Single-Cell Analysis in Python.

API

Import Scanpy as:

```
import scanpy as sc
```

Note

Additional functionality is available in the broader ecosystem, with some tools being wrapped in the `scanpy.external` module.

Preprocessing: `pp`

Filtering of highly-variable genes, batch-effect correction, per-cell normalization, preprocessing recipes.

Any transformation of the data matrix that is not a *tool*. Other than *tools*, preprocessing steps usually don't return an easily interpretable annotation, but perform a basic transformation on the data matrix.

Basic Preprocessing

For visual quality control, see `highest_expr_genes()` and `filter_genes_dispersion()` in `scanpy.pl`.

`pp.calculate_qc_metrics`(adata, *[, ...])	Calculate quality control metrics.
`pp.filter_cells`(data[, min_counts, ...])	Filter cell outliers based on counts and numbers
`pp.filter_genes`(data[, min_counts, ...])	Filter genes based on number of cells or counts.
`pp.highly_variable_genes`(adata[, layer, ...])	Annotate highly variable genes [Satija15] [Zheng
`pp.log1p`(X, *[, base, copy, chunked, ...])	Logarithmize the data matrix.
`pp.pca`(data[, n_comps, zero_center, ...])	Principal component analysis [Pedregosa11].
`pp.normalize_total`(adata[, target_sum, ...])	Normalize counts per cell.

`pp.regress_out` (adata, keys[, n_jobs, copy])	Regress out (mostly) unwanted sources of variati
`pp.scale` (X[, zero_center, max_value, copy, ...])	Scale data to unit variance and zero mean.
`pp.subsample` (data[, fraction, n_obs, ...])	Subsample to a fraction of the number of observ
`pp.downsample_counts` (adata[, ...])	Downsample counts from count matrix.

Recipes

`pp.recipe_zheng17` (adata[, n_top_genes, log, ...])	Normalization and filtering as of [Zheng17].
`pp.recipe_weinreb17` (adata[, log, ...])	Normalization and filtering as of [Weinreb17].
`pp.recipe_seurat` (adata[, log, plot, copy])	Normalization and filtering as of Seurat [Satija15

Batch effect correction

Also see [Data integration]. Note that a simple batch correction method is available via `pp.regress_out()`. Checkout `scanpy.external` for more.

`pp.combat` (adata[, key, covariates, inplace])	ComBat function for batch effect correction [Johnso

Neighbors

`pp.neighbors` (adata[, n_neighbors, n_pcs, ...])	Compute a neighborhood graph of observations [M

Tools: `tl`

Any transformation of the data matrix that is not *preprocessing*. In contrast to a *preprocessing* function, a *tool* usually adds an easily interpretable annotation to the data matrix, which can then be visualized with a corresponding plotting function.

Embeddings

`tl.pca` (data[, n_comps, zero_center, ...])	Principal component analysis [Pedregosa11].
`tl.tsne` (adata[, n_pcs, use_rep, perplexity, ...])	t-SNE [Maaten08] [Amir13] [Pedregosa11].
`tl.umap` (adata[, min_dist, spread, ...])	Embed the neighborhood graph using UMAP [M
`tl.draw_graph` (adata[, layout, init_pos, ...])	Force-directed graph drawing [Islam11] [Jacomy
`tl.diffmap` (adata[, n_comps, neighbors_key, ...])	Diffusion Maps [Coifman05] [Haghverdi15] [W

Compute densities on embeddings.

