SpaOTsc

Release 0.2

Jan 18, 2020

CONTENTS:

CHAPTER

ONE

SPAOTSC API REFERENCE

1.1 The SpaOTsc module

`spaotsc.SpaOTsc.`**`choose_landmarks`**(*pts*, *n*, *dmat=None*, *method='maxmin'*, *assignment='nearest'*)

Choose a set of landmark points from a set of points.

[1] De Silva, Vin, and Gunnar E. Carlsson. "Topological estimation using witness complexes." SPBG 4 (2004): 157-166.

Parameters

- **`pts`** (class:*numpy.ndarray*) – coordinates of points (n_points, nD) needed if dmat not given
- **`n`** (`int`) – number of landmark points to select
- **`dmat`** (class:*numpy.ndarray*) – the distance matrix for the points (n_points, n_points)

Returns the indices of selected points and an assignment matrix to assign original points to landmark points (n_landmarks, n_points)

Return type class:*numpy.ndarray*

`spaotsc.SpaOTsc.`**`compute_mcc`**(*true_labels*, *pred_labels*)

Compute matthew's correlation coefficient.

Parameters

- **`true_labels`** (class:*numpy.ndarray*) – 1D integer array
- **`pred_labels`** (class:*numpy.ndarray*) – 1D integer array

Returns mcc

Return type float

`spaotsc.SpaOTsc.`**`knn_graph`**(*D*, *k*)

Construct a k-nearest-neighbor graph as igraph object.

Parameters

- **`D`** (class:*numpy.ndarray*) – a distance matrix for constructing the knn graph
- **`k`** (`int`) – number of nearest neighbors

Returns a knn graph object

Return type class:*igraph.Graph*

`spaotsc.SpaOTsc.`**`knn_graph_nx`**(*D*, *k*)

Construct a k-nearest-neighbor graph as networkx object.

Parameters

- **D** (class:*numpy.ndarray*) – a distance matrix for constructing the knn graph
- **k** (*int*) – number of nearest neighbors

Returns a knn graph object and a list of edges

Return type class:*networkx.Graph*, list of tuples

`spaotsc.SpaOTsc.`**`phi_exp`**(*x*, *eta*, *nu*, *p*)

The exponential weight kernel. Computes exp(-(x/eta)^(p*nu)).

Parameters

- **x** (float or class:*numpy.ndarray*) – the input value
- **eta** (*float*) – the cutoff for this soft thresholding kernel
- **nu** (*int*) – a possitive integer for the power term, a bigger nu gives sharper threshold boundary
- **p** (*int*) – p=1: emphasize elements lower than cutoff; p=-1: emphasize elements higher than cutoff

Returns the kernel output with same shape of x

Return type same as x

`spaotsc.SpaOTsc.`**`sci`**(*x*, *y*, *W*, *scale=False*)

Computes the spatial correlation index in Eq. (9) of [1].

[1] Chen, Yanguang. "A new methodology of spatial cross-correlation analysis." PloS one 10.5 (2015): e0126158.

Parameters

- **x** (class:*numpy.ndarray*) – the variable's values at the spatial locations
- **y** (class:*numpy.ndarray*) – the other variable's values at the spatial locations
- **W** (class:*numpy.ndarray*) – weight matrix (symmetric) among the locations with W[i,i] = 0
- **scale** (*boolean*) – whether to scale the inputs s.t. (1) sum_{i,j}W_{ij} = 1 and (2) x = (x-mu(x))/sigma(x)

Returns a global spatial cross correlation index

Return type float

class `spaotsc.SpaOTsc.`**`spatial_sc`**(*sc_data=None*, *is_data=None*, *sc_data_bin=None*, *is_data_bin=None*, *is_pos=None*, *is_dmat=None*, *sc_dmat=None*)

An object for connecting and analysis of spatial data and single-cell transcriptomics data.

