
scvelo documentation

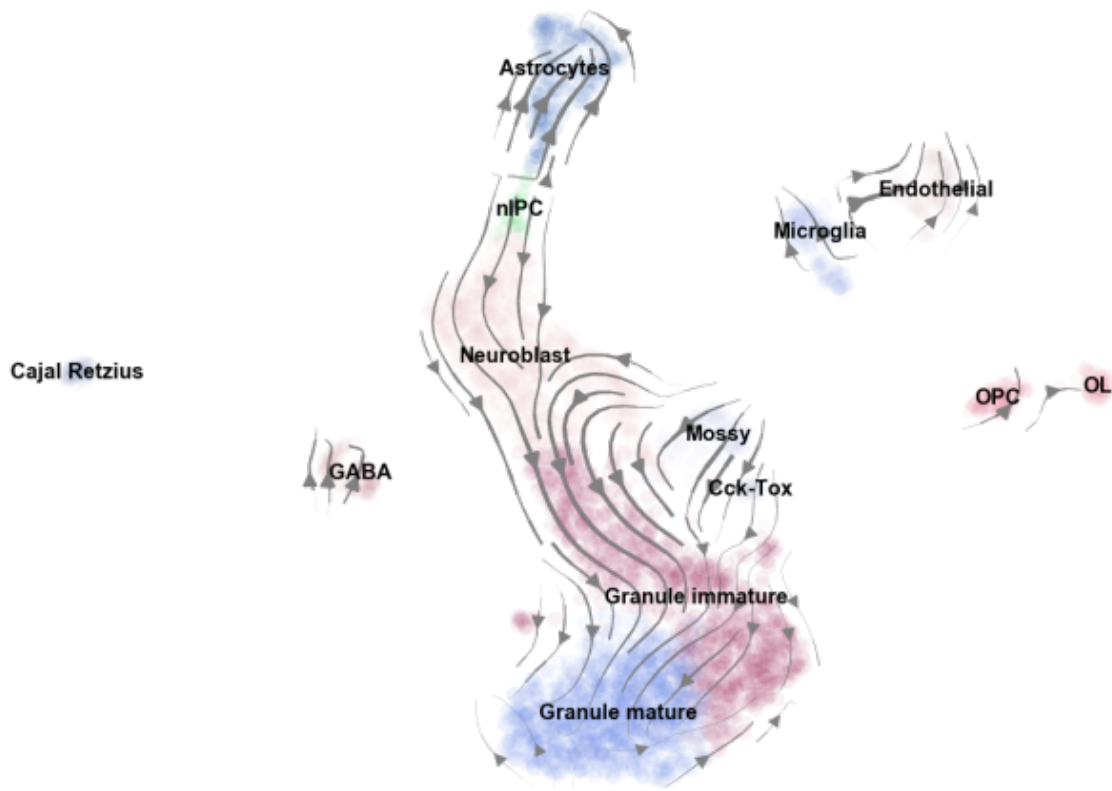
Release 0.1.19

Volker Bergen

Jul 08, 2019

Contents

1	Getting Started	3
2	API	7
3	Release Notes	31
4	References	33
	Bibliography	35
	Python Module Index	37
	Index	39



scVelo is a scalable toolkit for estimating and analyzing RNA velocities in single cells.

RNA velocity, the time derivative of mRNA abundance, enables you to infer directionality in your data by superimposing splicing information. The main principles have been presented in [La Manno et al. \(2018\)](#), and are based on a deterministic steady-state model of transcriptional dynamics. **scVelo** uses a stochastic dynamical model that captures the complete molecular kinetics, thereby recovering the systematic relationship of unspliced to spliced mRNA. That enables you to estimate velocities even in non-stationary populations.

It is compatible with [scanpy \(Wolf et al., 2018\)](#). Making use of sparse implementation, iterative neighbors search and other techniques, it is remarkably efficient in terms of memory and runtime without loss in accuracy and runs easily on your local machine (30k cells in a few minutes).

Report issues and see the code on [GitHub](#).

Welcome to scVelo!

scVelo is a scalable toolkit for estimating and analyzing RNA velocities in single cells.

1.1 Installation

scVelo requires Python 3.6 or later. We recommend to use [Miniconda](#).

Install scVelo from [PyPI](#) using:

```
pip install -U scvelo
```

or from source using:

```
pip install git+https://github.com/theislab/scvelo
```

Parts of scVelo require (optional):

```
conda install -c conda-forge numba pytables louvain
```

The splicing data can be obtained using the [velocityto command line interface](#).

1.2 Basic Usage

Import scVelo as:

```
import scvelo as scv
```

For beautiful visualization you can change the matplotlib settings to our defaults with:

```
scv.settings.set_figure_params('scvelo')
```

1.2.1 Read your data

Read your data file (loom, h5ad, csv, ...) using:

```
adata = scv.read(filename, cache=True)
```

which stores the data matrix (`adata.X`), annotation of cells / observations (`adata.obs`) and genes / variables (`adata.var`), unstructured annotation such as graphs (`adata.uns`) and additional data layers where spliced and unspliced counts are stored (`adata.layers`).

If you already have an existing preprocessed `adata` object you can simply merge the spliced/unspliced counts via:

```
ldata = scv.read(filename.loom, cache=True)
adata = scv.utils.merge(adata, ldata)
```

If you do not have a datasets yet, you can still play around using one of the in-built datasets, e.g.:

```
adata = scv.datasets.dentategyrus()
```

The typical workflow consists of subsequent calls of preprocessing (`scv.pp.*`), analysis tools (`scv.tl.*`) and plotting (`scv.pl.*`).

1.2.2 Basic preprocessing

For velocity estimation basic preprocessing (i.e. gene selection and normalization) is sufficient, e.g. using:

```
scv.pp.filter_and_normalize(adata, **params)
```

For velocity estimation we need the first- and second-order moments (basically means and variances), computed with:

```
scv.pp.moments(adata, **params)
```

1.2.3 Velocity Tools

The core of the software is the efficient and robust estimation of velocities, obtained with:

```
scv.tl.velocity(adata, mode='stochastic', **params)
```

The velocities are vectors in gene expression space obtained by solving a stochastic model of transcriptional dynamics. The solution to the deterministic model is obtained by setting `mode='deterministic'`.

The velocities are stored in `adata.layers` just like the count matrices.

Now we would like to predict cell transitions that are in accordance with the velocity directions. These are computed using cosine correlation (i.e. find potential cell transitions that correlate with the velocity vector) and are stored in a matrix called velocity graph:

```
scv.tl.velocity_graph(adata, **params)
```

Using the graph you can then project the velocities into any embedding (such as UMAP, e.g. obtained with `scanpy`):


```
scv.tl.velocity_embedding(adata, basis='umap', **params)
```

Note, that translation of velocities into a graph is only needed for non-linear embeddings. In PCA space you can skip the velocity graph and directly project into the embedding using `scv.tl.velocity_embedding(adata, basis='pca', direct_projection=True)`.

1.2.4 Visualization

Finally the velocities can be projected and visualized in any embedding (e.g. UMAP) on single cell level, grid level, or as streamplot:

```
scv.pl.velocity_embedding(adata, basis='umap', **params)
scv.pl.velocity_embedding_grid(adata, basis='umap', **params)
scv.pl.velocity_embedding_stream(adata, basis='umap', **params)
```

For every tool module there is a plotting counterpart, which allows you to examine your results in detail, e.g.:

```
scv.pl.velocity(adata, var_names=['gene_A', 'gene_B'], **params)
scv.pl.velocity_graph(adata, **params)
```

scvelo - stochastic single cell RNA velocity

Import scVelo as:

```
import scvelo as scv
```

2.1 Read / Load

<code>read(filename[, backed, sheet, ext, ...])</code>	Read file and return <code>AnnData</code> object.
<code>read_loom(filename[, sparse, cleanup, ...])</code>	Read <code>.loom</code> -formatted hdf5 file.

2.1.1 scvelo.read

`scvelo.read(filename, backed=False, sheet=None, ext=None, delimiter=None, first_column_names=False, backup_url=None, cache=False, **kwargs)`
 Read file and return `AnnData` object.