`tl.embedding_density` (adata[, basis, ...])	Calculate the density of cells in an embedding (per condi

Clustering and trajectory inference

`tl.leiden` (adata[, resolution, restrict_to, ...])	Cluster cells into subgroups [Traag18].
`tl.louvain` (adata[, resolution, ...])	Cluster cells into subgroups [Blondel08] [Levine15
`tl.dendrogram` (adata, groupby[, n_pcs, ...])	Computes a hierarchical clustering for the given `g`
`tl.dpt` (adata[, n_dcs, n_branchings, ...])	Infer progression of cells through geodesic distanc
`tl.paga` (adata[, groups, use_rna_velocity, ...])	Mapping out the coarse-grained connectivity struc

Data integration

`tl.ingest` (adata, adata_ref[, obs, ...])	Map labels and embeddings from reference data to new da

Marker genes

`tl.rank_genes_groups` (adata, groupby[, ...])	Rank genes for characterizing groups.
`tl.filter_rank_genes_groups` (adata[, key, ...])	Filters out genes based on log fold change and frac categories.
`tl.marker_gene_overlap` (adata, ...[, key, ...])	Calculate an overlap score between data-deriven m

Gene scores, Cell cycle

`tl.score_genes` (adata, gene_list[, ...])	Score a set of genes [Satija15].
`tl.score_genes_cell_cycle` (adata, s_genes, ...)	Score cell cycle genes [Satija15].

Simulations

`tl.sim` (model[, params_file, tmax, ...])	Simulate dynamic gene expression data [Wittmann09] [W

Plotting: `pl`

The plotting module `scanpy.pl` largely parallels the `tl.*` and a few of the `pp.*` functions. For most tools and for some preprocessing functions, you'll find a plotting function with the same name.

See → tutorial: plotting/core for an overview of how to use these functions.

Note

See the Settings section for all important plotting configurations.

Generic

`pl.scatter` (adata[, x, y, color, use_raw, ...])	Scatter plot along observations or variables axes.
`pl.heatmap` (adata, var_names, groupby[, ...])	Heatmap of the expression values of genes.
`pl.dotplot` (adata, var_names, groupby[, ...])	Makes a *dot plot* of the expression values of `var_`
`pl.tracksplot` (adata, var_names, groupby[, ...])	In this type of plot each var_name is plotted as a
`pl.violin` (adata, keys[, groupby, log, ...])	Violin plot.
`pl.stacked_violin` (adata, var_names, groupby)	Stacked violin plots.
`pl.matrixplot` (adata, var_names, groupby[, ...])	Creates a heatmap of the mean expression value
`pl.clustermap` (adata[, obs_keys, use_raw, ...])	Hierarchically-clustered heatmap.
`pl.ranking` (adata, attr, keys[, dictionary, ...])	Plot rankings.
`pl.dendrogram` (adata, groupby, *[, ...])	Plots a dendrogram of the categories defined in

Classes

These classes allow fine tuning of visual parameters.

`pl.DotPlot` (adata, var_names, groupby[, ...])	Allows the visualization of two values that are en
`pl.MatrixPlot` (adata, var_names, groupby[, ...])	Allows the visualization of values using a color m
`pl.StackedViolin` (adata, var_names, groupby)	Stacked violin plots.

Preprocessing

Methods for visualizing quality control and results of preprocessing functions.

`pl.highest_expr_genes` (adata[, n_top, show, ...])	Fraction of counts assigned to each gene over all
`pl.filter_genes_dispersion` (result[, log, ...])	Plot dispersions versus means for genes.
`pl.highly_variable_genes` (adata_or_result[, ...])	Plot dispersions or normalized variance versus m

Tools

Methods that extract and visualize tool-specific annotation in an `AnnData` object. For any method in module `tl`, there is a method with the same name in `pl`.

PCA

`pl.pca`(adata, *[, color, gene_symbols, ...])	Scatter plot in PCA coordinates.
`pl.pca_loadings`(adata[, components, ...])	Rank genes according to contributions to PCs.
`pl.pca_variance_ratio`(adata[, n_pcs, log, ...])	Plot the variance ratio.
`pl.pca_overview`(adata, **params)	Plot PCA results.

Embeddings

`pl.tsne`(adata, *[, color, gene_symbols, ...])	Scatter plot in tSNE basis.
`pl.umap`(adata, *[, color, gene_symbols, ...])	Scatter plot in UMAP basis.
`pl.diffmap`(adata, *[, color, gene_symbols, ...])	Scatter plot in Diffusion Map basis.
`pl.draw_graph`(adata, *[, color, ...])	Scatter plot in graph-drawing basis.
`pl.spatial`(adata, *[, color, gene_symbols, ...])	Scatter plot in spatial coordinates.
`pl.embedding`(adata, basis, *[, color, ...])	Scatter plot for user specified embedding basis (e.