A minimal example usage: Assume we have (1) a pandas DataFrame for single-cell data `df_sc` with rows being cells and columns being genes (2) a numpy array for distance matrix among spatial locations `is_dmat` (3) a numpy array for dissimilarity between single-cell data and spatial data `cost_matrix` (4) a numpy array for dissimilarity matrix within single-cell data `sc_dmat`

```
>>> import spaotsc
>>> spsc = spaotsc.SpaOTsc.spatial_sc(sc_data=df_sc, is_dmat=is_dmat, sc_dmat=sc_
↪dmat)
>>> spsc.transport_plan(cost_matrix)
>>> spsc.cell_cell_distance(use_landmark=True)
>>> spsc.clustering()
>>> spsc.spatial_signaling_ot(['Wnt5'],['fz'],DSgenes_up=['CycD'],DSgenes_down=[
↪'dpp'])
>>> signal_strengths,_=spsc.infer_signal_range_ml(['Wnt5'],['fz'],['CycD','dpp'],␣
↪effect_ranges=[10,50,100])
>>> intercellular_grn=spsc.spatial_grn_range(['Wnt5','fz','CycD','dpp'])
```

Parameters

- **`sc_data`** (class:*pandas.DataFrame*) – single-cell data of size (n_cells, n_genes)
- **`is_data`** (class:*pandas.DataFrame*) – spatial data of size (n_locations, n_genes)
- **`sc_data_bin`** (class:*pandas.DataFrame*) – binarized single-cell data
- **`is_data_bin`** (class:*pandas.DataFrame*) – binarized spatial data
- **`is_pos`** (class:*numpy.ndarray*) – coordinates of spatial locations (n_locations, n_dimensions)
- **`is_dmat`** (class:*numpy.ndarray*) – distance matrix for spatial locations (n_locations, n_locations)
- **`sc_dmat`** (class:*numpy.ndarray*) – dissimilarity matrix for single-cell data (n_cells, n_cells)

List of instance attributes:

Variables

- **`sc_data`** (class:*pandas.DataFrame*) – single-cell data of size (n_cells, n_genes) `__init__`
- **`is_data`** (class:*pandas.DataFrame*) – spatial data of size (n_locations, n_genes) `__init__`
- **`sc_data_bin`** (class:*pandas.DataFrame*) – binarized single-cell data `__init__`
- **`is_data_bin`** (class:*pandas.DataFrame*) – binarized spatial data `__init__`
- **`is_pos`** (class:*numpy.ndarray*) – coordinates of spatial locations (n_locations, n_dimensions) `__init__`
- **`is_dmat`** (class:*numpy.ndarray*) – distance matrix for spatial locations (n_locations, n_locations) `__init__`
- **`sc_dmat`** (class:*numpy.ndarray*) – dissimilarity matrix for single-cell data (n_cells, n_cells) `__init__`
- **`gamma_mapping`** (class:*numpy.ndarray*) – the mapping matrix between single-cell data and spatial data (n_cells, n_locations) `transport_plan`
- **`sc_dmat_spatial`** (class:*numpy.ndarray*) – the spatial cell-cell distance for single-cell data (n_cells, n_cells) `cell_cell_distance`
- **`clustering_ncluster_org`** (*`int`*) – number of clusters in original clustering of single-cell data `clustering`
- **`clustering_nsubcluster`** (*`list of int`*) – number of cell spatial subclusters within each original cluster `clustering`
- **`clustering_partition_org`** (*`list of numpy integer arrays`*) – the cell indices for each original cluster `clustering`

- **`clustering_partition_inds`** (*`dictionary`*) – the cell indices for the cell spatial subclusters, e.g. the key (0,1) returns the cell indices for the second spatial subcluster within the first original cell cluster. `clustering`
- **`gene_cor_scc`** (class:*pandas.DataFrame*) – the intracellular spearmanr correlation between genes `nonspatial_correlation`
- **`gene_cor_is`** (class:*pandas.DataFrame*) – the intercellular spatial correlation between genes `spatial_correlation`
- **`g_bin_edges`** (*`dictionary`*) – the bin edges for the discretization of gene expressions with gene name string as dictionary key `discretize_expression`

`cell_cell_distance`(*epsilon=0.01*, *rho=inf*, *scaling=True*, *sc_dmat_spatial=None*, *use_landmark=False*, *n_landmark=100*)

Compute spatial distance between single cells using optimal transport.

