singlet Documentation

Release 0.1

Fabio Zanini

Aug 27, 2017

Contents

Single cell RNA-Seq analysis.

CHAPTER 1

Requirements

Python 3.4+ is required. Moreover, you will need:

- pyyaml
- numpy
- scipy
- pandas
- scikit-learn
- matplotlib
- seaborn
- bhtsne (for t-SNE dimensionality reduction)

Get those from pip or conda.

CHAPTER 2

Install

TODO: upload on pypi.

For the time being, you can clone the git repo and then call:

```
python3 setup.py install
```

CHAPTER 3

Usage example

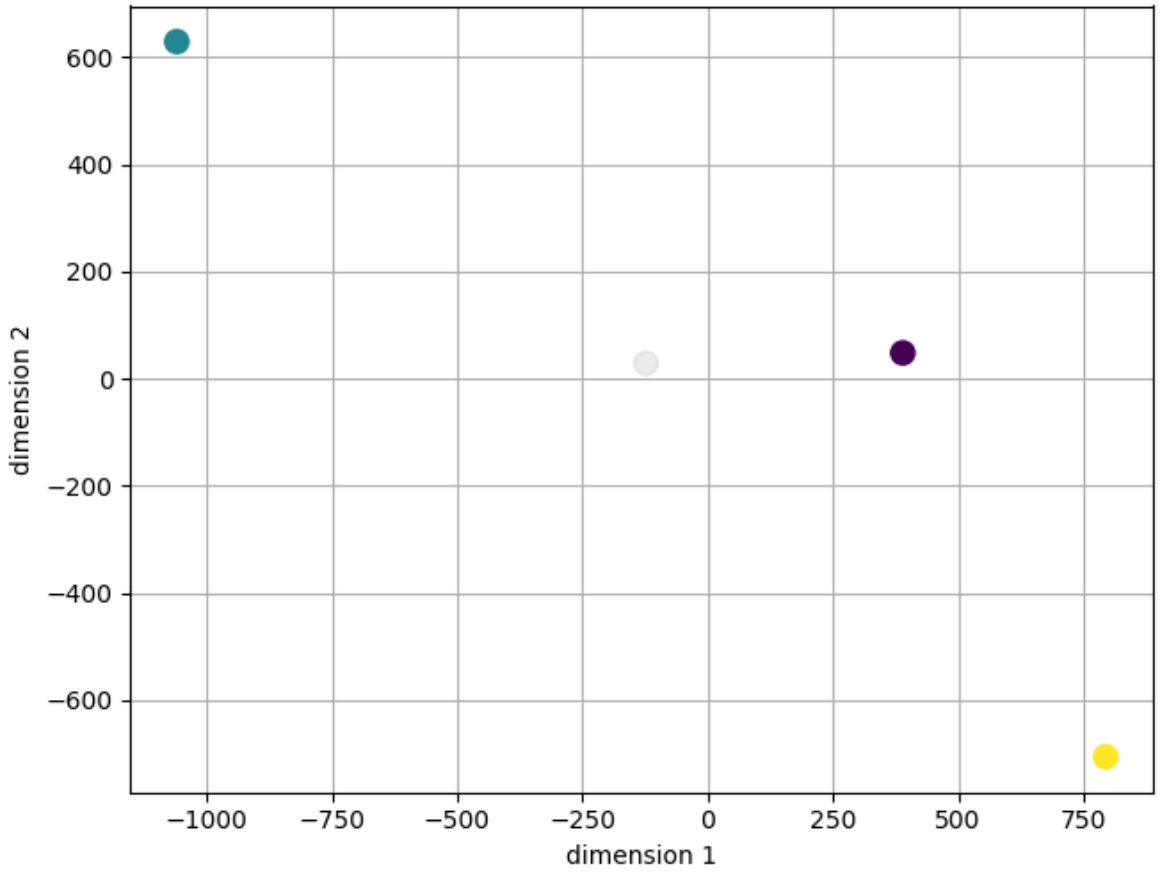
You can have a look inside the *test* folder for examples. To start using the example dataset:

- Set the environment variable *SINGLET_CONFIG_FILENAME* to the location of the example YAML file
- Open a Python/IPython shell and type:

```
from singlet.dataset import Dataset
ds = Dataset(
        samplesheet='example_sheet_tsv',
        counts_table='example_table_tsv')

ds.counts = ds.counts.iloc[:200]
vs = ds.dimensionality.tsne(
        n_dims=2,
        transform='log10',
        theta=0.5,
        perplexity=0.8)
ax = ds.plot.scatter_reduced_samples(
        vs,
        color_by='quantitative_phenotype_1_[A.U.]')
plt.show()
```

This will calculate a t-SNE embedding of the first 200 features and then show your samples in the reduced space. It should look like this:


600
400
200
0
−200
−400
−600
dimension 2
−1000
−750
−500
−250
0
250
500
750
dimension 1

CHAPTER 4

Contents

API reference

Singlet analysis is centered around the *Dataset* class, which describes a set of samples (usually single cells). Each *Dataset* has two main properties:

- a *CountsTable* with the counts of genomic features, typically transcripts
- a *SampleSheet* with the metdata and phenotypic information.