Compute densities on embeddings.

`pl.embedding_density`(adata[, basis, key, ...])	Plot the density of cells in an embedding (per conditi

Branching trajectories and pseudotime, clustering

Visualize clusters using one of the embedding methods passing `color='louvain'`.

`pl.dpt_groups_pseudotime`(adata[, color_map, ...])	Plot groups and pseudotime.
`pl.dpt_timeseries`(adata[, color_map, show, ...])	Heatmap of pseudotime series.
`pl.paga`(adata[, threshold, color, layout, ...])	Plot the PAGA graph through thresholding low-
`pl.paga_path`(adata, nodes, keys[, use_raw, ...])	Gene expression and annotation changes along
`pl.paga_compare`(adata[, basis, edges, ...])	Scatter and PAGA graph side-by-side.

Marker genes

`pl.rank_genes_groups` (adata[, groups, ...])	Plot ranking of genes.
`pl.rank_genes_groups_violin` (adata[, groups, ...])	Plot ranking of genes for all tested comparisons
`pl.rank_genes_groups_stacked_violin` (adata[, ...])	Plot ranking of genes using stacked_violin plot (
`pl.rank_genes_groups_heatmap` (adata[, ...])	Plot ranking of genes using heatmap plot (see
`pl.rank_genes_groups_dotplot` (adata[, ...])	Plot ranking of genes using dotplot plot (see `do`
`pl.rank_genes_groups_matrixplot` (adata[, ...])	Plot ranking of genes using matrixplot plot (see
`pl.rank_genes_groups_tracksplot` (adata[, ...])	Plot ranking of genes using heatmap plot (see

Simulations

`pl.sim` (adata[, tmax_realization, ...])	Plot results of simulation.

Reading

Note

For reading annotation use pandas.read_… and add it to your `anndata.AnnData` object. The following read functions are intended for the numeric data in the data matrix `X`.

Read common file formats using

`read` (filename[, backed, sheet, ext, ...])	Read file and return `AnnData` object.

Read 10x formatted hdf5 files and directories containing `.mtx` files using

`read_10x_h5` (filename[, genome, gex_only, ...])	Read 10x-Genomics-formatted hdf5 file.
`read_10x_mtx` (path[, var_names, make_unique, ...])	Read 10x-Genomics-formatted mtx directory.
`read_visium` (path[, genome, count_file, ...])	Read 10x-Genomics-formatted visum dataset.

Read other formats using functions borrowed from `anndata`

`read_h5ad` (filename[, backed, as_sparse, ...])	Read *.h5ad*-formatted hdf5 file.
`read_csv` (filename[, delimiter, ...])	Read *.csv* file.
`read_excel` (filename, sheet[, dtype])	Read *.xlsx* (Excel) file.
`read_hdf` (filename, key)	Read *.h5* (hdf5) file.
`read_loom` (filename, *[, sparse, cleanup, ...])	Read *.loom*-formatted hdf5 file.
`read_mtx` (filename[, dtype])	Read *.mtx* file.

`read_text` (filename[, delimiter, ...])	Read *.txt*, *.tab*, *.data* (text) file.
`read_umi_tools` (filename[, dtype])	Read a gzipped condensed count matrix from umi_to

Get object from `AnnData`: `get`

The module `sc.get` provides convenience functions for getting values back in useful formats.

`get.obs_df` (adata[, keys, obsm_keys, layer, ...])	Return values for observations in adata.
`get.var_df` (adata[, keys, varm_keys, layer])	Return values for observations in adata.
`get.rank_genes_groups_df` (adata, group, *[, ...])	`scanpy.tl.rank_genes_groups()` results in the form of a `DataFrame`.

Queries

This module provides useful queries for annotation and enrichment.

