output

```
------------------------------------------------------------
Bootstrap ID:  0 | res: 1.2 | frac_cells: 0.95 | n_trials: 10
------------------------------------------------------------
Bootstrap ID:  1 | res: 1.0 | frac_cells: 0.95 | n_trials: 20
------------------------------------------------------------
Bootstrap ID:  2 | res: 1.0 | frac_cells: 0.95 | n_trials: 100
```

Running `sl.tl.getScore()` in parallel

Running `leiden` on datasets with many cells, or running the experiment with many trials can slow down the computation performance. You can tweak the argument `n_procs` from `sl.tl.getScore()` to run different trials at the same time.:

```
sl.tl.getScore(scal, res=1.2, n_trials=100, n_procs=8)
```

Note: We recommend using between 4 and 12 processors. A higher number of processors does not always improve substantially the computation time for this case.

Warning: If your dataset has a low number of cells, or `n_trials` is small, we recommend trying `n_procs = 1`. For those cases, the preparation and coordination times for parallelization can be greater than the time saved.

2.3 API

2.3.1 scallop functions

scallop.pl

scallop.tl

scallop.datasets

2.3.2 scallop classes

2.4 Changelog

2.4.1 v1.0

Main components of `scallop`:

- `sl.Scallop` and `sl.Bootstrap` objects.
- tl: `sl.tl.getScore()` .
- pl: `sl.pl.plotScore()`.
- datasets: `sl.datasets.delete_datasets()` and `sl.datasets.joost2016()`.

2.4.2 v1.0.1

Fixed pip dependencies.

2.4.3 v1.0.2

Corrected calculation of `freqScore` to range between 0 and 1.

2.4.4 v1.0.3

Fixed minor issues.

2.4.5 v1.0.4

Corrected scallop clustering `use_weights` parameter inferred from scanpy's.

2.4.6 v1.1.0

- Added logging support.
- Implemented parallelization of `sl.tl.getScore()`.
- Added `louvain` to clustering algorithms.
- Added 10x mouse brain and heart 10k datasets.
- Remapping of unknown clusters.

2.4.7 v1.1.1

Corrected error on sampling for bootstrap part of `sl.tl.getScore()`. When bootstrapping, a set of random indices are selected. If `frac_cells=1` then all indices are selected. However, those indices were randomized, and even with the same `seed` value, clustering would yield different results. Fixing the issue implies to sort the selected indices, regardless of `frac_cells` selected.

2.4.8 v1.2.0

- Implemented `force` option for `sl.tl.getScore()`.
- Applied some assertions on `sl.tl.getScore()`.
- Added dictionary of arguments to add seed to parallel calculation of `bt_matrix`.
- Added `seed_in_bt` as argument for `sl.tl.getScore()`.
- Corrected `seed_in_bt` to solve `None` problem from leiden. In some datasets, if seed is None, the random generation algorithm only considers 6-or-so different seeds.

2.4.9 v1.3.0

- Ensure compatibility with latest Scanpy AnnData structure (v1.8).
- Complete the documentation.
- Ensure compatibility with ray latest version.
- Minor fix in nan management during freqScore calculation.

2.5 License

- Neither the name of the copyright holder nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.