PyPairs Documentation

Release v3.1.0+7.gaf06b18

R. Fechtner, A. Scialdone

Apr 11, 2019

Contents

CHAPTER 1

PyPairs - A python scRNA-Seq classifier

This is a python-reimplementation of the Pairs algorithm as described by A. Scialdone et. al. (2015). Original Paper available under: <https://doi.org/10.1016/j.ymeth.2015.06.021>

A supervided maschine learning algorithm aiming to classify single cells based on their transcriptomic signal. Initially created to predict cell cycle phase from scRNA-Seq data, this algorithm can be used for various applications.

Build to be fully compatible with Scanpy [Wolf18].

Code available on GitHub.

1.1 Core Dependencies

- Numpy
- Numba
- Pandas
- AnnData

1.2 Authors

- **Antonio Scialdone** - *original algorithm*
- **Ron Fechtner** - *implementation and extension in Python*

1.3 Release notes

1.3.1 Announcement

Note: Please only use *pypairs* >= 3.0.9.

1.3.2 Versions

Version 3.0.1 - 3.0.9, Feb 1, 2019

- Various bug fixes

Version 3.0.0, Jan 18, 2019 - Jan 31, 2019

- Complete restructuring of the package. Now fully compatiple with `scanpy` .
- **Added:**
  - This documentation
  - Default (oscope) dataset & marker pairs [Leng15]
- **Changed:**
  - ***sandbag()* and *cyclone()* now accept `AnnData`,** `DataFrame` and `ndarray`
  - Multiprocessing now completely handled by `numba` .

Version 2.0.1 - 2.0.6, Nov 22, 2018

- Minor bug fixes and improvements.

Version 2.0.0, Aug 14, 2018

- Major restructuring of the package
- Improved parallel processing
- **New features:**
  - *sandbag()* and *cyclone()* can now deal with any number of classes to predict

Version 1.0.1 - 1.0.3, Jul 29, 2018

- Bug fixes and improvements. (Mostly bugs though)
- Added multi-core processing

Version 1.0.0, Mar 4, 2018

- Speed and performance improvements.

Version 0.1, Feb 22, 2018

- Simple python reimplementation of the Pairs algorithm.
- Included sandbag() and cyclone() algorithms

CHAPTER 2

Getting Started

2.1 Installation

This package is hosted at PyPi (https://pypi.org/project/pypairs/) and can be installed on any system running Python3 via pip with:

```
pip install pypairs
```

Alternatively, pypairs can be installed using Conda (most easily obtained via the Miniconda Python distribution:

```
conda install -c bioconda pypairs
```

2.2 Minimal Example

Datasets provide a example scRNA dataset and default marker pairs for cell cycle prediction:

```
from pypairs import pairs, datasets

# Load samples from the oscope scRNA-Seq dataset with known cell cycle
training_data = datasets.leng15(mode='sorted')

# Run sandbag() to identify marker pairs
marker_pairs = pairs.sandbag(training_data, fraction=0.6)

# Load samples from the oscope scRNA-Seq dataset without known cell cycle
testing_data = datasets.leng15(mode='unsorted')

# Run cyclone() score and predict cell cycle classes
result = pairs.cyclone(testing_data, marker_pairs)

# Further downstream analysis
print(result)
```

CHAPTER 3

Documentation

To use PyPairs import the package as i.e. follows:

```
from pypairs import pairs, datasets, settings, utils
```

3.1 Sandbag

This function implements the classification step of the pair-based prediction method described by Scialdone et al. (2015) [Scialdone15].

To illustrate, consider classification of cells into G1 phase. Pairs of marker genes are identified with *sandbag()*, where the expression of the first gene in the training data is greater than the second in G1 phase but less than the second in all other phases.

pairs.sandbag(data[, annotation, âĂę])	Calculate âĂŸmarker pairsâĂŹ from a genecount matrix.

3.1.1 pypairs.pairs.sandbag

pypairs.pairs.**sandbag**(*data*, *annotation=None*, *gene_names=None*, *sample_names=None*, *fraction=0.65*, *filter_genes=None*, *filter_samples=None*)

Calculate âĂŸmarker pairsâĂŹ from a genecount matrix. Cells x Genes.