Moreover, a *Dataset* has a number of "action properties" that perform operations on the data:

- *Dataset.correlations*: correlate feature expressions and phenotypes
- *Dataset.dimensionality*: reduce dimensionality of the data including phenotypes
- *Dataset.cluster*: cluster samples, features, and phenotypes
- *Dataset.plot*: plot the results of various analyses

Supporting modules are useful for particular purposes or internal use only:

- *config*
- *utils*
- *io*

singlet.counts_table module

class `singlet.counts_table.`**`CountsTable`**(*data=None*, *index=None*, *columns=None*, *dtype=None*, *copy=False*)

Bases: `pandas.core.frame.DataFrame`

Table of gene expression counts

•Rows are features, e.g. genes.

•Columns are samples.

exclude_features(*spikeins=True*, *other=True*, *inplace=False*, *errors='raise'*)
Get a slice that excludes secondary features.

Parameters

- **spikeins** (*bool*) – Whether to exclude spike-ins
- **other** (*bool*) – Whether to exclude other features, e.g. unmapped reads
- **inplace** (*bool*) – Whether to drop those features in place.
- **errors** (*string*) – Whether to raise an exception if the features to be excluded are already not present.

Returns a slice of self without those features.

Return type *CountsTable*

classmethod from_tablename(*tablename*)
Instantiate a CountsTable from its name in the config file.

Parameters **tablename** (*string*) – name of the counts table in the config file.

Returns the counts table.

Return type *CountsTable*

get_other_features()
Get other features

Returns a slice of self with only other features (e.g. unmapped).

Return type *CountsTable*

get_spikeins()
Get spike-in features

Returns a slice of self with only spike-ins.

Return type *CountsTable*

get_statistics(*metrics=('mean', 'cv')*)
Get statistics of the counts.

Parameters **metrics** (*sequence of strings*) – any of 'mean', 'var', 'std', 'cv', 'fano', 'min', 'max'.

Returns pandas.DataFrame with features as rows and metrics as columns.

normalize(*method='counts_per_million'*, *include_spikeins=False*, *inplace=False*, ***kwargs*)
Normalize counts and return new CountsTable.

Parameters

- **method** (*string or function*) – The method to use for normalization. One of 'counts_per_million', 'counts_per_thousand_spikeins', 'counts_per_thousand_features'. If this argument is a function, it must take the CountsTable as input and return the normalized one as output.
- **include_spikeins** (*bool*) – Whether to include spike-ins in the normalization and result.
- **inplace** (*bool*) – Whether to modify the CountsTable in place or return a new one.

Returns If *inplace* is False, a new, normalized CountsTable.

pseudocount = 0.1

singlet.samplesheet module

class singlet.samplesheet.**SampleSheet**(*data=None*, *index=None*, *columns=None*, *dtype=None*, *copy=False*)

Bases: pandas.core.frame.DataFrame

classmethod from_sheetname(*sheetname*)

singlet.dataset module

class singlet.dataset.**Dataset**(*samplesheet*, *counts_table*)

Bases: object

Collection of cells, with feature counts and metadata

copy()

Copy of the Dataset including a new SampleSheet and CountsTable

counts

Matrix of gene expression counts.

Rows are features, columns are samples.

featurenames

pandas.Index of feature names

metadatanames

pandas.Index of metadata column names

n_features

Number of features

n_samples

Number of samples

query_features(*expression*, *inplace=False*)

Select features based on their expression.

Parameters

- **expression** (*string*) – An expression compatible with pandas.DataFrame.query.
- **inplace** (*bool*) – Whether to change the Dataset in place or return a new one.

Returns If *inplace* is True, None. Else, a Dataset.

query_samples_by_counts(*expression*, *inplace=False*)

Select samples based on gene expression.

Parameters

- **expression** (*string*) – An expression compatible with pandas.DataFrame.query.
- **inplace** (*bool*) – Whether to change the Dataset in place or return a new one.

