scanpy.tl.dpt

scanpy.tl.dpt(*adata, n_dcs=10, n_branchings=0, min_group_size=0.01, allow_kendall_tau_shift=True, neighbors_key=None, copy=False*)

Infer progression of cells through geodesic distance along the graph [Haghverdi16] [Wolf19].

Reconstruct the progression of a biological process from snapshot data. `Diffusion Pseudotime` has been introduced by [Haghverdi16] and implemented within Scanpy [Wolf18]. Here, we use a further developed version, which is able to deal with disconnected graphs [Wolf19] and can be run in a `hierarchical` mode by setting the parameter `n_branchings>1`. We recommend, however, to only use **`dpt()`** for computing pseudotime (`n_branchings=0`) and to detect branchings via **`paga()`**. For pseudotime, you need to annotate your data with a root cell. For instance:

```
adata.uns['iroot'] = np.flatnonzero(adata.obs['cell_types'] == 'Stem')[0]
```

This requires to run **`neighbors()`**, first. In order to reproduce the original implementation of DPT, use `method=='gauss'` in this. Using the default `method=='umap'` only leads to minor quantitative differences, though.

New in version 1.1.

`dpt()` also requires to run **`diffmap()`** first. As previously, **`dpt()`** came with a default parameter of `n_dcs=10` but **`diffmap()`** has a default parameter of `n_comps=15`, you need to pass `n_comps=10` in **`diffmap()`** in order to exactly reproduce previous **`dpt()`** results.

Parameters:

adata : **`AnnData`**

Annotated data matrix.

n_dcs : **`int`** (default: `10`)

The number of diffusion components to use.

n_branchings : **`int`** (default: `0`)

Number of branchings to detect.

min_group_size : **`float`** (default: `0.01`)

During recursive splitting of branches (‘dpt groups’) for `n_branchings` > 1, do not consider groups that contain less than `min_group_size` data points. If a float, `min_group_size` refers to a fraction of the total number of data points.

allow_kendall_tau_shift : **`bool`** (default: `True`)

If a very small branch is detected upon splitting, shift away from maximum correlation in Kendall tau criterion of [Haghverdi16] to stabilize the splitting.

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

If not specified, dpt looks .uns[‘neighbors’] for neighbors settings and .obsp[‘connectivities’], .obsp[‘distances’] for connectivities and distances respectively (default storage places for pp.neighbors). If specified, dpt 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.

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

Copy instance before computation and return a copy. Otherwise, perform computation inplace and return `None`.

Return type: **`Optional`**[**`AnnData`**]

Returns:

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

If `n_branchings==0`, no field `dpt_groups` will be written.

`dpt_pseudotime` : **`pandas.Series`** (`adata.obs`, dtype `float`)

Array of dim (number of samples) that stores the pseudotime of each cell, that is, the DPT distance with respect to the root cell.

`dpt_groups` : **`pandas.Series`** (`adata.obs`, dtype `category`)

Array of dim (number of samples) that stores the subgroup id (‘0’, ‘1’, ...) for each cell. The groups typically correspond to ‘progenitor cells’, ‘undecided cells’ or ‘branches’ of a process.

Notes

The tool is similar to the R package `destiny` of [Angerer16].