A Pair of genes *(g1, g2)* is considered a marker for a category if its expression changes from *g1 > g2* in one category to *g1 < g2* in all other categories, for at least a `fraction` of cells in this category.

`data` can be of type `AnnData`, `DataFrame` or `ndarray` and should contain the raw or normalized gene counts of shape `n_obs` * `n_vars`. Rows correspond to cells and columns to genes.

- If data is `AnnData` object, the category for each sample should be in in `data.vars['category']`, gene names in `data.var_names` and sample names in `data.obs_names`.

- If data is `DataFrame` object, gene names can be in `df.columns` or passed via `gene_names` and sample names in `df.index` or passed via `sample_names`. The category for each sample must be passed via `annotation`.
  - `annotation` must be in form of *{âĂŸcategory1âĂŹ: [âĂŸsample_1âĂŹ,âĂŹsample_2âĂŹ,âĂę], âĂę}*. List of samples for indexing can be integer, str or a boolean mask of `len(sample_names)`.
- If data `ndarray`, all information must be passed via `annotation`, `gene_names` and `sample_names` parameters.

Marker pairs are returned as a mapping from category to list of 2-tuple Genes: *{âĂŸcategoryâĂŹ: [(Gene_1,Gene_2), âĂę], âĂę}*

Parameters

data : Union[AnnData, DataFrame, ndarray, Collection[Collection[float]]]
The (annotated) data matrix of shape `n_obs` * `n_vars`. Rows correspond to cells and columns to genes.

annotation : Optional[Mapping[str, Collection[Union[str, int, bool]]]]
Mapping from category to genes. If `data` is not `AnnData`, this is required. List of genes can be index, names or logical mask.

gene_names : Optional[Collection[str]] Names for genes, must be same length as `n_vars`. If `data` is not `AnnData`, this is required.

sample_names : Optional[Collection[str]] Names for samples, must be same length as `n_obs`. If `data` is not `AnnData`, this is required.

fraction : float Fraction of cells per category where marker criteria must be satisfied. Default: 0.65

filter_genes : Optional[Collection[Union[str, int, bool]]] A list of genes to keep. If not `None` all genes not in this list will be removed. List can be index, names or logical mask.

filter_samples : Optional[Collection[Union[str, int, bool]]] A list of samples to keep. If not `None` all samples not in this list will be removed. List can be index, names or logical mask.

Return type `Mapping[str, Collection[Tuple[str, str]]]`

Returns *marker_pairs_dict* – A dict mapping from str to a list of 2-tuple, where the key is the category and the list contains the marker pairs: *{âĂŸCategory_1âĂŹ: [(Gene_1, Gene_2), âĂę], âĂę}*.

Examples

To generate marker pairs for a different fraction than the default (0.65) based on the bundled `oscope`-dataset [Leng15] run:

```
from pypairs import pairs, datasets

adata = datasets.leng15()
marker_pairs = pairs.sandbag(adata, fraction=0.5)
```

3.2 Cyclone

For each cell, *cyclone()* calculates the proportion of all marker pairs where the expression of the first gene is greater than the second in the new data (pairs with the same expression are ignored). A high proportion suggests that the cell is likely to belong to this category, as the expression ranking in the new data is consistent with that in the training data. Proportions are not directly comparable between phases due to the use of different sets of gene pairs for each phase. Instead, proportions are converted into scores that account for the size and precision of the proportion estimate. The same process is repeated for all phases, using the corresponding set of marker pairs in pairs.

pairs.cyclone(data[, marker_pairs, âĂę])	Score samples for each category based on marker pairs.

3.2.1 pypairs.pairs.cyclone

pypairs.pairs.**cyclone**(*data*, *marker_pairs=None*, *gene_names=None*, *sample_names=None*, *iterations=1000*, *min_iter=100*, *min_pairs=50*)

Score samples for each category based on marker pairs.

`data` can be of type `AnnData`, `DataFrame` or `ndarray` and should contain the raw or normalized gene counts of shape `n_obs` * `n_vars`. Rows correspond to cells and columns to genes.