Returns If *inplace* is True, None. Else, a Dataset.

samplenames

pandas.Index of sample names

samplesheet

Matrix of metadata.

Rows are samples, columns are metadata (e.g. phenotypes).

split (*phenotypes*, *copy=True*)
Split Dataset based on one or more categorical phenotypes

Parameters **phenotypes** (*string or list of strings*) – one or more phenotypes to use for the split. Unique values of combinations of these determine the split Datasets.

Returns the keys are either unique values of the phenotype chosen or, if more than one, tuples of unique combinations.

Return type dict of Datasets

Dataset action properties

singlet.dataset.correlations module

class singlet.dataset.correlations.**Correlation** (*dataset*)
Bases: object

Correlate gene expression and phenotype in single cells

correlate_features_features (*features='all'*, *features2=None*, *method='spearman'*)
Correlate feature expression with one or more phenotypes.

Parameters

- **features** (*list or string*) – list of features to correlate. Use a string for a single feature. The special string 'all' (default) uses all features.
- **features** – list of features to correlate with. Use a string for a single feature. The special string 'all' uses all features. None (default) takes the same list as features, returning a square matrix.
- **method** (*string*) – type of correlation. Must be one of 'pearson' or 'spearman'.

Returns pandas.DataFrame with the correlation coefficients. If either features or features2 is a single string, the function returns a pandas.Series. If both are a string, it returns a single correlation coefficient.

correlate_features_phenotypes (*phenotypes*, *features='all'*, *method='spearman'*, *fillna=None*)
Correlate feature expression with one or more phenotypes.

Parameters

- **phenotypes** (*list of string*) – list of phenotypes, i.e. columns of the samplesheet. Use a string for a single phenotype.
- **features** (*list or string*) – list of features to correlate. Use a string for a single feature. The special string 'all' (default) uses all features.
- **method** (*string*) – type of correlation. Must be one of 'pearson' or 'spearman'.
- **fillna** (*dict, int, or None*) – a dictionary with phenotypes as keys and numbers to fill for NaNs as values. None will do nothing, potentially yielding NaN as correlation coefficients.

Returns pandas.DataFrame with the correlation coefficients. If either phenotypes or features is a single string, the function returns a pandas.Series. If both are a string, it returns a single correlation coefficient.

correlate_phenotypes_phenotypes(*phenotypes*, *phenotypes2=None*, *method='spearman'*, *fillna=None*, *fillna2=None*)

Correlate feature expression with one or more phenotypes.

Parameters

- **phenotypes** (*list of string*) – list of phenotypes, i.e. columns of the samplesheet. Use a string for a single phenotype.
- **phenotypes2** (*list of string*) – list of phenotypes, i.e. columns of the samplesheet. Use a string for a single phenotype. None (default) uses the same as phenotypes.
- **method** (*string*) – type of correlation. Must be one of 'pearson' or 'spearman'.
- **fillna** (*dict, int, or None*) – a dictionary with phenotypes as keys and numbers to fill for NaNs as values. None will do nothing, potentially yielding NaN as correlation coefficients.
- **fillna2** (*dict, int, or None*) – as fillna, but for phenotypes2.

Returns pandas.DataFrame with the correlation coefficients. If either phenotypes or features is a single string, the function returns a pandas.Series. If both are a string, it returns a single correlation coefficient.

singlet.dataset.dimensionality module

class singlet.dataset.dimensionality.**DimensionalityReduction**(*dataset*)

Bases: object

Reduce dimensionality of gene expression and phenotype in single cells

pca(*n_dims=2*, *transform='log10'*, *robust=True*, *random_state=None*)

Principal component analysis

Parameters

- **n_dims** (*int*) – Number of dimensions (2+).
- **transform** (*string or None*) – Whether to preprocess the data.
- **robust** (*bool*) – Whether to use Principal Component Pursuit to exclude outliers.

Returns dict of the left eigenvectors (vs), right eigenvectors (us) of the singular value decomposition, eigenvalues (lambdas), the transform, and the whiten function (for plotting).

tsne(*n_dims=2*, *transform='log10'*, *perplexity=30*, *theta=0.5*, *rand_seed=0*, ***kwargs*)

t-SNE algorithm.

Parameters

- **n_dims** (*int*) – Number of dimensions to use.
- **perplexity** (*float*) – Perplexity of the algorithm.
- **theta** (*float*) – A number between 0 and 1. Higher is faster but less accurate (via the Barnes-Hut approximation).
- **rand_seed** (*int*) – Random seed. -1 randomizes each run.
- ****kwargs** – Named arguments passed to the t-SNE algorithm.