To speed up reading, consider passing `cache=True`, which creates an hdf5 cache file.

Parameters

filename : *str* If the filename has no file extension, it is interpreted as a key for generating a filename via `sc.settings.writedir + filename + sc.settings.file_format_data`. This is the same behavior as in `sc.read(filename, ...)`.

backed : `{False, True, 'r', 'r+'}`, optional (default: `False`) Load `AnnData` in *backed* mode instead of fully loading it into memory (*memory* mode). Only applies to `.h5ad` files. `True` and `'r'` are equivalent. If you want to modify backed attributes of the `AnnData` object, you need to choose `'r+'`.

sheet : *str*, optional (default: `None`) Name of sheet/table in hdf5 or Excel file.

cache : *bool*, optional (default: `False`) If `False`, read from source, if `True`, read from fast 'h5ad' cache.

ext : *str*, optional (default: *None*) Extension that indicates the file type. If *None*, uses extension of filename.

delimiter : *str*, optional (default: *None*) Delimiter that separates data within text file. If *None*, will split at arbitrary number of white spaces, which is different from enforcing splitting at any single white space ' '.

first_column_names : *bool*, optional (default: *False*) Assume the first column stores row names. This is only necessary if these are not strings: strings in the first column are automatically assumed to be row names.

backup_url : *str*, optional (default: *None*) Retrieve the file from an URL if not present on disk.

Return type AnnData

Returns An AnnData object

2.1.2 scvelo.read_loom

`scvelo.read_loom(filename, sparse=True, cleanup=False, X_name='spliced', obs_names='CellID', var_names='Gene', dtype='float32', **kwargs)`
 Read *.loom*-formatted hdf5 file.

This reads the whole file into memory.

Beware that you have to explicitly state when you want to read the file as sparse data.

Parameters

filename : *PathLike* The filename.

sparse : *bool* Whether to read the data matrix as sparse.

cleanup : *bool* Whether to collapse all obs/var fields that only store one unique value into `.uns['loom-']`.

X_name : *str* Loompy key with which the data matrix *.X* is initialized.

obs_names : *str* Loompy key where the observation/cell names are stored.

var_names : *str* Loompy key where the variable/gene names are stored.

****kwargs** Arguments to loompy.connect

Return type AnnData

2.2 Preprocessing (pp)

<code>pp.filter_genes(data[, min_counts, ...])</code>	Filter genes based on number of cells or counts.
<code>pp.filter_genes_dispersion(data[, flavor, ...])</code>	Extract highly variable genes.
<code>pp.normalize_per_cell(data[, ...])</code>	Normalize each cell by total counts over all genes.
<code>pp.filter_and_normalize(data[, min_counts, ...])</code>	Filtering, normalization and log transform
<code>pp.moments(data[, n_neighbors, n_pcs, mode, ...])</code>	Computes moments for velocity estimation.

2.2.1 scvelo.pp.filter_genes

`scvelo.pp.filter_genes` (*data*, *min_counts=None*, *min_cells=None*, *max_counts=None*,
max_cells=None, *min_counts_u=None*, *min_cells_u=None*,
max_counts_u=None, *max_cells_u=None*, *min_shared_counts=None*,
min_shared_cells=None, *copy=False*)

Filter genes based on number of cells or counts. Keep genes that have at least *min_counts* counts or are expressed in at least *min_cells* cells or have at most *max_counts* counts or are expressed in at most *max_cells* cells. Only provide one of the optional parameters *min_counts*, *min_cells*, *max_counts*, *max_cells* per call.

Parameters

- data** : **AnnData**, **np.ndarray**, **sp.spmatrix** The (annotated) data matrix of shape $n_{obs} \times n_{vars}$. Rows correspond to cells and columns to genes.
- min_counts** : **int**, **optional (default: None)** Minimum number of counts required for a gene to pass filtering.
- min_cells** : **int**, **optional (default: None)** Minimum number of cells expressed required for a gene to pass filtering.
- max_counts** : **int**, **optional (default: None)** Maximum number of counts required for a gene to pass filtering.
- max_cells** : **int**, **optional (default: None)** Maximum number of cells expressed required for a gene to pass filtering.
- min_counts_u** : **int**, **optional (default: None)** Minimum number of unspliced counts required for a gene to pass filtering.
- min_cells_u** : **int**, **optional (default: None)** Minimum number of unspliced cells expressed required for a gene to pass filtering.
- max_counts_u** : **int**, **optional (default: None)** Maximum number of unspliced counts required for a gene to pass filtering.
- max_cells_u** : **int**, **optional (default: None)** Maximum number of unspliced cells expressed required for a gene to pass filtering.
- min_shared_counts** : **int**, **optional (default: None)** Minimum number of counts (in cells expressed simultaneously in unspliced and spliced) required for a gene.
- min_shared_cells** : **int**, **optional (default: None)** Minimum number of cells required for a gene to be expressed simultaneously in unspliced and spliced.
- copy** : **bool**, **optional (default: False)** Determines whether a copy is returned.

Returns Filters the object and adds *n_counts* to *adata.var*.

2.2.2 scvelo.pp.filter_genes_dispersion

`scvelo.pp.filter_genes_dispersion` (*data*, *flavor='seurat'*, *min_disp=None*, *max_disp=None*,
min_mean=None, *max_mean=None*, *n_bins=20*,
n_top_genes=None, *log=True*, *copy=False*)

Extract highly variable genes. The normalized dispersion is obtained by scaling with the mean and standard deviation of the dispersions for genes falling into a given bin for mean expression of genes. This means that for each bin of mean expression, highly variable genes are selected.

Parameters

data : `AnnData`, `np.ndarray`, `sp.sparse` The (annotated) data matrix of shape $n_{obs} \times n_{vars}$. Rows correspond to cells and columns to genes.

flavor : `{'seurat', 'cell_ranger', 'svr'}`, optional (default: `'seurat'`)

Choose the flavor for computing normalized dispersion. If choosing `'seurat'`, this expects non-logarithmized data - the logarithm of mean and dispersion is taken internally when `log` is at its default value `True`. For `'cell_ranger'`, this is usually called for logarithmized data - in this case you should set `log` to `False`. In their default workflows, Seurat passes the cutoffs whereas Cell Ranger passes `n_top_genes`.

max_mean=3, min_disp=0.5, max_disp=None : **min_mean=0.0125**, If `n_top_genes` unequals `None`, these cutoffs for the means and the normalized dispersions are ignored.

n_bins : `int` (default: **20**) Number of bins for binning the mean gene expression. Normalization is done with respect to each bin. If just a single gene falls into a bin, the normalized dispersion is artificially set to 1. You'll be informed about this if you set `settings.verbosity = 4`.

n_top_genes : `int` or `None` (default: `None`) Number of highly-variable genes to keep.

log : `bool`, optional (default: `True`) Use the logarithm of the mean to variance ratio.

copy : `bool`, optional (default: `False`) If an `AnnData` is passed, determines whether a copy is returned.

Returns If an `AnnData` `adata` is passed, returns or updates `adata` depending on `copy`. It filters the `adata` and adds the annotations

2.2.3 scvelo.pp.normalize_per_cell

`scvelo.pp.normalize_per_cell` (`data`, `counts_per_cell_after=None`, `counts_per_cell=None`, `key_n_counts=None`, `max_proportion_per_cell=None`, `use_initial_size=True`, `layers=['spliced', 'unspliced']`, `enforce=False`, `copy=False`)

Normalize each cell by total counts over all genes.

Parameters

data : `AnnData`, `np.ndarray`, `sp.sparse` The (annotated) data matrix of shape $n_{obs} \times n_{vars}$. Rows correspond to cells and columns to genes.

counts_per_cell_after : `float` or `None`, optional (default: `None`) If `None`, after normalization, each cell has a total count equal to the median of the `counts_per_cell` before normalization.

counts_per_cell : `np.array`, optional (default: `None`) Precomputed counts per cell.

key_n_counts : `str`, optional (default: `'n_counts'`) Name of the field in `adata.obs` where the total counts per cell are stored.

max_proportion_per_cell : `int` (default: `None`) Exclude genes counts that account for more than a specific proportion of cell size, e.g. 0.05.

use_initial_size : `bool` (default: `True`) Whether to use initial cell sizes oder actual cell sizes.

layers : `str` or `list` (default: `{'spliced', 'unspliced'}`) Keys for layers to be also considered for normalization.

copy : *bool*, optional (default: *False*) If an `AnnData` is passed, determines whether a copy is returned.