- If a `AnnData` object is passed, the category scores and the final prediction will be added to `data.obs` with key `pypairs_{category}_score` and `pypairs_max_class`.
  - If marker pairs contain only the cell cycle categories G1, S and G2M an additional column `pypairs_cc_prediction` will be added. Where category S is assigned to samples where G1 and G2M score are below 0.5, as described in [Scialdone15].

`marker_pairs`, i.e. output from `sandbag()`, must be a mapping from category to list of 2-tuple Genes: *{âĂŸcategoryâĂŹ: [(Gene_1,Gene_2), âĂę], âĂę}*.

- If no `marker_pairs` are passed the default are used from `default_marker()` based on [Leng15] (marker pairs for cell cycle prediction).

Parameters

data : Union[AnnData, DataFrame, ndarray, Collection[Collection[float]]] The (annotated) data matrix of shape `n_obs` * `n_vars`. Rows correspond to cells and columns to genes.

marker_pairs : Optional[Mapping[str, Collection[Tuple[str, str]]]] A dict mapping from str to a list of 2-tuple, where the key is the category and the list contains the marker pairs: {âĂŸCategory_1âĂŹ: [(Gene_1, Gene_2), âĂę], âĂę}. If not provided default marker pairs are used

gene_names : Optional[Collection[str]] Names for genes, must be same length as `n_vars`.

sample_names : Optional[Collection[str]] Names for samples, must be same length as `n_obs`.

iterations : Optional[int] An integer specifying the number of iterations for random sampling to obtain a cycle score. Default: 1000

min_iter : Optional[int] An integer specifying the minimum number of iterations for score estimation. Default: 100

min_pairs : Optional[int] An integer specifying the minimum number of pairs for cycle estimation. Default: 50

Return type DataFrame

Returns

- A DataFrame with samples as index and categories as columns with scores for each category for each
- *sample and a additional column with the name of the max scoring category for each sample.* –
  - If marker pairs contain only the cell cycle categories G1, S and G2M an additional column pypairs_cc_prediction will be added. Where category S is assigned to samples where G1 and G2M score are below 0.5, as described in [Scialdone15].

Examples

To predict the cell cycle phase of the unsorted cell from the [Leng15] dataset run:

```
import pypairs import pairs, datasets

adata = datasets.leng15('unsorted')
marker_pairs = datasets.default_cc_marker()
scores = pairs.cyclone(adata, marker_pairs)
print(scores)
```

While this method is described for cell cycle phase classification, any biological groupings can be used here. However, for non-cell cycle phase groupings users should manually apply their own score thresholds for assigning cells into specific groups.

3.3 Datasets

datasets.leng15([mode, gene_sub, sample_sub])	Single cell RNA-seq data of human hESCs to evaluate Oscope [Leng15]
datasets.default_cc_marker([dataset])	Cell cycle marker pairs derived from [Leng15] with the default *sandbag()* settings.

3.3.1 pypairs.datasets.leng15

pypairs.datasets.**leng15**(*mode='all'*, *gene_sub=None*, *sample_sub=None*)

Single cell RNA-seq data of human hESCs to evaluate Oscope [Leng15]

Total 213 H1 single cells and 247 H1-Fucci single cells were sequenced. The 213 H1 cells were used to evaluate Oscope in identifying oscillatory genes. The H1-Fucci cells were used to confirm the cell cycle gene cluster identified by Oscope in the H1 hESCs. Normalized expected counts are provided in GSE64016_H1andFUCCI_normalized_EC.csv.gz

GEO-Dataset: https://www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GSE64016

Parameters

mode : Optional[str]

sample selection mode:

- âĂŹallâĂŹ for all samples, default

- âĂŹsortedâĂŹ for all samples with known cell cycle (G2, S or G1)
- âĂŹunsortedâĂŹ for all samples with unknown cell cycle (H1)

gene_sub : Optional[Iterable[int]] Index based array of subsetted genes

sample_sub : Optional[Iterable[int]] Index based array of subsetted samples

Return type AnnData

Returns **adata** (AnnData) – Annotated data matrix containing the normalized gene counts

3.3.2 pypairs.datasets.default_cc_marker

pypairs.datasets.**default_cc_marker**(*dataset='leng15'*)

Cell cycle marker pairs derived from [Leng15] with the default *sandbag()* settings.

