| **Function/Method** | **Purpose** | **When to Use** | **Example Code** |
| --- | --- | --- | --- |
| scanpy.read() | Reads data into AnnData format. | To load datasets in .h5ad, .loom, or other compatible formats. | adata = sc.read("data.h5ad") |
| scanpy.pp.filter\_cells() | Filters out unwanted cells based on QC metrics. | To remove cells with low counts, genes, or high mitochondrial gene expression. | sc.pp.filter\_cells(adata, min\_genes=200) |
| scanpy.pp.filter\_genes() | Filters out unwanted genes based on expression levels. | To remove lowly expressed genes that don't contribute to analysis. | sc.pp.filter\_genes(adata, min\_cells=3) |
| scanpy.pp.normalize\_total() | Normalizes counts per cell to a fixed total count. | Apply to make expression values comparable across cells. | sc.pp.normalize\_total(adata, target\_sum=1e4) |
| scanpy.pp.log1p() | Log-transforms the data matrix. | Use after normalization to reduce variance and stabilize gene expression values. | sc.pp.log1p(adata) |
| scanpy.pp.scale() | Z-scores the data matrix. | Use for PCA or other dimensionality reduction techniques that assume normalized data. | sc.pp.scale(adata, max\_value=10) |
| scanpy.pp.neighbors() | Computes a neighborhood graph of observations (cells) | Use to capture the underlying structure and relationships in the data | sc.pp.neighbors(adata, n\_neighbors=10, n\_pcs=30) |
| scanpy.pp.pca() | Computes Principal Component Analysis (PCA). | Use to reduce dimensionality and identify major trends in data. | sc.tl.pca(adata) |
| scanpy.tl.tsne() | Computes t-SNE for data visualization. | Use to visualize clusters in 2D or 3D when working with high-dimensional datasets. | sc.tl.tsne(adata, n\_pcs=50) |
| scanpy.tl.marker\_gene\_overlap() | Computes the overlap of marker genes between clusters or groups. | Use to identify and visualize shared and unique marker genes. | sc.tl.marker\_gene\_overlap(adata, reference\_markers, ...) |
| scanpy.tl.umap() | Computes UMAP for visualization. | Use for visualizing cell clusters in a reduced dimensional space. | sc.tl.umap(adata) |
| scanpy.tl.leiden() | Performs clustering using the Leiden algorithm. | Use to identify clusters or cell populations in your dataset. | sc.tl.leiden(adata, resolution=0.5) |
| scanpy.tl.rank\_genes\_groups() | Identifies marker genes for each cluster. | Use to find differentially expressed genes between clusters. | sc.tl.rank\_genes\_groups(adata, groupby='leiden', method='wilcoxon') |
| scanpy.tl.dendrogram() | Creates a hierarchical dendrogram of cell clusters. | Use to explore relationships between clusters. | sc.tl.dendrogram(adata, groupby='leiden') |
| scanpy.pl.pca() | Plots PCA results. | Use to visualize principal components in scatterplots. | sc.pl.pca(adata, color='leiden') |
| scanpy.pl.umap() | Plots UMAP results. | Use for cluster visualization in 2D or 3D. | sc.pl.umap(adata, color='leiden') |
| scanpy.pl.tsne() | Plots t-SNE results. | Use to visualize t-SNE results and explore cluster separability. | sc.pl.tsne(adata, color='cell\_type') |
| scanpy.pl.rank\_genes\_groups() | Visualizes marker genes in clusters. | Use to identify gene expression patterns associated with specific cell populations. | sc.pl.rank\_genes\_groups(adata, n\_genes=10, sharey=False) |
| scanpy.tl.paga() | Computes trajectory inference with PAGA. | Use to infer connectivity between clusters and pseudotime ordering. | sc.tl.paga(adata, groups='leiden') |
| scanpy.tl.dpt() | Computes diffusion pseudotime (DPT). | Use to infer temporal progression or developmental trajectories in single-cell datasets. | sc.tl.dpt(adata) |
| scanpy.pl.matrixplot() | Creates a matrix plot of gene expression. | Use for comparing expression of selected genes across clusters. | sc.pl.matrixplot(adata, var\_names=['Gene1', 'Gene2'], groupby='leiden') |
| scanpy.pl.heatmap() | Generates a heatmap of gene expression. | Use to visualize differentially expressed genes across clusters. | sc.pl.heatmap(adata, var\_names=adata.var\_names[:20], groupby='leiden') |
| scanpy.pl.dotplot() | Creates a dot plot for selected genes. | Use to visualize average expression and frequency of genes across clusters. | sc.pl.dotplot(adata, var\_names=['Gene1', 'Gene2'], groupby='leiden') |
| scanpy.external.pp.bbknn() | Batch correction using BBKNN. | Use when analyzing data with batch effects to align datasets across conditions or replicates. | import scanpy.external as sce; sce.pp.bbknn(adata, batch\_key='batch') |