`queries.biomart_annotations` (org, attrs, *[, ...])	Retrieve gene annotations from ensembl biomart.
`queries.gene_coordinates` (org, gene_name, *)	Retrieve gene coordinates for specific organism th
`queries.mitochondrial_genes` (org, *[, ...])	Mitochondrial gene symbols for specific organism
`queries.enrich` (container, *[, org, ...])	Get enrichment for DE results.

Metrics

Collections of useful measurements for evaluating results.

`metrics.confusion_matrix` (orig, new[, data, ...])	Given an original and new set of labels, create a la
`metrics.gearys_c` (adata, *[, vals, ...])	Calculate Geary's C, as used by VISION.
`metrics.morans_i` (adata, *[, vals, ...])	Calculate Moran's I Global Autocorrelation Statist

Experimental

New methods that are in early development which are not (yet) integrated in Scanpy core.

`experimental.pp.normalize_pearson_residuals` (...)	Applies analytic Pearson residual normaliza
`experimental.pp.normalize_pearson_residuals_pca` (...)	Applies analytic Pearson residual normaliza

`experimental.pp.highly_variable_genes`(adata, *)	Select highly variable genes using analytic I
`experimental.pp.recipe_pearson_residuals`(...)	Full pipeline for HVG selection and normali

Classes

`AnnData` is reexported from `anndata`.

Represent data as a neighborhood structure, usually a knn graph.

`Neighbors`(adata[, n_dcs, neighbors_key])	Data represented as graph of nearest neighbors.

Settings

A convenience function for setting some default `matplotlib.rcParams` and a high-resolution jupyter display backend useful for use in notebooks.

`set_figure_params`([scanpy, dpi, dpi_save, ...])	Set resolution/size, styling and format of figures.

An instance of the `ScanpyConfig` is available as `scanpy.settings` and allows configuring Scanpy.

`_settings.ScanpyConfig`(*[, verbosity, ...])	Config manager for scanpy.

Some selected settings are discussed in the following.

Influence the global behavior of plotting functions. In non-interactive scripts, you'd usually want to set `settings.autoshow` to `False`.

`autoshow`	Automatically show figures if `autosave == False` (default `True`).
`autosave`	Automatically save figures in `figdir` (default `False`).

The default directories for saving figures, caching files and storing datasets.

`figdir`	Directory for saving figures (default `'./figures/'`).
`cachedir`	Directory for cache files (default `'./cache/'`).
`datasetdir`	Directory for example `datasets` (default `'./data/'`).

The verbosity of logging output, where verbosity levels have the following meaning: 0='error', 1='warning', 2='info', 3='hint', 4=more details, 5=even more details, etc.

`verbosity`	Verbosity level (default `warning`)

Print versions of packages that might influence numerical results.

`logging.print_header`(*[, file])	Versions that might influence the numerical results.
`logging.print_versions`(*[, file])	Print versions of imported packages, OS, and jupyter environme

Datasets

`datasets.blobs`([n_variables, n_centers, ...])	Gaussian Blobs.
`datasets.ebi_expression_atlas`(accession, *)	Load a dataset from the EBI Single Cell Expression A
`datasets.krumsiek11`()	Simulated myeloid progenitors [Krumsiek11].
`datasets.moignard15`()	Hematopoiesis in early mouse embryos [Moignard1
`datasets.pbmc3k`()	3k PBMCs from 10x Genomics.
`datasets.pbmc3k_processed`()	Processed 3k PBMCs from 10x Genomics.
`datasets.pbmc68k_reduced`()	Subsampled and processed 68k PBMCs.
`datasets.paul15`()	Development of Myeloid Progenitors [Paul15].
`datasets.toggleswitch`()	Simulated toggleswitch.
`datasets.visium_sge`([sample_id, ...])	Processed Visium Spatial Gene Expression data fror

Deprecated functions

`pp.filter_genes_dispersion`(data[, flavor, ...])	Extract highly variable genes [Satija15] [Zheng17].
`pp.normalize_per_cell`(data[, ...])	Normalize total counts per cell.