Returns Returns or updates `adata` with normalized version of the original `adata.X`, depending on `copy`.

2.2.4 scvelo.pp.filter_and_normalize

`scvelo.pp.filter_and_normalize` (`data`, `min_counts=None`, `min_counts_u=None`, `min_cells=None`, `min_cells_u=None`, `min_shared_counts=None`, `min_shared_cells=None`, `n_top_genes=None`, `flavor='seurat'`, `log=True`, `copy=False`)

Filtering, normalization and log transform

Expects non-logarithmized data. If using logarithmized data, pass `log=False`.

Runs the following steps

```
scv.pp.filter_genes(adata)
scv.pp.normalize_per_cell(adata)
if n_top_genes is not None:
    scv.pp.filter_genes_dispersion(adata)
if log:
    scv.pp.log1p(adata)
```

Parameters

data : `AnnData` Annotated data matrix.

min_counts : *int* (default: *None*) Minimum number of counts required for a gene to pass filtering (spliced).

min_counts_u : *int* (default: *None*) Minimum number of counts required for a gene to pass filtering (unspliced).

min_cells : *int* (default: *None*) Minimum number of cells expressed required for a gene to pass filtering (spliced).

min_cells_u : *int* (default: *None*) Minimum number of cells expressed required for a gene to pass filtering (unspliced).

min_shared_counts : *int*, optional (default: *None*) Minimum number of counts (in cells expressed simultaneously in unspliced and spliced) required for a gene.

min_shared_cells : *int*, optional (default: *None*) Minimum number of cells required for a gene to be expressed simultaneously in unspliced and spliced.

n_top_genes : *int* (default: *None*) Number of genes to keep.

flavor : {'seurat', 'cell_ranger', 'svr'}, optional (default: 'seurat')
Choose the flavor for computing normalized dispersion. If choosing 'seurat', this expects non-logarithmized data.

log : *bool* (default: *True*) Take logarithm.

copy : *bool* (default: *False*) Return a copy of `adata` instead of updating it.

Returns Returns or updates `adata` depending on `copy`.

2.2.5 scvelo.pp.moments

`scvelo.pp.moments` (*data*, *n_neighbors*=30, *n_pcs*=None, *mode*='connectivities', *method*='umap', *use_rep*=None, *copy*=False)

Computes moments for velocity estimation.

Parameters

- data** : `AnnData` Annotated data matrix.
- n_neighbors** : *int* (default: 30) Number of neighbors to use.
- n_pcs** : *int* (default: None) Number of principal components to use. If not specified, the full space is used of a pre-computed PCA, or 30 components are used when PCA is computed internally.
- mode** : 'connectivities' or 'distances' (default: 'connectivities') Distance metric to use for moment computation.
- method** : {'umap', 'gauss', 'hns', 'sklearn', None} (default: 'umap') Use 'umap' [McInnes18] or 'gauss' (Gauss kernel following [Coifman05] with adaptive width [Haghverdi16]) for computing connectivities.
- use_rep** : None, 'X' or any key for *.obs* (default: None) Use the indicated representation. If None, the representation is chosen automatically: for *.n_vars* < 50, *.X* is used, otherwise 'X_pca' is used.
- copy** : *bool* (default: False) Return a copy instead of writing to *adata*.

Returns

- Returns or updates *adata* with the attributes
- **Ms** (*.layers*) – dense matrix with first order moments of spliced counts.
- **Mu** (*.layers*) – dense matrix with first order moments of unspliced counts.

2.3 Tools (tl)

<code>tl.velocity</code> (<i>data</i> [, <i>vkey</i> , <i>mode</i> , <i>fit_offset</i> , ...])	Estimates velocities in a gene-specific manner
<code>tl.velocity_graph</code> (<i>data</i> [, <i>vkey</i> , <i>xkey</i> , <i>tkey</i> , ...])	Computes velocity graph based on cosine similarities.
<code>tl.velocity_embedding</code> (<i>data</i> [, <i>basis</i> , <i>vkey</i> , ...])	Computes the single cell velocities in the embedding
<code>tl.transition_matrix</code> (<i>adata</i> [, <i>vkey</i> , <i>basis</i> , ...])	Computes transition probabilities from velocity graph
<code>tl.terminal_states</code> (<i>data</i> [, <i>vkey</i> , <i>groupby</i> , ...])	Computes terminal states (root and end points).
<code>tl.rank_velocity_genes</code> (<i>data</i> [, <i>vkey</i> , ...])	Rank genes for velocity characterizing groups.
<code>tl.velocity_confidence</code> (<i>data</i> [, <i>vkey</i> , <i>copy</i>])	Computes confidences of velocities.

2.3.1 scvelo.tl.velocity

`scvelo.tl.velocity` (*data*, *vkey*='velocity', *mode*=None, *fit_offset*=False, *fit_offset2*=False, *filter_genes*=False, *groups*=None, *groupby*=None, *groups_for_fit*=None, *use_raw*=False, *perc*=[5, 95], *copy*=False)

Estimates velocities in a gene-specific manner

Parameters

- data** : `AnnData` Annotated data matrix.
- vkey** : *str* (default: 'velocity') Name under which to refer to the computed velocities for *ve-*

locity_graph and *velocity_embedding*.

mode : *'deterministic', 'stochastic' or 'bayes' (default: 'stochastic')* Whether to run the estimation using the deterministic or stochastic model of transcriptional dynamics. *'bayes'* solves the stochastic model and accounts for heteroscedasticity, but is slower than *'stochastic'*.

fit_offset : *bool (default: False)* Whether to fit with offset for first order moment dynamics.

fit_offset2 : *bool, (default: False)* Whether to fit with offset for second order moment dynamics.

filter_genes : *bool (default: True)* Whether to remove genes that are not used for further velocity analysis.

groups : *str, list (default: None)* Subset of groups, e.g. ['g1', 'g2', 'g3'], to which velocity analysis shall be restricted.

groupby : *str, list or np.ndarray (default: None)* Key of observations grouping to consider.

groups_for_fit : *str, list or np.ndarray (default: None)* Subset of groups, e.g. ['g1', 'g2', 'g3'], to which steady-state fitting shall be restricted.

use_raw : *bool (default: False)* Whether to use raw data for estimation.

perc : *float (default: None)* Percentile, e.g. 98, upon for extreme quantile fit (to better capture steady states for velocity estimation).

copy : *bool (default: False)* Return a copy instead of writing to *adata*.

Returns

- Returns or updates *adata* with the attributes
- **velocity** (*.layers*) – velocity vectors for each individual cell
- **variance_velocity** (*.layers*) – velocity vectors for the cell variances
- **velocity_offset**, **velocity_beta**, **velocity_gamma**, **velocity_r2** (*.var*) – parameters

2.3.2 scvelo.tl.velocity_graph

`scvelo.tl.velocity_graph` (*data*, *vkey='velocity'*, *xkey='Ms'*, *tkey=None*, *basis=None*, *n_neighbors=None*, *n_recurse_neighbors=None*, *random_neighbors_at_max=None*, *sqrt_transform=False*, *approx=False*, *copy=False*)

Computes velocity graph based on cosine similarities.

The cosine similarities are computed between velocities and potential cell state transitions.

Parameters

data : **AnnData** Annotated data matrix.

vkey : *str (default: 'velocity')* Name of velocity estimates to be used.

n_neighbors : *int or None (default: None)* Use fixed number of neighbors or do recursive neighbor search (if *None*).

n_recurse_neighbors : *int (default: 2)* Number of recursions to be done for neighbors search.

random_neighbors_at_max : *int* or *None* (**default: None**) If number of iterative neighbors for an individual cell is higher than this threshold, a random selection of such are chosen as reference neighbors.

sqrt_transform : *bool* (**default: False**) Whether to variance-transform the cell states changes and velocities before computing cosine similarities.

copy : *bool* (**default: False**) Return a copy instead of writing to *adata*.

