scanpy.tl.diffmap

scanpy.tl.diffmap(*adata*, *n_comps=15*, *neighbors_key=None*, *random_state=0*, *copy=False*)

Diffusion Maps [Coifman05] [Haghverdi15] [Wolf18].

Diffusion maps [Coifman05] has been proposed for visualizing single-cell data by [Haghverdi15]. The tool uses the adapted Gaussian kernel suggested by [Haghverdi16] in the implementation of [Wolf18].

The width ("sigma") of the connectivity kernel is implicitly determined by the number of neighbors used to compute the single-cell graph in `neighbors()`. To reproduce the original implementation using a Gaussian kernel, use `method=='gauss'` in `neighbors()`. To use an exponential kernel, use the default `method=='umap'`. Differences between these options shouldn't usually be dramatic.

Parameters:

adata : `AnnData`

Annotated data matrix.

n_comps : `int` (default: `15`)

The number of dimensions of the representation.

neighbors_key : `Optional` [`str`] (default: `None`)

If not specified, diffmap looks .uns['neighbors'] for neighbors settings and .obsp['connectivities'], .obsp['distances'] for connectivities and distances respectively (default storage places for pp.neighbors). If specified, diffmap looks .uns[neighbors_key] for neighbors settings and .obsp[.uns[neighbors_key]['connectivities_key']], .obsp[.uns[neighbors_key]['distances_key']] for connectivities and distances respectively.

random_state : `Union` [`None` , `int` , `RandomState`] (default: `0`)

A numpy random seed

copy : `bool` (default: `False`)

Return a copy instead of writing to adata.

Returns:

: Depending on `copy`, returns or updates `adata` with the following fields.

`X_diffmap` : **`numpy.ndarray`** (`adata.obsm`)

Diffusion map representation of data, which is the right eigen basis of the transition matrix with eigenvectors as columns.

`diffmap_evals` : **`numpy.ndarray`** (`adata.uns`)

Array of size (number of eigen vectors). Eigenvalues of transition matrix.

Notes

The 0-th column in `adata.obsm["X_diffmap"]` is the steady-state solution, which is non-informative in diffusion maps. Therefore, the first diffusion component is at index 1, e.g. `adata.obsm["X_diffmap"][:,1]`