Generates: *self.sc_dmat_spatial*: (n_cell, n_cell) *numpy.ndarray*

Requires: *self.gamma_mapping*, *self.is_dmat*

Parameters

- **`epsilon`** (*`float, defaults to 0.01`*) – weight for entropy regularization term
- **`rho`** (*`float, defaults to inf`*) – weight for KL divergence penalizing unbalanced transport
- **`scaling`** (*`boolean, defaults to True`*) – whether to scale the cost_matrix (is_dmat) to avoid numerical overflow
- **`sc_dmat_spatial`** (class:*numpy.ndarray*, optional) – the spatial distance matrix for single cells (n_cells, n_cells). If given, simply set the distance matrix without computing.
- **`use_landmark`** (*`boolean, defaults to False`*) – whether to use landmark points for computing transport distance.
- **`n_landmark`** (*`int, defaults to 100`*) – number of landmark points to use if use_landmark

Returns (spatial) cell-cell distance matrix (n_cells, n_cells)

Return type class:*numpy.ndarray*

`clustering`(*genes=None*, *pca_n_components=None*, *res_sc=0.5*, *res_is=0.3*, *min_n=3*)

Clustering and spatial subclustering.

Generates:

self.clustering_nsubcluster: list of int, numbers of subclusters in each cluster obtained in regular clustering of single-cell data

self.clustering_partition_inds: list of cell index arrays for clusters

self.clustering_partition_org: a dictionary for cell index arrays of spatial subclusters. The key (1,0) gives the first subcluster for the second cluster.

Requires:

self.sc_dmat_spatial, *self.sc_data*

Parameters

- **`genes`** (*`list`*) – genes to use when clustering single-cell data. All genes in self.sc_data are used if not specified.

- **`pca_n_components`** (*`int`*) – number of pca components when clustering single-cell data
- **`res_sc`** (*`float, defaults to 0.5`*) – resolution parameter in louvain clustering for single-cell data
- **`res_is`** (*`float, defaults to 0.3`*) – resolution parameter in louvain clustering for spatial subclustering of single-cel data
- **`min_n`** (*`int, defaults to 3`*) – minimum number of members to be considered a cluster

`discretize_expression`(*genes=None*, *p0=1e-15*)

Discretize gene expression using Bayesian blocks.

Generate: *self.g_bin_edges*: a dictionary of block edges with gene names as keys

Requires: *self.sc_data*

Parameters **`p0`** (*`float, defaults to 1E-15`*) – the p0 score in Bayesian blocks. A smaller p0 has lower tolerance of false rate, i.e. resulting in fewer blocks.

`gene_clustering`(*gene_dmat*, *res=3*, *k=5*, *rng_seed=48823*)

Cluster the genes based on their spatial pattern difference.

Parameters

- **`gene_dmat`** (class:*numpy.ndarray*) – the distance matrix for genes (n_gene, n_gene)
- **`res`** (*`float, defaults to 3`*) – resolution parameter used by louvain clustering, higher res gives more clusters
- **`k`** (*`int, defaults to 5`*) – the k for knn graph fed to louvain algorithm
- **`rng_seed`** (*`int`*) – random seed for louvain algorithm to get consistent results

Returns a list of index vectors for the clusters

Return type list of list of int

`gene_gene_distance`(*genes=None*, *epsilon=0.01*, *rho=inf*, *scaling=True*, *sc_dmat_spatial=None*, *use_landmark=False*, *n_landmark=100*)

Compute Wasserstein distance between gene expressions in scRNA-seq data.