Returns

- Returns or updates *adata* with the attributes
- **velocity_graph** (*.uns*) – sparse matrix with transition probabilities

2.3.3 scvelo.tl.velocity_embedding

```
scvelo.tl.velocity_embedding(data, basis=None, vkey='velocity', scale=10,
                             self_transitions=True, use_negative_cosines=True,
                             direct_pca_projection=None, retain_scale=False,
                             autoscale=True, all_comps=True, T=None, copy=False)
```

Computes the single cell velocities in the embedding

Parameters

data : *AnnData* Annotated data matrix.

basis : *str* (**default: 'tsne'**) Which embedding to use.

vkey : *str* (**default: 'velocity'**) Name of velocity estimates to be used.

scale : *int* (**default: 10**) Scale parameter of gaussian kernel for transition matrix.

self_transitions : *bool* (**default: True**) Whether to allow self transitions, based on the confidences of transitioning to neighboring cell.

use_negative_cosines : *bool* (**default: True**) Whether to use not only positive, but also negative cosines and use those transitions to the opposite way.

direct_pca_projection : *bool* (**default: None**) Whether to directly project the velocities into PCA space, thus skipping velocity graph.

retain_scale : *bool* (**default: False**) Whether to retain scale from high dimensional space in embedding.

autoscale : *bool* (**default: True**) Whether to scale the embedded velocities by a scalar multiplier, which simply ensures that the arrows in the embedding are properly scaled.

all_comps : *bool* (**default: True**) Whether to compute the velocities on all embedding components or just the first two.

T : *csr_matrix* (**default: None**) Allows the user to directly pass a transition matrix.

Returns

- Returns or updates *adata* with the attributes
- **velocity_basis** (*.obs*) – coordinates of velocity projection on embedding

2.3.4 scvelo.tl.transition_matrix

`scvelo.tl.transition_matrix` (*adata*, *vkey*='velocity', *basis*=None, *backward*=False, *self_transitions*=True, *scale*=10, *perc*=None, *use_negative_cosines*=False, *weight_diffusion*=0, *scale_diffusion*=1, *weight_indirect_neighbors*=None, *n_neighbors*=None, *vgraph*=None)

Computes transition probabilities from velocity graph

Parameters

- adata** : `AnnData` Annotated data matrix.
- vkey** : `str` (default: 'velocity') Name of velocity estimates to be used.
- basis** : `str` or `None` (default: `None`) Restrict transition to embedding if specified
- backward** : `bool` (default: `False`) Whether to use the transition matrix to push forward (`False`) or to pull backward (`True`)
- scale** : `float` (default: `10`) Scale parameter of gaussian kernel.
- weight_diffusion** : `float` (default: `0`) Relative weight to be given to diffusion kernel (Brownian motion)
- scale_diffusion** : `float` (default: `1`) Scale of diffusion kernel.

Returns Returns sparse matrix with transition probabilities.

2.3.5 scvelo.tl.terminal_states

`scvelo.tl.terminal_states` (*data*, *vkey*='velocity', *groupby*=None, *groups*=None, *self_transitions*=False, *basis*=None, *weight_diffusion*=0, *scale_diffusion*=1, *eps*=0.001, *copy*=False)

Computes terminal states (root and end points).

Parameters

- data** : `AnnData` Annotated data matrix.
- vkey** : `str` (default: 'velocity') Name of velocity estimates to be used.
- self_transitions** : `bool` (default: `False`) Allow transitions from one node to itself.
- basis** : `str` (default: `None`) Basis to use.
- weight_diffusion** : `float` (default: `0`) Relative weight to be given to diffusion kernel (Brownian motion)
- scale_diffusion** : `float` (default: `1`) Scale of diffusion kernel.
- eps** : `float` (default: `1e-3`) Tolerance for eigenvalue selection.
- copy** : `bool` (default: `False`) Return a copy instead of writing to data.

Returns

- Returns or updates *data* with the attributes
- **root** (*.obs*) – sparse matrix with transition probabilities.
- **end** (*.obs*) – sparse matrix with transition probabilities.

2.3.6 scvelo.tl.rank_velocity_genes

`scvelo.tl.rank_velocity_genes` (*data*, *vkey*='velocity', *n_genes*=10, *groupby*=None, *match_with*=None, *resolution*=None, *min_counts*=None, *min_r2*=None, *min_dispersion*=None, *copy*=False)

Rank genes for velocity characterizing groups.

Parameters

- data** : `AnnData` Annotated data matrix.
- vkey** : *str* (default: 'velocity') Key of velocities computed in *tl.velocity*
- n_genes** : *int*, optional (default: 100) The number of genes that appear in the returned tables.
- groupby** : *str*, *list* or *np.ndarray* (default: None) Key of observations grouping to consider.
- min_counts** : *float* (default: None) Minimum count of genes for consideration.
- min_r2** : *float* (default: None) Minimum r2 value of genes for consideration.
- min_dispersion** : *float* (default: None) Minimum dispersion norm value of genes for consideration.
- copy** : *bool* (default: False) Return a copy instead of writing to data.

Returns

- Returns or updates *data* with the attributes
- **rank_velocity_genes** (*.uns*) – Structured array to be indexed by group id storing the gene names. Ordered according to scores.
- **velocity_score** (*.var*) – Storing the score for each gene for each group. Ordered according to scores.

2.3.7 scvelo.tl.velocity_confidence

`scvelo.tl.velocity_confidence` (*data*, *vkey*='velocity', *copy*=False)

Computes confidences of velocities.

Parameters

- data** : `AnnData` Annotated data matrix.
- vkey** : *str* (default: 'velocity') Name of velocity estimates to be used.
- copy** : *bool* (default: False) Return a copy instead of writing to *adata*.

Returns

- Returns or updates *adata* with the attributes
- **velocity_length** (*.obs*) – Length of the velocity vectors for each individual cell
- **velocity_confidence** (*.obs*) – Confidence for each cell

2.4 Plotting (pl)

<code>pl.scatter([adata, x, y, basis, vkey, ...])</code>	Scatter plot along observations or variables axes.
<code>pl.velocity(adata[, var_names, basis, ...])</code>	Phase and velocity plot for set of genes.
<code>pl.velocity_graph(adata[, basis, vkey, ...])</code>	Plot of the velocity graph.
<code>pl.velocity_embedding(adata[, basis, vkey, ...])</code>	Scatter plot of velocities on the embedding.
<code>pl.velocity_embedding_grid(adata[, basis, ...])</code>	Scatter plot of velocities on a grid.
<code>pl.velocity_embedding_stream(adata[, basis, ...])</code>	Stream plot of velocities on the embedding.

2.4.1 scvelo.pl.scatter

`scvelo.pl.scatter` (*adata=None, x=None, y=None, basis=None, vkey=None, color=None, use_raw=None, layer=None, color_map=None, colorbar=True, palette=None, size=None, alpha=None, linewidth=None, perc=None, sort_order=True, groups=None, components=None, projection='2d', legend_loc=None, legend_fontsize=None, legend_fontweight=None, right_margin=None, left_margin=None, xlabel=None, ylabel=None, title=None, fontsize=None, figsize=None, xlim=None, ylim=None, show_density=None, show_assignments=None, show_linear_fit=None, dpi=None, frameon=None, show=True, save=None, ax=None, zorder=None, ncols=None, **kwargs*)

Scatter plot along observations or variables axes.