Returns:

singlet.dataset.cluster module

class singlet.dataset.cluster.**Cluster**(*dataset*)
Bases: object

Cluster samples, features, and phenotypes

hierarchical(*axis*, *phenotypes=()*, *metric='correlation'*, *method='average'*, *log_features=True*, *optimal_ordering=False*, ***kwargs*)
Hierarchical clustering.

Parameters

- **axis** (*string*) – It must be 'samples' or 'features'. The Dataset.counts matrix is used and either samples or features are clustered.
- **phenotypes** (*iterable of strings*) – Phenotypes to add to the features for joint clustering.
- **metric** (*string*) – Metric to calculate the distance matrix. Should be a string accepted by scipy.spatial.distance.pdist.
- **method** (*string*) – Clustering method. Must be a string accepted by scipy.cluster.hierarchy.linkage.
- **log_features** (*bool*) – Whether to add pseudocounts and take a log of the feature counts before calculating distances.
- **optimal_ordering** (*bool*) – Whether to resort the linkage so that nearest neighbours have shortest distance. This may take longer than the clustering itself.

Returns dict with the linkage, distance matrix, and ordering.

singlet.dataset.plot module

class singlet.dataset.plot.**Plot**(*dataset*)
Bases: object

Plot gene expression and phenotype in single cells

clustermap(*cluster_samples=False*, *cluster_features=False*, *phenotypes_cluster_samples=()*, *phenotypes_cluster_features=()*, *subtract_mean=False*, *divide_std=False*, *orientation='horizontal'*, *legend=False*, ***kwargs*)
Samples versus features / phenotypes.

Parameters

- **cluster_samples** (*bool or linkage*) – Whether to cluster samples and show the dendrogram. Can be either, False, True, or a linkage from scipy.cluster.hierarchy.linkage.
- **cluster_features** (*bool or linkage*) – Whether to cluster features and show the dendrogram. Can be either, False, True, or a linkage from scipy.cluster.hierarchy.linkage.
- **phenotypes_cluster_samples** (*iterable of strings*) – Phenotypes to add to the features for joint clustering of the samples. If the clustering has been precomputed including phenotypes and the linkage matrix is explicitely set as cluster_samples, the *same* phenotypes must be specified here, in the same order.
- **phenotypes_cluster_features** (*iterable of strings*) – Phenotypes to add to the features for joint clustering of the features and phenotypes. If the clustering

has been precomputed including phenotypes and the linkage matrix is explicitely set as cluster_features, the *same* phenotypes must be specified here, in the same order.

- **orientation** (*string*) – Whether the samples are on the abscissa ('horizontal') or on the ordinate ('vertical').
- **tight_layout** (*bool or dict*) – Whether to call matplotlib.pyplot.tight_layout at the end of the plotting. If it is a dict, pass it unpacked to that function.
- **legend** (*bool or dict*) – If True, call ax.legend(). If a dict, pass as **kwargs to ax.legend.
- ****kwargs** – named arguments passed to the plot function.

Returns A seaborn ClusterGrid instance.

gate_features_from_statistics (*features='mapped'*, *x='mean'*, *y='cv'*, ***kwargs*)

Select features for downstream analysis with a gate.

Usage: Click with the left mouse button to set the vertices of a polygon. Double left-click closes the shape. Right click resets the plot.

Parameters

- **features** (*list or string*) – List of features to plot. The string 'mapped' means everything excluding spikeins and other, 'all' means everything including spikeins and other.
- **x** (*string*) – Statistics to plot on the x axis.
- **y** (*string*) – Statistics to plot on the y axis.
- ****kwargs** – named arguments passed to the plot function.

Returns pd.Index of features within the gate.

plot_coverage (*features='total'*, *kind='cumulative'*, *ax=None*, *tight_layout=True*, *legend=False*, ***kwargs*)

Plot number of reads for each sample

Parameters

- **features** (*list or string*) – Features to sum over. The string 'total' means all features including spikeins and other, 'mapped' means all features excluding spikeins and other, 'spikeins' means only spikeins, and 'other' means only 'other' features.
- **kind** (*string*) – Kind of plot (default: cumulative distribution).
- **ax** (*matplotlib.axes.Axes*) – The axes to plot into. If None (default), a new figure with one axes is created. ax must not strictly be a matplotlib class, but it must have common methods such as 'plot' and 'set'.
- **tight_layout** (*bool or dict*) – Whether to call matplotlib.pyplot.tight_layout at the end of the plotting. If it is a dict, pass it unpacked to that function.
- **legend** (*bool or dict*) – If True, call ax.legend(). If a dict, pass as **kwargs to ax.legend.
- ****kwargs** – named arguments passed to the plot function.