Parameters

- **`genes`** (*`list of str`*) – the gene names to compute distance
- **`epsilon`** (*`float, defaults to 0.01`*) – weight for entropy regularization term
- **`rho`** (*`float, defaults to inf`*) – weight for KL divergence penalizing unbalanced transport
- **`scaling`** (*`boolean, defaults to True`*) – whether to scale the cost matrix
- **`sc_dmat_spatial`** (class:*numpy.ndarray*) – spatial distance matrix over the single cells
- **`use_landmark`** (*`boolean, defaults to False`*) – whether to use landmark points to accelarate computation
- **`n_landmark`** (*`int, defaults to 100`*) – number of landmark genes to use

Returns gene-gene distance matrix

Return type class:*numpy.ndarray*

gene_pair_ml_effect_range(*gene_1*, *gene_2*, *background_genes=None*, *cor_cut=None*, *n_top_g=None*, *effect_ranges=None*, *method='Importance'*)

Deriving scores for intercellular gene regulation (how much effect does gene_1 in neiborhood have on gene_2) using random forest.

Requires: *self.sc_dmat_spatial*, *self.sc_data*, *self.gene_cor_scc*

Parameters

- **gene_1** (*str*) – the name of source gene whose expression in the neighborhood will be examined
- **gene_2** (*str*) – the name of target gene whose cellular expression will be used
- **background_genes** (*list of str*) – a name list for gene that are correlated to gene_2
- **cor_cut** (*float*) – the cut_off choosing background genes. used when background_genes is not specified
- **n_top_g** (*int*) – the number of genes with highest correlation to gene_2 to be used as background_genes. used when both background_genes and cor_cut are not specified
- **effect_ranges** (*list of float*) – list of spatial distances to consider
- **method** (*str, defaults to 'Importance'*) – 'Importance': interpret the feature importance as regulation strength; 'Prediction': interpret prediction accuracy in cross-validation as regulation strength.

Returns a (n_distance, 2) array with the first row recording the spatial distances examined and the second row being the effect strength

Return type class:*numpy.ndarray*

gene_pair_pid_effect_range(*gene_1*, *gene_2*, *background_genes=None*, *cor_cut=None*, *n_top_g=None*, *effect_ranges=None*, *p0=1e-15*, *cell_id=None*, *output_individual=False*)

The unique information provided by G1_nb (within various ranges) to G2 considering background genes Gi

Requires: *self.sc_dmat_spatial*, *self.sc_data*, *self.gene_cor_scc*

Parameters

- **gene_1** (*str*) – the name of source gene whose expression in the neighborhood will be examined
- **gene_2** (*str*) – the name of target gene whose cellular expression will be used
- **background_genes** (*list of str*) – a name list for gene that are correlated to gene_2
- **cor_cut** (*float*) – the cut_off choosing background genes. used when background_genes is not specified
- **n_top_g** (*int*) – the number of genes with highest correlation to gene_2 to be used as background_genes. used when both background_genes and cor_cut are not specified
- **effect_ranges** (*list of float*) – list of spatial distances to consider
- **p0** (*float, defaults to 1E-15*) – the p0 score in Bayesian blocks. A smaller p0 has lower tolerance of false rate, i.e. resulting in fewer blocks.
- **output_individual** (*boolean, defaults to False*) – where to output the information computed with each background gene

Returns a (n_distance, 2) array with the first row recording the spatial distances examined and the second row being the effect strength

Return type class:*numpy.ndarray*

`infer_signal_range_ml`(*Lgenes*, *Rgenes*, *Dgenes*, *n_top_g=50*, *effect_ranges=None*, *method='Importance'*, *custom_dmat=None*)

Determine spatial distance for given signaling using random forest.