Parameters

- adata** : `AnnData` Annotated data matrix.
- x** : `str, np.ndarray` or `None` (default: `None`) x coordinate
- y** : `str, np.ndarray` or `None` (default: `None`) y coordinate
- basis** : `str` (default='umap') Key for embedding.
- color** : `str, list of str` or `None` (default: `None`) Key for annotations of observations/cells or variables/genes
- use_raw** : `bool` (default: `None`) Use `raw` attribute of `adata` if present.
- layer** : `str, list of str` or `None` (default: `None`) Specify the layer for `color`.
- color_map** : `str` (default: `matplotlib.rcParams['image.cmap']`) String denoting matplotlib color map.
- colorbar** : `bool` (default: `False`) Whether to show colorbar.
- palette** : `list of str` (default: `None`) Colors to use for plotting groups (categorical annotation).
- size** : `float` (default: `5`) Point size.
- alpha** : `float` (default: `1`) Set blending - 0 transparent to 1 opaque.
- linewidth** : `float` (default: `1`) Scaling factor for the width of occurring lines.
- perc** : `tuple, e.g. [2,98]` (default: `None`) Specify percentile for continuous coloring.
- sort_order** : `bool` (default: `True`) For continuous annotations used as color parameter, plot data points with higher values on top of others.
- groups** : `str` (default: `all groups`) Restrict to a few categories in categorical observation annotation.

components : *str* or list of *str* (default: '1,2') For instance, ['1,2', '2,3'].

projection : {'2d', '3d'} (default: '2d') Projection of plot.

legend_loc : *str* (default: 'none') Location of legend, either 'on data', 'right margin' or valid keywords for matplotlib.legend.

legend_fontsize : *int* (default: None) Legend font size.

legend_fontweight : {'normal', 'bold', ...} (default: None) Legend font weight. Defaults to 'bold' if *legend_loc* = 'on data', otherwise to 'normal'. Available are ['light', 'normal', 'medium', 'semibold', 'bold', 'heavy', 'black'].

right_margin : *float* or list of *float* (default: None) Adjust the width of the space right of each plotting panel.

left_margin : *float* or list of *float* (default: None) Adjust the width of the space left of each plotting panel.

xlabel : *str* (default: None) Label of x-axis.

ylabel : *str* (default: None) Label of y-axis.

title : *str* (default: None) Provide title for panels either as, e.g. ["title1", "title2", ...].

fontsize : *float* (default: None) Label font size.

figsize : tuple (default: (7,5)) Figure size.

dpi : *int* (default: 80) Figure dpi.

frameon : *bool* (default: True) Draw a frame around the scatter plot.

ncols : *int* (default: None) Number of panels per row.

show : *bool*, optional (default: None) Show the plot, do not return axis.

save : *bool* or *str*, optional (default: None) If True or a *str*, save the figure. A string is appended to the default filename. Infer the filetype if ending on {'.pdf', '.png', '.svg'}.

ax : *matplotlib.Axes*, optional (default: None) A matplotlib axes object. Only works if plotting a single component.

Returns If *show==False* a *matplotlib.Axis*

2.4.2 scvelo.pl.velocity

`scvelo.pl.velocity` (*adata*, *var_names=None*, *basis=None*, *groupby=None*, *groups=None*, *mode=None*, *fits='all'*, *layers='all'*, *color=None*, *color_map='RdBu_r'*, *colorbar=False*, *perc=[2, 98]*, *use_raw=False*, *size=None*, *alpha=0.5*, *fontsize=None*, *figsize=None*, *dpi=None*, *show=True*, *save=None*, *ax=None*, *ncols=None*, ***kwargs*)

Phase and velocity plot for set of genes.

The phase plot shows spliced against unspliced expressions with steady-state fit. Further the embedding is shown colored by velocity and expression.

Parameters

adata : *AnnData* Annotated data matrix.

var_names : *str* or list of *str* (default: None) Which variables to show.

basis : *str* (default: 'umap') Key for embedding coordinates.

mode : *'stochastic'* or *None* (default: *None*) Whether to show show covariability phase portrait.

fits : *str* or list of *str* (default: *'all'*) Which steady-state estimates to show.

layers : *str* or list of *str* (default: *'all'*) Which layers to show.

color : *str*, list of *str* or *None* (default: *None*) Key for annotations of observations/cells or variables/genes

color_map : *str* (default: *matplotlib.rcParams['image.cmap']*) String denoting matplotlib color map.

perc : tuple, e.g. [2,98] (default: *None*) Specify percentile for continuous coloring.

size : *float* (default: 5) Point size.

alpha : *float* (default: 1) Set blending - 0 transparent to 1 opaque.

fontsize : *float* (default: *None*) Label font size.

figsize : tuple (default: (7,5)) Figure size.

dpi : *int* (default: 80) Figure dpi.

show : *bool*, optional (default: *None*) Show the plot, do not return axis.

save : *bool* or *str*, optional (default: *None*) If *True* or a *str*, save the figure. A string is appended to the default filename. Infer the filetype if ending on {*'pdf'*, *'png'*, *'svg'*}.

ax : *matplotlib.Axes*, optional (default: *None*) A matplotlib axes object. Only works if plotting a single component.

2.4.3 scvelo.pl.velocity_graph

```
scvelo.pl.velocity_graph(adata, basis=None, vkey='velocity', which_graph='velocity',
                        n_neighbors=10, alpha=0.8, perc=90, edge_width=0.2,
                        edge_color='grey', color=None, use_raw=None, layer=None,
                        color_map=None, colorbar=True, palette=None, size=None,
                        sort_order=True, groups=None, components=None, projection='2d',
                        legend_loc='on data', legend_fontsize=None, legend_fontweight=None,
                        right_margin=None, left_margin=None, xlabel=None, ylabel=None,
                        title=None, fontsize=None, figsize=None, dpi=None, frameon=None,
                        show=True, save=None, ax=None)
```

Plot of the velocity graph.

Parameters

adata : *AnnData* Annotated data matrix.

vkey : *str* or *None* (default: *None*) Key for annotations of observations/cells or variables/genes.

which_graph : *'velocity'* or *'neighbors'* (default: *'velocity'*) Whether to show transitions from velocity graph or connectivities from neighbors graph.

basis : *str* (default=*'umap'*) Key for embedding.

color : *str*, list of *str* or *None* (default: *None*) Key for annotations of observations/cells or variables/genes

use_raw : *bool* (default: *None*) Use *raw* attribute of *adata* if present.

layer : *str*, list of *str* or *None* (default: *None*) Specify the layer for *color*.

color_map : *str* (default: *matplotlib.rcParams['image.cmap']*) String denoting matplotlib color map.

colorbar : *bool* (default: *False*) Whether to show colorbar.

palette : list of *str* (default: *None*) Colors to use for plotting groups (categorical annotation).

size : *float* (default: *5*) Point size.

alpha : *float* (default: *1*) Set blending - 0 transparent to 1 opaque.

linewidth : *float* (default: *1*) Scaling factor for the width of occurring lines.

perc : tuple, e.g. *[2,98]* (default: *None*) Specify percentile for continuous coloring.

sort_order : *bool* (default: *True*) For continuous annotations used as color parameter, plot data points with higher values on top of others.

groups : *str* (default: *all groups*) Restrict to a few categories in categorical observation annotation.

components : *str* or list of *str* (default: *'1,2'*) For instance, *['1,2', '2,3']*.

projection : { *'2d'*, *'3d'* } (default: *'2d'*) Projection of plot.

legend_loc : *str* (default: *'none'*) Location of legend, either 'on data', 'right margin' or valid keywords for matplotlib.legend.

legend_fontsize : *int* (default: *None*) Legend font size.

legend_fontweight : { *'normal'*, *'bold'*, ... } (default: *None*) Legend font weight. Defaults to 'bold' if *legend_loc* = 'on data', otherwise to 'normal'. Available are [*'light'*, *'normal'*, *'medium'*, *'semibold'*, *'bold'*, *'heavy'*, *'black'*].

right_margin : *float* or list of *float* (default: *None*) Adjust the width of the space right of each plotting panel.

left_margin : *float* or list of *float* (default: *None*) Adjust the width of the space left of each plotting panel.

xlabel : *str* (default: *None*) Label of x-axis.

ylabel : *str* (default: *None*) Label of y-axis.

title : *str* (default: *None*) Provide title for panels either as, e.g. [*"title1"*, *"title2"*, ...].

fontsize : *float* (default: *None*) Label font size.

figsize : tuple (default: *(7,5)*) Figure size.

dpi : *int* (default: *80*) Figure dpi.

frameon : *bool* (default: *True*) Draw a frame around the scatter plot.

ncols : *int* (default: *None*) Number of panels per row.

show : *bool*, optional (default: *None*) Show the plot, do not return axis.

save : *bool* or *str*, optional (default: *None*) If *True* or a *str*, save the figure. A string is appended to the default filename. Infer the filetype if ending on { *'pdf'*, *'png'*, *'svg'* }.

ax : *matplotlib.Axes*, optional (default: *None*) A matplotlib axes object. Only works if plotting a single component.

