Scanpy Utils

Release 0.1.1

Nikolay Markov

Aug 11, 2021

CONTENTS

CHAPTER

ONE

OVERVIEW

docs	
tests	
package	

Utility functions for scanpy

- Free software: MIT license

1.1 Installation

```
pip install scanpy-utils
```

1.2 Documentation

See https://scanpy-utils.readthedocs.io/en/latest/reference/index.html

1.3 Development

To run all the tests run:

```
tox
```

Note, to combine the coverage data from all the tox environments run:

Windows	`set PYTEST_ADDOPTS=--cov-append` `tox`
Other	`PYTEST_ADDOPTS=--cov-append tox`

CHAPTER

TWO

INSTALLATION

At the command line:

```
pip install scanpy-utils
```

CHAPTER

THREE

REFERENCE

feature_plot(adata, feature[, gridsize, ...])	Plot expression of gene or feature in hexbin
plot_composition(adata, group_by, color[, ...])	Plot composition of clusters or other metadata
get_markers(adata, groupby[, key, ...])	Extract markers from adata into Seurat-like table
merge_gene_info(adata)	Merges gene information from different batches
write_mtx(adata, output_dir)	Save scanpy object in mtx cellranger v3 format.

3.1 sc_utils.feature_plot

sc_utils.**feature_plot**(*adata: anndata._core.anndata.AnnData*, *feature: str*, *gridsize: tuple = (180, 70)*, *linewidths: float = 0.15*, *figsize: Optional[float] = None*) → matplotlib.figure.Figure

Plot expression of gene or feature in hexbin

Plots numeric feature value, commonly gene expression, on UMAP coordinates using hexbin. Feature is taken from `adata.obs` if it is found there, otherwise from `adata.raw`.

Parameters

- **adata** – Annotated data matrix
- **feature** – Name of the feature to plot
- **gridsize** – Tuple of hexbin dimentions, larger numbers produce smaller hexbins
- **linewidths** – Width of the lines to draw around each hexbin
- **figsize** – Optional, make figure of this size

Returns *Matplotlib figure with colorbar added.*

3.2 sc_utils.plot_composition

sc_utils.**plot_composition**(*adata: anndata._core.anndata.AnnData*, *group_by: str*, *color: str*, *relative: bool = False*, *palette: Optional[Collection] = None*, *plot_numbers: bool = False*) → matplotlib.axes._axes.Axes

Plot composition of clusters or other metadata

Groups cells by one metadata field and plots stacked barplot colored by another metadata field. Common use case is to see which samples contribute to which clusters. Plots horizontally.

Parameters

- **adata** – Annotated data matrix

- **group_by** – Name of the field to group by on y axis
- **color** – Name of the field to color by
- **relative** – Plot percentage for each cluster if `True` or absolute counts if `False`
- **palette** – Optional, pass your own palette
- **plot_numbers** – If `True`, plot number of cells next to the bars

Returns *Matplotlib axes with the plot.*

3.3 sc_utils.get_markers

`sc_utils.`**`get_markers`**(*adata*, *groupby*, *key='rank_genes_groups'*, *p_val_cutoff=0.05*, *logfc_cutoff=0.5*)

Extract markers from adata into Seurat-like table

Extracts markers after they are computed by `scanpy`. Produces Seurat-like table with fields `"p_val"`, `"avg_logFC"`, `"pct.1"`, `"pct.2"`, `"p_val_adj"`, `"cluster"`, `"gene"`

Calculates the percentage of cells that express a given gene in the target cluster (`pct.1` field) and outside the cluster (`pct.2` field) from `adata.raw` matrix.

Parameters

- **adata** – Annotated data matrix.
- **groupby** – `adata.obs` field used for marker calculation
- **key** – `adata.uns` key that has computed markers
- **p_val_cutoff** – Drop all genes with adjusted p-value greater than or equal to this
- **logfc_cutoff** – Drop all genes with average logFC less than or equal to this

Returns

- *Returns a pandas dataframe with above listed columns, optionally*
- *subsetted on the genes that pass the cutoffs.*
- `p_val` field is a copy of adjusted p-value field.