For description of the dataset see *leng15()*.

Parameters

dataset : Optional[str] placeholder. only options currently âĂŸleng15âĂŹ. Everything else with raise a NotImplementedError

Return type Mapping[str, Collection[Tuple[str, str]]]

Returns Mapping from category *[âĂŸG1âĂŹ,âĂŹG2MâĂŹ,âĂŹSâĂŹ]* to list of gene âĂIJmarker pairsâĂİ *[(âĂŸGene1âĂŹ, âĂŸGene2âĂŹ), ..]*.

3.4 Quality Assesment

utils.evaluate_prediction(prediction, reference)	Calculates F1 Score, Recall and Precision of a cyclone() prediction.

3.4.1 pypairs.utils.evaluate_prediction

pypairs.utils.**evaluate_prediction**(*prediction*, *reference*)

Calculates F1 Score, Recall and Precision of a cyclone() prediction.

Parameters

prediction : Iterable[str] List of predicted classes.

reference : Iterable[str] List of actual classes

Return type DataFrame

Returns

- A DataFrame with columns âĂIJf1âĂİ, âĂIJrecallâĂİ, âĂIJprecisionâĂİ and âĂIJaverageâĂİ
- *for all categories and a overall average containing the respective score.*

Example

To get the prediction quality for the example usecase of cyclone() run:

```
from pypairs import pairs, datasets, utils, plotting
import numpy as np

adata = datasets.leng15('sorted')
marker_pairs = datasets.default_cc_marker()
scores = pairs.cyclone(adata, marker_pairs)

ref_labels = list(np.repeat("G2M", 76)) + list(np.repeat("S", 80)) + list(np.
↪repeat("G1", 91))

prediction_quality = utils.evaluate_prediction(scores['max_class'], ref_labels)

print(prediction_quality)
```

3.5 Utils

utils.export_marker(marker, fname[, defaultpath])	Export marker pairs to json-File.
utils.load_marker(fname[, defaultpath])	Export marker pairs to json-File.

3.5.1 pypairs.utils.export_marker

pypairs.utils.**export_marker**(*marker*, *fname*, *defaultpath=True*)

Export marker pairs to json-File.

Parameters

marker : Mapping[str, Iterable[Tuple[str, str]]] Marker pairs as from *sandbag()*

fname : str Name of the json-File in the writedir (see settings)

defaultpath : Optional[bool] Use settings.writedir as root. Default: True

3.5.2 pypairs.utils.load_marker

pypairs.utils.**load_marker**(*fname*, *defaultpath=True*)

Export marker pairs to json-File.

Parameters

fname : str Name of the json-File to write to

defaultpath : Optional[bool] Use settings.writedir as root. Default: True

3.6 Settings

The default directories for saving figures and caching files.

settings.figdir	Directory for saving figures (default: './figures/').
settings.cachedir	Directory for cache files (default: './cache/').

The verbosity of logging output, where verbosity levels have the following meaning: 0=âĂŹerrorâĂŹ, 1=âĂŹwarningâĂŹ, 2=âĂŹinfoâĂŹ, 3=âĂŹhintâĂŹ

`settings.verbosity`	Verbosity level (default: 1).

Print versions of packages that might influence numerical results.

log.print_versions()	Versions that might influence the numerical results.

3.6.1 pypairs.log.print_versions

pypairs.log.**print_versions**()
Versions that might influence the numerical results. Matplotlib and Seaborn are excluded from this.

CHAPTER 4

References

Bibliography

[Leng15] Leng *et al.* (2015) *Oscope identifies oscillatory genes in unsynchronized single-cell RNA-seq experiments.*, Nat Methods.

[Scialdone15] Scialdone *et al.* (2015), *Computational assignment of cell-cycle stage from single-cell transcriptome data*, Methods.

[Wolf18] Wolf *et al.* (2018) *SCANPY: large-scale single-cell gene expression data analysis*, Genome Biology.

Python Module Index

p

Index