Requires: *self.sc_dmat_spatial*, *self.sc_data*, *self.gene_cor_scc*

Parameters

- **`Lgenes`** (`list of str`) – name list of ligand genes
- **`Rgenes`** (`list of str`) – name list of receptor genes
- **`Dgenes`** (`list of str`) – name list of downstream genes
- **`n_top_g`** (`int, defaults to 50`) – number of background genes to use when building predictive model.
- **`effect_ranges`** (`list of float`) – the spatial distances to examine
- **`method`** (`str, defaults to 'Importance'`) – the way of interpreting likelihood for each spatial distance
- **`custom_dmat`** (class:*numpy.ndarray*) – a cell-cell distance matrix given by user. *self.sc_dmat_spatial* is used if not given.

Returns (n_distance, 2) array for spatial distances (first row) and effect strengths (second row); and a (n_distance, n_DSgenes) array for the effect strength of each downstream genes.

Return type two class:*numpy.ndarray*

`nonspatial_correlation`(*genes=None*)

Compute gene-gene correlation matrix for pre-screening of genes.

Generates: *self.gene_cor_scc*

Requires: self.sc_data', *self.sc_genes*

Parameters **`genes`** (`list of str`) – list of gene names. If not specified, all genes in self.sc_data are used.

`rank_marker_genes`(*cid*, *genes=None*, *method='ranksum'*, *return_scores=False*)

Rank genes to identify markers for cell clusters.

Parameters

- **`cid`** (class:*numpy.1darray*) – cell indices for the cluster
- **`genes`** (`list`) – candidate genes to examine. If not specified, all genes are used.
- **`method`**(`str, defaults to 'ranksum'`) – method to use. 1. 'roc', using auc-roc score to rank; 2. 'ranksum', using ranksum statistics.
- **`return_scores`** (`boolean, defaults to False`) – whether to return scores instead of sorted gene indices

Returns sorted gene indices (if return_scores==False) or gene scores (if return_scores==True)

Return type class:*numpy.1darray*

`spatial_correlation`(*genes=None*, *effect_range=None*, *kernel='lorentz'*, *kernel_nu=10*)

Computes spatial correlation between genes for pre-screening.

Generates: *self.gene_cor_is* pandas DataFrame

Requires: *self.sc_data*

Parameters

- **`genes`** (*`list of str`*) – list of gene to examine
- **`effect_range`** (*`float`*) – spatial distance
- **`kernel`** (*`str, defaults to 'lorentz'`*) – type of kernels for weight matrix
- **`kernel_nu`** (*`int, defaults to 10`*) – power for weight kernel

`spatial_grn_range`(*genes*, *effect_range=None*, *cor_cut=None*, *n_top_edge=None*, *cor_cut_bg=None*, *n_top_g_bg=None*, *method='pid'*, *p0=1e-15*, *output_individual=False*)

Generate the spatial map for intercellular gene-gene regulatory information flow.

Requires: *self.sc_data*, *self.sc_dmat_spatial*, *self.gene_cor_scc*, *self.gene_cor_is*

Parameters

- **`genes`** (*`list of str`*) – name list of genes to be examined
- **`effect_range`** (*`float`*) – spatial distance for analyzing the intercellular processes
- **`cor_cut`** (*`float`*) – the cutoff for spatial correlation between two genes for further examination (used if n_top_edge not specified)
- **`n_top_edge`** (*`int`*) – the number of gene pairs to examine with highest spatial correlation
- **`cor_cut_bg`** (*`float`*) – the cutoff for intracellular gene correlation to select background genes
- **`n_top_g_bg`** (*`int`*) – the number of genes with highest intracellular gene correlation with the target gene to use as background genes (used if cor_cut_bg not specified)
- **`p0`** (*`float, defaults to 1E-15`*) – the p0 value for Bayesian blocks (lower p0 gives fewer number of bins)
- **`output_individual`** (*`boolean, defaults to False`*) – whether to output the individual values computed with each background gene

Returns a data frame with rows being source genes and columns being target genes

Return type class:*pandas.DataFrame*

`spatial_signaling_ot`(*Lgenes*, *Rgenes*, *Tgenes=[]*, *Rbgenes=[]*, *DSgenes_up=[]*, *DSgenes_down=[]*, *gene_bandwidth={}*, *effect_range=None*, *rho=10.0*, *epsilon=0.2*, *kernel_nu=5*, *use_kernel_ligand=False*, *use_kernel_receptor=False*, *return_weight_only=False*)

Generate cell-cell signaling using optimal transport for a list of ligands and a list of receptors.