Example

```
>>> sc.tl.rank_genes_groups(adata, "leiden", method="wilcoxon", n_genes=200)
>>> markers = sc_utils.get_markers(adata, "leiden")
>>> markers.to_csv("markers.csv")
```

3.4 sc_utils.merge_gene_info

sc_utils.**merge_gene_info**(*adata: anndata._core.anndata.AnnData*)

Merges gene information from different batches

After concatenating several datasets, the gene information dataframe `adata.var` can have a lot of duplicate columns from all the batches.

This function merges `gene_ids`, `feature_types` and `genome` information from batches, inserts them in the table and removes the batch-associated columns.

Parameters **adata** – Annotated data matrix.

Example

```
>>> datasets = [sc.read_h5ad(path) for path in paths]
>>> adata = datasets[0].concatenate(datasets[1:], join="outer")
>>> sc_utils.merge_gene_info(adata)
```

3.5 sc_utils.write_mtx

sc_utils.**write_mtx**(*adata*, *output_dir*)

Save scanpy object in mtx cellranger v3 format.

Saves basic information from adata object as cellranger v3 mtx folder. Saves only `adata.var_names`, `adata.obs_names` and `adata.X` fields. Creates directory `output_dir` if it does not exist. Creates 3 files: `features.tsv.gz`, `barcodes.tsv.gz` and `matrix.mtz.gz`. Will overwrite files in the output directory.

Parameters

- **adata** – Annotated data matrix.
- **output_dir** – Directory where to save results

sc_utils.**expr_colormap**()

Gray-to-blue colormap for expression data

sc_utils.**feature_plot**(*adata: anndata._core.anndata.AnnData*, *feature: str*, *gridsize: tuple = (180, 70)*, *linewidths: float = 0.15*, *figsize: Optional[float] = None*) → matplotlib.figure.Figure

Plot expression of gene or feature in hexbin

Plots numeric feature value, commonly gene expression, on UMAP coordinates using hexbin. Feature is taken from `adata.obs` if it is found there, otherwise from `adata.raw`.

Parameters

- **adata** – Annotated data matrix
- **feature** – Name of the feature to plot
- **gridsize** – Tuple of hexbin dimentions, larger numbers produce smaller hexbins
- **linewidths** – Width of the lines to draw around each hexbin
- **figsize** – Optional, make figure of this size

Returns *Matplotlib figure with colorbar added.*

sc_utils.**get_markers**(*adata*, *groupby*, *key='rank_genes_groups'*, *p_val_cutoff=0.05*, *logfc_cutoff=0.5*)
Extract markers from adata into Seurat-like table

Extracts markers after they are computed by `scanpy`. Produces Seurat-like table with fields `"p_val"`, `"avg_logFC"`, `"pct.1"`, `"pct.2"`, `"p_val_adj"`, `"cluster"`, `"gene"`

Calculates the percentage of cells that express a given gene in the target cluster (`pct.1` field) and outside the cluster (`pct.2` field) from `adata.raw` matrix.

Parameters

- **adata** – Annotated data matrix.
- **groupby** – `adata.obs` field used for marker calculation
- **key** – `adata.uns` key that has computed markers
- **p_val_cutoff** – Drop all genes with adjusted p-value greater than or equal to this
- **logfc_cutoff** – Drop all genes with average logFC less than or equal to this

Returns

- *Returns a pandas dataframe with above listed columns, optionally*
- *subsetted on the genes that pass the cutoffs.*
- `p_val` field is a copy of adjusted p-value field.

Example

```
>>> sc.tl.rank_genes_groups(adata, "leiden", method="wilcoxon", n_genes=200)
>>> markers = sc_utils.get_markers(adata, "leiden")
>>> markers.to_csv("markers.csv")
```

sc_utils.**merge_gene_info**(*adata: anndata._core.anndata.AnnData*)
Merges gene information from different batches

After concatenating several datasets, the gene information dataframe `adata.var` can have a lot of duplicate columns from all the batches.

This function merges `gene_ids`, `feature_types` and `genome` information from batches, inserts them in the table and removes the batch-associated columns.

Parameters **adata** – Annotated data matrix.

