scanpy.tl.ingest

`scanpy.tl.ingest`*(adata, adata_ref, obs=None, embedding_method=('umap', 'pca'), labeling_method='knn', neighbors_key=None, inplace=True, **kwargs)*

Map labels and embeddings from reference data to new data.

→ tutorial: integrating-data-using-ingest

Integrates embeddings and annotations of an `adata` with a reference dataset `adata_ref` through projecting on a PCA (or alternate model) that has been fitted on the reference data. The function uses a knn classifier for mapping labels and the UMAP package [McInnes18] for mapping the embeddings.

Note

We refer to this *asymmetric* dataset integration as *ingesting* annotations from reference data to new data. This is different from learning a joint representation that integrates both datasets in an unbiased way, as CCA (e.g. in Seurat) or a conditional VAE (e.g. in scVI) would do.

You need to run `neighbors()` on `adata_ref` before passing it.

Parameters:

adata : `AnnData`

The annotated data matrix of shape `n_obs` × `n_vars`. Rows correspond to cells and columns to genes. This is the dataset without labels and embeddings.

adata_ref : `AnnData`

The annotated data matrix of shape `n_obs` × `n_vars`. Rows correspond to cells and columns to genes. Variables (`n_vars` and `var_names`) of `adata_ref` should be the same as in `adata`. This is the dataset with labels and embeddings which need to be mapped to `adata`.

obs : `Union`[`str`, `Iterable`[`str`], `None`] (default: `None`)

Labels' keys in `adata_ref.obs` which need to be mapped to `adata.obs` (inferred for observation of `adata`).

embedding_method : `Union`[`str`, `Iterable`[`str`]] (default: `('umap', 'pca')`)

Embeddings in `adata_ref` which need to be mapped to `adata`. The only supported values are 'umap' and 'pca'.

labeling_method : `str` (default: `'knn'`)

The method to map labels in `adata_ref.obs` to `adata.obs`. The only supported value is 'knn'.

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

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

inplace : `bool` (default: `True`)

Only works if `return_joint=False`. Add labels and embeddings to the passed `adata` (if `True`) or return a copy of `adata` with mapped embeddings and labels.

Returns:

: * if `inplace=False` returns a copy of `adata`

with mapped embeddings and labels in `obsm` and `obs` correspondingly

- if `inplace=True` returns `None` and updates `adata.obsm` and `adata.obs` with mapped embeddings and labels

Example

Call sequence:

```
>>> import scanpy as sc
>>> sc.pp.neighbors(adata_ref)
>>> sc.tl.umap(adata_ref)
>>> sc.tl.ingest(adata, adata_ref, obs='cell_type')
```