Requires: *self.sc_dmat_spatial*, *self.sc_data*

Parameters

- **`Lgenes`** – name list of the ligand gene
- **`Rgenes`** (*`list of str`*) – name list of receptor genes
- **`Tgenes`** (*`list of str, optional`*) – name list of genes for transporters of ligands
- **`Rbgenes`** (*`list of str, optional`*) – name list of genes for proteins bound to receptor for the receptor to work
- **`DSgenes_up`** (*`list of str`*) – name list of up regulated genes by the ligand-receptor

- **DSgenes_down** (*list of str*) – name list of down regulated genes by the ligand-receptor
- **gene_bandwidth** (*dictionary (str to scalar), all outputs default to 1*) – the cutoffs for each gene to be considered expressed
- **effect_range** (*float*) – spatial distance cutoff for the signaling
- **epsilon** (*float, defaults to 0.2*) – weight for entropy regularization term
- **rho** (*float, defaults to inf*) – weight for KL divergence penalizing unbalanced transport
- **kernel_nu** (*float, defaults to 5*) – the power parameter for the exponential kernel, bigger nu means sharper soft cutoff
- **use_kernel_ligand** (*boolean, defaults to False*) – whether use kernel function to rescale ligand expression
- **use_kernel_receptor** (*boolean, defaults to False*) – whether use kernel function to rescale receptor expression
- **return_weight_only** (*boolean, defaults to False*) – whether to only return the weight for source distribution and destination distribution

Returns a scoring matrix for the given signaling genes (cells, cells), (i,j) entry is the score for cell i sending signals to cell j

Return type class:*numpy.ndarray*

spatial_signaling_ot_singleligand(*Lgene*, *Rgene*, *Tgenes=None*, *Rbgene=None*, *DSgenes_up=None*, *DSgenes_down=None*, *effect_range=None*, *rho=10.0*, *epsilon=0.2*)

Generate cell-cell signaling using optimal transport for a single ligand.

Requires: *self.sc_dmat_spatial*, *self.sc_data*

Parameters

- **Lgene** (*str*) – name of the ligand gene
- **Rgene** (*list of str*) – name list of receptor genes
- **Tgenes** (*list of str, optional*) – name list of genes for transporters of ligands
- **Rbgene** (*list of str, optional*) – name list of genes for proteins bound to receptor for the receptor to work
- **DSgenes_up** (*list of str*) – name list of up regulated genes by the ligand-receptor
- **DSgenes_down** (*list of str*) – name list of down regulated genes by the ligand-receptor
- **effect_range** (*float*) – spatial distance cutoff for the signaling
- **epsilon** (*float, defaults to 0.2*) – weight for entropy regularization term
- **rho** (*float, defaults to inf*) – weight for KL divergence penalizing unbalanced transport

Returns a scoring matrix for the given signaling genes (cells, cells), (i,j) entry is the score for cell i sending signals to cell j

Return type class:*numpy.ndarray*

spatial_signaling_scoring(*Lgene*, *Rgene*, *Rbgene=None*, *Tgenes=None*, *DSgenes_up=None*, *DSgenes_down=None*, *effect_range=None*, *kernel='exp'*, *kernel_nu=5*, *gene_eta=None*, *penalty_type='addition'*)

Generate cell-cell signaling using predefined scoring function.