Example

```
>>> datasets = [sc.read_h5ad(path) for path in paths]
>>> adata = datasets[0].concatenate(datasets[1:], join="outer")
>>> sc_utils.merge_gene_info(adata)
```

sc_utils.**plot_composition**(*adata: anndata._core.anndata.AnnData*, *group_by: str*, *color: str*, *relative: bool = False*, *palette: Optional[Collection] = None*, *plot_numbers: bool = False*) → matplotlib.axes._axes.Axes
Plot composition of clusters or other metadata

Groups cells by one metadata field and plots stacked barplot colored by another metadata field. Common use case is to see which samples contribute to which clusters. Plots horizontally.

Parameters

- **adata** – Annotated data matrix
- **group_by** – Name of the field to group by on y axis
- **color** – Name of the field to color by
- **relative** – Plot percentage for each cluster if `True` or absolute counts if `False`
- **palette** – Optional, pass your own palette
- **plot_numbers** – If `True`, plot number of cells next to the bars

Returns *Matplotlib axes with the plot.*

sc_utils.**write_mtx**(*adata*, *output_dir*)

Save scanpy object in mtx cellranger v3 format.

Saves basic information from adata object as cellranger v3 mtx folder. Saves only `adata.var_names`, `adata.obs_names` and `adata.X` fields. Creates directory `output_dir` if it does not exist. Creates 3 files: `features.tsv.gz`, `barcodes.tsv.gz` and `matrix.mtz.gz`. Will overwrite files in the output directory.

Parameters

- **adata** – Annotated data matrix.
- **output_dir** – Directory where to save results

CHAPTER

FOUR

CONTRIBUTING

Contributions are welcome, and they are greatly appreciated! Every little bit helps, and credit will always be given.

4.1 Bug reports

When reporting a bug please include:

- Your operating system name and version.
- Any details about your local setup that might be helpful in troubleshooting.
- Detailed steps to reproduce the bug.

4.2 Documentation improvements

Scanpy Utils could always use more documentation, whether as part of the official Scanpy Utils docs, in docstrings, or even on the web in blog posts, articles, and such.

4.3 Feature requests and feedback

The best way to send feedback is to file an issue at https://github.com/NUPulmonary/scanpy-utils/issues.

If you are proposing a feature:

- Explain in detail how it would work.
- Keep the scope as narrow as possible, to make it easier to implement.
- Remember that this is a volunteer-driven project, and that code contributions are welcome :)

4.4 Development

To set up *scanpy-utils* for local development:

1. Fork scanpy-utils (look for the "Fork" button).
2. Clone your fork locally:

```
git clone git@github.com:YOURGITHUBNAME/scanpy-utils.git
```

3. Create a branch for local development:

```
git checkout -b name-of-your-bugfix-or-feature
```

   Now you can make your changes locally.

4. When you're done making changes run all the checks and docs builder with tox one command:

```
tox
```

5. Commit your changes and push your branch to GitHub:

```
git add .
git commit -m "Your detailed description of your changes."
git push origin name-of-your-bugfix-or-feature
```

6. Submit a pull request through the GitHub website.

4.4.1 Pull Request Guidelines

If you need some code review or feedback while you're developing the code just make the pull request.

For merging, you should:

1. Include passing tests (run `tox`)[1].
2. Update documentation when there's new API, functionality etc.
3. Add a note to `CHANGELOG.rst` about the changes.
4. Add yourself to `AUTHORS.rst`.

4.4.2 Tips

To run a subset of tests:

```
tox -e envname -- pytest -k test_myfeature
```

To run all the test environments in *parallel*:

```
tox -p auto
```

[1] If you don't have all the necessary python versions available locally you can rely on Travis - it will run the tests for each change you add in the pull request.
It will be slower though ...

CHAPTER

FIVE

AUTHORS

- Nikolay Markov - https://twitter.com/nsmarkov

CHAPTER

SIX

CHANGELOG

6.1 0.1.1 (2021-08-11)

- Fix `get_markers` function to add `pct.1` and `pct.2` columns even if no markers were found.

6.2 0.1 (2021-06-24) Initial release

- `get_markers` function to extract markers from adata and add `pct.1`, `pct.2` fields
- `write_mtx` function to dump adata in cellranger mtx format
- `clean_metadata` function to merge and remove duplicates from `adata.var` after merging

Plotting:

- `feature_plot` function to plot hexbin feature plot on UMAP
- `plot_composition` function to plot composition of clusters based on other metadata
- `expr_colormap` function with custom colormap for expression

CHAPTER

SEVEN

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

S

INDEX