Returns *matplotlib.Axis* if *show==False*

2.4.4 scvelo.pl.velocity_embedding

```
scvelo.pl.velocity_embedding(adata, basis=None, vkey='velocity', density=None, arrow_size=None, arrow_length=None, scale=None, X=None, V=None, color=None, use_raw=None, layer=None, color_map=None, colorbar=True, palette=None, size=None, alpha=0.2, perc=None, sort_order=True, groups=None, components=None, projection='2d', legend_loc='none', legend_fontsize=None, legend_fontweight=None, right_margin=None, left_margin=None, xlabel=None, ylabel=None, title=None, fontsize=None, figsize=None, dpi=None, frameon=None, show=True, save=None, ax=None, ncols=None, **kwargs)
```

Scatter plot of velocities on the embedding.

Parameters

- adata** : **AnnData** Annotated data matrix.
- x** : *str*, *np.ndarray* or *None* (default: *None*) x coordinate
- y** : *str*, *np.ndarray* or *None* (default: *None*) y coordinate
- vkey** : *str* or *None* (default: *None*) Key for annotations of observations/cells or variables/genes.
- density** : *float* (default: **1**) Amount of velocities to show - 0 none to 1 all
- arrow_size** : *float* or 3-tuple for **headlength**, **headwidth** and **headaxislength** (default: **1**) Size of arrows.
- arrow_length** : *float* (default: **1**) Length of arrows.
- scale** : *float* (default: **1**) Length of velocities in the embedding.
- basis** : *str* (default='umap') Key for embedding.
- color** : *str*, list of *str* or *None* (default: *None*) Key for annotations of observations/cells or variables/genes
- use_raw** : *bool* (default: *None*) Use *raw* attribute of *adata* if present.
- layer** : *str*, list of *str* or *None* (default: *None*) Specify the layer for *color*.
- color_map** : *str* (default: *matplotlib.rcParams['image.cmap']*) String denoting matplotlib color map.
- colorbar** : *bool* (default: *False*) Whether to show colorbar.
- palette** : list of *str* (default: *None*) Colors to use for plotting groups (categorical annotation).
- size** : *float* (default: **5**) Point size.
- alpha** : *float* (default: **1**) Set blending - 0 transparent to 1 opaque.
- linewidth** : *float* (default: **1**) Scaling factor for the width of occurring lines.
- perc** : tuple, e.g. [2,98] (default: *None*) Specify percentile for continuous coloring.
- sort_order** : *bool* (default: *True*) For continuous annotations used as color parameter, plot data points with higher values on top of others.
- groups** : *str* (default: *all groups*) Restrict to a few categories in categorical observation annotation.

components : *str* or list of *str* (default: '1,2') For instance, ['1,2', '2,3'].

projection : {'2d', '3d'} (default: '2d') Projection of plot.

legend_loc : *str* (default: 'none') Location of legend, either 'on data', 'right margin' or valid keywords for matplotlib.legend.

legend_fontsize : *int* (default: None) Legend font size.

legend_fontweight : {'normal', 'bold', ...} (default: None) Legend font weight. Defaults to 'bold' if *legend_loc* = 'on data', otherwise to 'normal'. Available are ['light', 'normal', 'medium', 'semibold', 'bold', 'heavy', 'black'].

right_margin : *float* or list of *float* (default: None) Adjust the width of the space right of each plotting panel.

left_margin : *float* or list of *float* (default: None) Adjust the width of the space left of each plotting panel.

xlabel : *str* (default: None) Label of x-axis.

ylabel : *str* (default: None) Label of y-axis.

title : *str* (default: None) Provide title for panels either as, e.g. ["title1", "title2", ...].

fontsize : *float* (default: None) Label font size.

figsize : *tuple* (default: (7,5)) Figure size.

dpi : *int* (default: 80) Figure dpi.

frameon : *bool* (default: True) Draw a frame around the scatter plot.

ncols : *int* (default: None) Number of panels per row.

show : *bool*, optional (default: None) Show the plot, do not return axis.

save : *bool* or *str*, optional (default: None) If True or a *str*, save the figure. A string is appended to the default filename. Infer the filetype if ending on {'pdf', 'png', 'svg'}.

ax : *matplotlib.Axes*, optional (default: None) A matplotlib axes object. Only works if plotting a single component.

Returns *matplotlib.Axis* if *show==False*

2.4.5 scvelo.pl.velocity_embedding_grid

`scvelo.pl.velocity_embedding_grid`(*adata*, *basis=None*, *vkey='velocity'*, *density=None*, *smooth=None*, *min_mass=None*, *arrow_size=None*, *arrow_length=None*, *arrow_color=None*, *scale=None*, *autoscale=True*, *n_neighbors=None*, *X=None*, *V=None*, *X_grid=None*, *V_grid=None*, *principal_curve=False*, *color=None*, *use_raw=None*, *layer=None*, *color_map=None*, *colorbar=True*, *palette=None*, *size=None*, *alpha=0.2*, *perc=None*, *sort_order=True*, *groups=None*, *components=None*, *projection='2d'*, *legend_loc='none'*, *legend_fontsize=None*, *legend_fontweight=None*, *right_margin=None*, *left_margin=None*, *xlabel=None*, *ylabel=None*, *title=None*, *fontsize=None*, *figsize=None*, *dpi=None*, *frameon=None*, *show=True*, *save=None*, *ax=None*, *ncols=None*, ***kwargs*)

Scatter plot of velocities on a grid.

Parameters

adata : `AnnData` Annotated data matrix.

x : `str`, `np.ndarray` or `None` (**default: `None`**) x coordinate

y : `str`, `np.ndarray` or `None` (**default: `None`**) y coordinate

vkey : `str` or `None` (**default: `None`**) Key for annotations of observations/cells or variables/genes.

density : `float` (**default: `1`**) Amount of velocities to show - 0 none to 1 all

arrow_size : `float` or 3-tuple for **headlength**, **headwidth** and **headaxislength** (**default: `1`**)
Size of arrows.

arrow_length : `float` (**default: `1`**) Length of arrows.

scale : `float` (**default: `1`**) Length of velocities in the embedding.

min_mass : `float` (**default: `0.5`**) Minimum threshold for mass to be shown.

smooth : `float` (**default: `0.5`**) Multiplication factor for scale in Gaussian kernel around grid point.

n_neighbors : `int` (**default: `None`**) Number of neighbors to consider around grid point.

X : `np.ndarray` (**default: `None`**) embedding grid point coordinates

V : `np.ndarray` (**default: `None`**) embedding grid velocity coordinates

basis : `str` (**default=`'umap'`**) Key for embedding.

color : `str`, list of `str` or `None` (**default: `None`**) Key for annotations of observations/cells or variables/genes

use_raw : `bool` (**default: `None`**) Use `raw` attribute of `adata` if present.

layer : `str`, list of `str` or `None` (**default: `None`**) Specify the layer for `color`.

color_map : `str` (**default: `matplotlib.rcParams['image.cmap']`**) String denoting matplotlib color map.

colorbar : `bool` (**default: `False`**) Whether to show colorbar.

palette : list of `str` (**default: `None`**) Colors to use for plotting groups (categorical annotation).

size : `float` (**default: `5`**) Point size.

alpha : `float` (**default: `1`**) Set blending - 0 transparent to 1 opaque.

linewidth : `float` (**default: `1`**) Scaling factor for the width of occurring lines.

perc : tuple, e.g. [2,98] (**default: `None`**) Specify percentile for continuous coloring.

sort_order : `bool` (**default: `True`**) For continuous annotations used as color parameter, plot data points with higher values on top of others.

groups : `str` (**default: `all groups`**) Restrict to a few categories in categorical observation annotation.