Requires: *self.sc_dmat_spatial*, *self.sc_data*

Parameters

- **Lgene** (`str`) – name of the ligand gene
- **Rgene** (`list of str`) – name list of receptor genes
- **Rbgene** (`list of str, optional`) – name list of genes for proteins bound to receptor for the receptor to work
- **Tgenes** (`list of str, optional`) – name list of genes for transporters of ligands
- **DSgenes_up** (`list of str`) – name list of up regulated genes by the ligand-receptor
- **DSgenes_down** (`list of str`) – name list of down regulated genes by the ligand-receptor
- **effect_range** (`float`) – spatial distance cutoff for the signaling
- **kernel** (`str, defaults to 'exp'`) – weight kernel to use for soft thresholding
- **kernel_nu** (`float, defaults to 5`) – power for weight kernel, a higher power gives a shaper edge
- **gene_eta** (`list of float, defaults to 1s`) – a list of threshold values for the downstream genes
- **penalty_type** (`str, defaults to 'addition'`) – how to penalize inconsistency of downstream genes. 'addition': relaxed penalty; 'multiplication': strict penalty

Returns a scoring matrix for the given signaling genes (cells, cells), (i,j) entry is the score for cell i sending signals to cell j

Return type class:*numpy.ndarray*

transport_plan(*cost_matrix*, *cor_matrix=None*, *alpha=0.1*, *epsilon=1.0*, *rho=100.0*, *G_sc=None*, *G_is=None*, *scaling=False*)

Mapping between single cells and spatial data as transport plan.

Generates: *self.gamma_mapping*: (n_cells, n_locations) *numpy.ndarray*

Parameters

- **cost_matrix** (class:*numpy.ndarray*) – dissimilarity matrix between single-cell data and spatial data (cells, locations)
- **cor_matrix** (class:*numpy.ndarray*, optional) – similarity matrix between single-cell data and spatial data (cells, locations)
- **alpha** (`float, [0,1], defaults to 0.1`) – weight for structured part (Gromov-Wassertein loss term)
- **epsilon** (`float, defaults to 1.0`) – weight for entropy regularization term
- **rho** (`float, defaults to 100.0`) – weight for KL divergence penalizing unbalanced transport
- **G_sc** (class:*numpy.ndarray*) – dissimilarity matrix within single-cell data (cells, cells)
- **G_is** (class:*numpy.ndarray*) – distance matrix within spatial data (locations, locations)

- **`scaling`** (*`boolean, defaults to False`*) – whether scale the cost_matrix to have max=1

Returns a mapping between single-cell data and spatial data (cells, locations)

Return type class:*numpy.ndarray*

`visualize_cells`(*type=1*, *method='umap'*, *perplexity=30.0*, *umap_n_neighbors=5*, *umap_min_dist=0.1*)

Visualization of cells.

Parameters **`type`** (*`int`*) – the type of visualization type=1 dimension reduction with spatial distance, label with original clusters; type=2 dimension reduction with scRNAseq, label with spatial subclusters; type=3 dimension reduction with spatial distance, label with spatial subclusters; type=4 dimension reduction with scRNAseq, label with original clusters.

`visualize_subclusters`(*pts=None*, *k=3*, *cut=None*, *vmin=0.005*, *vmax=0.03333333333333333*, *umap_k=3*, *figsize=(20, 20)*)

Visualize subclusters as a summary and distributions over the original geometry (2D).

Parameters

- **`pts`** (class:*numpy.ndarray*) – the coordinates of original geometry (n_locations, 2)
- **`k`** (*`int`*) – the number nearest neighbors to connect in the subcluster summary plot
- **`vmin`** (*`float`*) – the vmin for colormap of the edges in the summary plot
- **`vmax`** (*`float`*) – the vmax for colormap of the edges in the summary plot
- **`umap_k`** (*`int`*) – the n_neighbors parameter in umap dimension reduction

CHAPTER

TWO

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

INDEX