Returns matplotlib.axes.Axes with the axes contaiing the plot.

plot_distributions (*features*, *kind='violin'*, *ax=None*, *tight_layout=True*, *legend=False*, *orientation='vertical'*, *sort=False*, *bottom=0*, *grid=None*, ***kwargs*)

Plot distribution of spike-in controls

Parameters

- **features** (*list or string*) – List of features to plot. If it is the string 'spikeins', plot all spikeins, if the string 'other', plot other features.
- **kind** (*string*) – Kind of plot, one of 'violin' (default), 'box', 'swarm'.
- **ax** (*matplotlib.axes.Axes*) – Axes to plot into. If None (default), create a new figure and axes.
- **tight_layout** (*bool or dict*) – Whether to call matplotlib.pyplot.tight_layout at the end of the plotting. If it is a dict, pass it unpacked to that function.
- **legend** (*bool or dict*) – If True, call ax.legend(). If a dict, pass as **kwargs to ax.legend.
- **orientation** (*string*) – 'horizontal' or 'vertical'.
- **sort** (*bool or string*) – True or 'ascending' sorts the features by median, 'descending' uses the reverse order.
- **bottom** (*float or string*) – The value of zero-count features. If you are using a log axis, you may want to set this to 0.1 or any other small positive number. If a string, it must be 'pseudocount', then the CountsTable.pseudocount will be used.
- **grid** (*bool or None*) – Whether to add a grid to the plot. None defaults to your existing settings.
- ****kwargs** – named arguments passed to the plot function.

Returns The axes with the plot.

Return type matplotlib.axes.Axes

scatter_reduced_samples(*vectors_reduced*, *color_by=None*, *color_log=None*, *cmap='viridis'*, *ax=None*, *tight_layout=True*, *legend=False*, ***kwargs*)

Scatter samples after dimensionality reduction.

Parameters

- **vectors_reduced** (*pandas.Dataframe*) – matrix of coordinates of the samples after dimensionality reduction. Rows are samples, columns (typically 2 or 3) are the component in the low-dimensional embedding.
- **color_by** (*string or None*) – color sample dots by phenotype or expression of a certain feature.
- **color_log** (*bool or None*) – use log of phenotype/expression in the colormap. Default None only logs expression, but not phenotypes.
- **cmap** (*string or matplotlib colormap*) – color map to use for the sample dots.
- **ax** (*matplotlib.axes.Axes*) – The axes to plot into. If None (default), a new figure with one axes is created. ax must not strictly be a matplotlib class, but it must have common methods such as 'plot' and 'set'.
- **tight_layout** (*bool or dict*) – Whether to call matplotlib.pyplot.tight_layout at the end of the plotting. If it is a dict, pass it unpacked to that function.
- **legend** (*bool or dict*) – If True, call ax.legend(). If a dict, pass as **kwargs to ax.legend.
- ****kwargs** – named arguments passed to the plot function.

Returns matplotlib.axes.Axes with the axes containing the plot.

scatter_statistics(*features='mapped'*, *x='mean'*, *y='cv'*, *ax=None*, *tight_layout=True*, *legend=False*, *grid=None*, ***kwargs*)

Scatter plot statistics of features.

Parameters

- **features** (*list or string*) – List of features to plot. The string 'mapped' means everything excluding spikeins and other, 'all' means everything including spikeins and other.
- **x** (*string*) – Statistics to plot on the x axis.
- **y** (*string*) – Statistics to plot on the y axis.
- **ax** (*matplotlib.axes.Axes*) – The axes to plot into. If None (default), a new figure with one axes is created. ax must not strictly be a matplotlib class, but it must have common methods such as 'plot' and 'set'.
- **tight_layout** (*bool or dict*) – Whether to call matplotlib.pyplot.tight_layout at the end of the plotting. If it is a dict, pass it unpacked to that function.
- **legend** (*bool or dict*) – If True, call ax.legend(). If a dict, pass as **kwargs to ax.legend.
- **grid** (*bool or None*) – Whether to add a grid to the plot. None defaults to your existing settings.
- ****kwargs** – named arguments passed to the plot function.

Returns matplotlib.axes.Axes with the axes contaiing the plot.

Examples

- examples/quality_control
- examples/pca
- examples/tsne

CHAPTER 5

Indices and tables

- genindex
- modindex
- search

Python Module Index

S

Index

T