components : `str` or list of `str` (**default: `'1,2'`**) For instance, [`'1,2'`, `'2,3'`].

projection : { `'2d'`, `'3d'` } (**default: `'2d'`**) Projection of plot.

legend_loc : `str` (**default: `'none'`**) Location of legend, either `'on data'`, `'right margin'` or valid keywords for `matplotlib.legend`.

legend_fontsize : *int* (default: *None*) Legend font size.

legend_fontweight : {'normal', 'bold', ...} (default: *None*) Legend font weight. Defaults to 'bold' if *legend_loc* = 'on data', otherwise to 'normal'. Available are ['light', 'normal', 'medium', 'semibold', 'bold', 'heavy', 'black'].

right_margin : *float* or list of *float* (default: *None*) Adjust the width of the space right of each plotting panel.

left_margin : *float* or list of *float* (default: *None*) Adjust the width of the space left of each plotting panel.

xlabel : *str* (default: *None*) Label of x-axis.

ylabel : *str* (default: *None*) Label of y-axis.

title : *str* (default: *None*) Provide title for panels either as, e.g. ["title1", "title2", ...].

fontsize : *float* (default: *None*) Label font size.

figsize : *tuple* (default: (7,5)) Figure size.

dpi : *int* (default: 80) Figure dpi.

frameon : *bool* (default: *True*) Draw a frame around the scatter plot.

ncols : *int* (default: *None*) Number of panels per row.

show : *bool*, optional (default: *None*) Show the plot, do not return axis.

save : *bool* or *str*, optional (default: *None*) If *True* or a *str*, save the figure. A string is appended to the default filename. Infer the filetype if ending on {'.pdf', '.png', '.svg'}.

ax : *matplotlib.Axes*, optional (default: *None*) A matplotlib axes object. Only works if plotting a single component.

Returns *matplotlib.Axis* if *show==False*

2.4.6 scvelo.pl.velocity_embedding_stream

`scvelo.pl.velocity_embedding_stream`(*adata*, *basis=None*, *vkey='velocity'*, *density=None*, *smooth=None*, *linewidth=None*, *n_neighbors=None*, *X=None*, *V=None*, *X_grid=None*, *V_grid=None*, *color=None*, *use_raw=None*, *layer=None*, *color_map=None*, *colorbar=True*, *palette=None*, *size=None*, *alpha=0.1*, *perc=None*, *sort_order=True*, *groups=None*, *components=None*, *legend_loc='on data'*, *legend_fontsize=None*, *legend_fontweight=None*, *right_margin=None*, *left_margin=None*, *xlabel=None*, *ylabel=None*, *title=None*, *fontsize=None*, *figsize=None*, *dpi=None*, *frameon=None*, *show=True*, *save=None*, *ax=None*, *ncols=None*, ***kwargs*)

Stream plot of velocities on the embedding.

Parameters

adata : *AnnData* Annotated data matrix.

x : *str*, *np.ndarray* or *None* (default: *None*) x coordinate

y : *str*, *np.ndarray* or *None* (default: *None*) y coordinate

vkey : *str* or *None* (default: *None*) Key for annotations of observations/cells or variables/genes.

density : *float* (default: **1**) Amount of velocities to show - 0 none to 1 all

smooth : *float* (default: **0.5**) Multiplication factor for scale in Gaussian kernel around grid point.

linewidth : *float* (default: **1**) Line width for streamplot.

n_neighbors : *int* (default: **None**) Number of neighbors to consider around grid point.

X : *np.ndarray* (default: **None**) Embedding grid point coordinates

V : *np.ndarray* (default: **None**) Embedding grid velocity coordinates

basis : *str* (default='umap') Key for embedding.

color : *str*, *list of str* or *None* (default: *None*) Key for annotations of observations/cells or variables/genes

use_raw : *bool* (default: *None*) Use *raw* attribute of *adata* if present.

layer : *str*, *list of str* or *None* (default: *None*) Specify the layer for *color*.

color_map : *str* (default: *matplotlib.rcParams['image.cmap']*) String denoting matplotlib color map.

colorbar : *bool* (default: *False*) Whether to show colorbar.

palette : *list of str* (default: *None*) Colors to use for plotting groups (categorical annotation).

size : *float* (default: **5**) Point size.

alpha : *float* (default: **1**) Set blending - 0 transparent to 1 opaque.

linewidth : *float* (default: **1**) Scaling factor for the width of occurring lines.

perc : *tuple*, e.g. **[2,98]** (default: *None*) Specify percentile for continuous coloring.

sort_order : *bool* (default: *True*) For continuous annotations used as color parameter, plot data points with higher values on top of others.

groups : *str* (default: *all groups*) Restrict to a few categories in categorical observation annotation.

components : *str* or *list of str* (default: **'1,2'**) For instance, **['1,2', '2,3']**.

projection : *str* (default: **'2d'**) Projection of plot.

legend_loc : *str* (default: **'none'**) Location of legend, either 'on data', 'right margin' or valid keywords for matplotlib.legend.

legend_fontsize : *int* (default: *None*) Legend font size.

legend_fontweight : *str* (default: **'normal'**) Legend font weight. Defaults to 'bold' if *legend_loc* = 'on data', otherwise to 'normal'. Available are **['light', 'normal', 'medium', 'semibold', 'bold', 'heavy', 'black']**.

right_margin : *float* or *list of float* (default: *None*) Adjust the width of the space right of each plotting panel.

left_margin : *float* or *list of float* (default: *None*) Adjust the width of the space left of each plotting panel.

xlabel : *str* (default: *None*) Label of x-axis.

ylabel : *str* (default: *None*) Label of y-axis.

title : *str* (default: *None*) Provide title for panels either as, e.g. [*“title1”*, *“title2”*, ...].

fontsize : *float* (default: *None*) Label font size.

figsize : *tuple* (default: (7,5)) Figure size.

dpi : *int* (default: 80) Figure dpi.

frameon : *bool* (default: *True*) Draw a frame around the scatter plot.

ncols : *int* (default: *None*) Number of panels per row.

show : *bool*, optional (default: *None*) Show the plot, do not return axis.

save : *bool* or *str*, optional (default: *None*) If *True* or a *str*, save the figure. A string is appended to the default filename. Infer the filetype if ending on {*“.pdf”*, *“.png”*, *“.svg”*}.

ax : *matplotlib.Axes*, optional (default: *None*) A matplotlib axes object. Only works if plotting a single component.

Returns *matplotlib.Axis* if *show==False*

2.5 Datasets

<code>datasets.toy_data(n_obs)</code>	Randomly samples from the Dentate Gyrus dataset.
<code>datasets.dentategyrus([adjusted])</code>	Dentate Gyrus dataset from Hochgerner et al.
<code>datasets.forebrain()</code>	Developing human forebrain.

2.5.1 scvelo.datasets.toy_data

`scvelo.datasets.toy_data(n_obs)`
 Randomly samples from the Dentate Gyrus dataset.

Parameters

n_obs : *int* Size of the sampled dataset

Returns Returns *adata* object

2.5.2 scvelo.datasets.dentategyrus

`scvelo.datasets.dentategyrus(adjusted=True)`
 Dentate Gyrus dataset from Hochgerner et al. (2018).

Dentate gyrus is part of the hippocampus involved in learning, episodic memory formation and spatial coding. It is measured using 10X Genomics Chromium and described in Hochgerner et al. (2018). The data consists of 25,919 genes across 3,396 cells and provides several interesting characteristics.

Returns Returns *adata* object

2.5.3 scvelo.datasets.forebrain

`scvelo.datasets.forebrain()`
 Developing human forebrain. Forebrain tissue of a week 10 embryo, focusing on the glutamatergic neuronal lineage.

Returns Returns *adata* object

2.6 Utils

<code>utils.show_proportions(adata)</code>	Fraction of spliced/unspliced/ambiguous abundances
<code>utils.cleanup(data[, clean, keep, copy])</code>	Deletes attributes not needed.
<code>utils.clean_obs_names(data[, base, ...])</code>	Cleans up the obs_names and identifies sample names.
<code>utils.merge(adata, ldata[, copy])</code>	Merges two annotated data matrices.

2.6.1 scvelo.utils.show_proportions

`scvelo.utils.show_proportions(adata)`
 Fraction of spliced/unspliced/ambiguous abundances

Parameters

adata : `AnnData` Annotated data matrix.

Returns *Prints the fractions of abundances.*

2.6.2 scvelo.utils.cleanup

`scvelo.utils.cleanup(data, clean='layers', keep=None, copy=False)`
 Deletes attributes not needed.

Parameters

data : `AnnData` Annotated data matrix.

clean : *str* or list of *str* (default: *layers*) Which attributes to consider for freeing memory.

keep : *str* or list of *str* (default: **None**) Which attributes to keep.

copy : *bool* (default: *False*) Return a copy instead of writing to adata.

Returns Returns or updates *adata* with selection of attributes kept.

2.6.3 scvelo.utils.clean_obs_names

`scvelo.utils.clean_obs_names(data, base='[AGTCBDHKMNRSVWY]', ID_length=12, copy=False)`

Cleans up the obs_names and identifies sample names. For example an obs_name 'sample1_AGTCdate' is changed to 'AGTC' of the sample 'sample1_date'. The sample name is then saved in obs['sample_batch']. The genetic codes are identified according to <https://www.neb.com/tools-and-resources/usage-guidelines/the-genetic-code>.

Parameters

adata : `AnnData` Annotated data matrix.

base : *str* (default: `[AGTCBDHKMNRSVWY]`) Genetic code letters to be identified.

ID_length : *int* (default: **12**) Length of the Genetic Codes in the samples.

copy : *bool* (default: *False*) Return a copy instead of writing to adata.

Returns

- Returns or updates *adata* with the attributes
- **obs_names** (*list*) – updated names of the observations
- **sample_batch** (*.obs*) – names of the identified sample batches

2.6.4 scvelo.utils.merge

`scvelo.utils.merge(adata, ldata, copy=True)`

Merges two annotated data matrices.

Parameters

adata : **AnnData** Annotated data matrix (reference data set).

ldata : **AnnData** Annotated data matrix (to be merged into adata).

Returns Returns a **AnnData** object

2.7 Settings

`settings.set_figure_params([style, figsize, Set resolution/size, styling and format of figures. ...])`

2.7.1 scvelo.settings.set_figure_params

`scvelo.settings.set_figure_params(style='scvelo', figsize=None, dpi=None, dpi_save=None, frameon=None, vector_friendly=True, color_map=None, format='pdf', transparent=False, ipython_format='png2x')`

Set resolution/size, styling and format of figures.

Parameters

style : **str (default: None)** Init default values for `matplotlib.rcParams` suited for *scvelo* or *scanpy*. Use *None* for the default `matplotlib` values.

figsize : **[float, float] (default: None)** Width and height for default figure size.

dpi : **int (default: None)** Resolution of rendered figures - this influences the size of figures in notebooks.

dpi_save : **int (default: None)** Resolution of saved figures. This should typically be higher to achieve publication quality.

frameon : **bool (default: None)** Add frames and axes labels to scatter plots.

vector_friendly : **bool (default: True)** Plot scatter plots using *png* backend even when exporting as *pdf* or *svg*.

color_map : **str (default: None)** Convenience method for setting the default color map.

format : **{'png', 'pdf', 'svg', etc.} (default: 'pdf')** This sets the default format for saving figures: *file_format_figs*.

transparent : **bool (default: True)** Save figures with transparent back ground. Sets *rcParams['savefig.transparent']*.

ipython_format : list of *str* (default: 'png2x') Only concerns the notebook/IPython environment; see *IPython.core.display.set_matplotlib_formats* for more details.

3.1 Version 0.1.14 Dec 7, 2018

Plotting:

- New attributes *arrow_length* and *arrow_size* for flexible adjustment of embedded velocities.
- *pl.velocity_graph*: Scatter plot of embedding with cell-to-cell transition connectivities.
- *pl.velocity_embedding_stream*: Streamplot visualization of velocities.
- Improve visualization of embedded single cell velocities (autosize, colors etc.)

Tools:

- *tl.cell_fate*: compute cell-specific terminal state likelihood
- New attribute *approx=True* in *tl.velocity_graph* to enable approximate graph computation by performing cosine correlations on PCA space.

Preprocessing:

- Automatically detect whether data is already preprocessed.

3.2 Version 0.1.11 Oct 27, 2018

Plotting:

- *settings.set_figure_params()*: adjust matplotlib defaults for beautified plots
- improved default point and arrow sizes; improved quiver autoscale
- enable direct plotting of

Tools:

- *tl.velocity_confidence*: Added two confidence measures ‘velocity_confidence’ and ‘velocity_confidence_transition’.

- *tl.rank_velocity_genes*: Added functionality to rank genes for velocity characterizing groups using a t-test.
- New attribute *perc* in *tl.velocity* enables extreme quantile fit, e.g. set *perc=95*.
- New attribute *groups* in *tl.velocity* enables velocity estimation only on a subset of the data.
- Improved *tl.transition_matrix* by incorporating self-loops via *self_transitions=True* and state changes that have negative correlation with velocity (opposite direction) via *use_negative_cosines=True*

Utils:

- *utils.merge* to merge to AnnData objects such as already existing AnnData and newly generated Loom File.

3.3 Version 0.1.8 Sep 12, 2018

Plotting:

- support saving plots as pdf, png etc.
- support multiple colors and layers
- quiver autoscaling for velocity plots
- attributes added: figsize and dpi

Preprocessing:

- *filter_and_normalize()* instead of *recipe_velocity()*
- normalization of layers is done automatically when computing moments

Tools:

- *terminal_states*: computes root and end points via eigenvalue decomposition thanks to M Lange

3.4 Version 0.1.5 Sep 4, 2018

- Support writing loom files
- Support both dense and sparse layers
- Plotting bugfixes
- Added *pp.recipe_velocity()*

3.5 Version 0.1.2 Aug 21, 2018

First alpha release of scvelo.

CHAPTER 4

References

Bibliography

- [Wolf18] Wolf *et al.* (2018), *Scanpy: large-scale single-cell gene expression data analysis*, *Genome Biology*.
- [Manno18] La Manno *et al.* (2018), *RNA velocity of single cells*, *Nature*.

S

scvelo, 5

C

`clean_obs_names()` (in module `scvelo.utils`), 27
`cleanup()` (in module `scvelo.utils`), 27

D

`dentategyrus()` (in module `scvelo.datasets`), 26

F

`filter_and_normalize()` (in module `scvelo.pp`), 11
`filter_genes()` (in module `scvelo.pp`), 9
`filter_genes_dispersion()` (in module `scvelo.pp`), 9
`forebrain()` (in module `scvelo.datasets`), 26

M

`merge()` (in module `scvelo.utils`), 28
`moments()` (in module `scvelo.pp`), 12

N

`normalize_per_cell()` (in module `scvelo.pp`), 10

R

`rank_velocity_genes()` (in module `scvelo.tl`), 16
`read()` (in module `scvelo`), 7
`read_loom()` (in module `scvelo`), 8

S

`scatter()` (in module `scvelo.pl`), 17
`scvelo` (module), 5
`set_figure_params()` (in module `scvelo.settings`), 28
`show_proportions()` (in module `scvelo.utils`), 27

T

`terminal_states()` (in module `scvelo.tl`), 15
`toy_data()` (in module `scvelo.datasets`), 26
`transition_matrix()` (in module `scvelo.tl`), 15

V

`velocity()` (in module `scvelo.pl`), 18
`velocity()` (in module `scvelo.tl`), 12
`velocity_confidence()` (in module `scvelo.tl`), 16
`velocity_embedding()` (in module `scvelo.pl`), 21
`velocity_embedding()` (in module `scvelo.tl`), 14
`velocity_embedding_grid()` (in module `scvelo.pl`), 22
`velocity_embedding_stream()` (in module `scvelo.pl`), 24
`velocity_graph()` (in module `scvelo.pl`), 19
`velocity_graph()` (in module `scvelo.tl`), 13