FEATS

Release 1.0.1

Edwin Vans

Aug 04, 2021

CONTENTS:

CHAPTER

ONE

FEATS

1.1 feats package

1.1.1 Submodules

1.1.2 feats.feats_batchcorrection module

feats.feats_batchcorrection.**AdjustShiftVariance**(*X_ref*, *X*, *Corr*, *sigma*)

feats.feats_batchcorrection.**ComputeCorrectionVectors**(*V*, *X*, *tar_idx*, *sigma*)

feats.feats_batchcorrection.**ComputeMNNPairs**(*X_ref*, *X*, *k*)

feats.feats_batchcorrection.**ComputeSpan**(*X*, *idx*, *svd_dim*)

feats.feats_batchcorrection.**CorrectBatches**(*batches*, *correct_order*, *k=20*, *sigma=10*, *svd_dim=0*, *adj_var=True*)

Implements the Mutual Nearest Neighbour (MNN) approach to correcting batch effects in the integrated and merged datasets.

Parameters

- **batches** (*SingleCell*) – An integrated and merged SingleCell object (dataset) to correct. This dataset should have a 'batch' column in the celldata dataframe with batch information.
- **correct_order** (*list*) – A Python list by the batch number or name representing the order in which to correct the batches. The first batch in the list is the reference batch.
- **k** (*int, optional*) – The number of Mutual Nearest Neighbours to compute between the datasets to be corrected. default (20).
- **signa** (*float, optional*) – A parameter controlling the width of the Gaussian kernel smoothing function.
- **svd_dim** (*int, optional*) – The dimensionality of U when computing the SVD of data, where U, S, V^T = svd(X). default (0).
- **adj_var** (*bool, optional*) – Whether or not to adjust the variance of the computed batch vectors. True (default) if adjust the variance, False if not.

Returns The corrected dataset.

Return type SingleCell

Raises **ValueError** – If the length of 'correct_order' is less than 2, and if no batch information is found in the input parameter 'batches'.

`feats.feats_batchcorrection.`**`IntegrateBatches`**(*batches*, *name_by*)

Integrates a list of SingleCell objects storing single-cell data. This function first finds common genes across all the datasets in the list and then merges all the datasets together in one SingleCell object. If no common genes are found between the datasets, then an error is generated with a message.

Parameters

- **`batches`** (*`list`*) – A Python list of SingleCell objects (datasets) to integrate.
- **`name_by`** (*`list`*) – A list of str representing column names in the SingleCell objects where gene names are stored. The order in this list is same as the order of SingleCell objects in the batches list.

Returns A merged dataset where the number of rows (d) is equal to the number of common genes in all the datasets and the number columns (n) is the sum of the number of cells in all the datasets.

Return type SingleCell

Raises **`ValueError`** – If no common genes are found between the SingleCell datasets in the batches list.

`feats.feats_batchcorrection.`**`LogspaceAdd`**(*log_array*)

`feats.feats_batchcorrection.`**`MergeBatches`**(*batches*)

Merges a list of SingleCell objects storing single-cell data. This function assumes that the datasets are already integrated and the number and type of genes in all the datasets are the same. If this is not the case, then an error is generated.

Parameters **`batches`** (*`list`*) – A Python list of SingleCell objects (datasets) to merge. It is assumed that the SingleCell datasets are already integrated.

Returns A merged dataset where the number of rows (d) is equal to the number genes the datasets (which is the same for all the datasets) and the number columns (n) is the sum of the number of cells in all the datasets.

Return type SingleCell

Raises **`ValueError`** – If the number and type of genes are not the same in all the datasets.

`feats.feats_batchcorrection.`**`ReducedMean`**(*X*, *idx*)

`feats.feats_batchcorrection.`**`RemoveBatchEffects`**(*ref_batch*, *tar_batch*, *k*, *sigma*, *svd_dim*, *adj_var*)

`feats.feats_batchcorrection.`**`SqDistToLine`**(*ref*, *grad*, *point*)

1.1.3 feats.feats_clustering module

`feats.feats_clustering.`**`AnovaHierarchical`**(*sc*, *k*, *normalization*, *q*)

`feats.feats_clustering.`**`Cluster`**(*sc*, *k='gap'*, *k_max=10*, *normalization='mean'*, *q=None*)

Clusters gene expression data stored in SingleCell object. Stores the cluster information (labels) in the celldata assay of the SingleCell object. The column name under which it stores the cluster information is 'FEATS_k_Clusters', where k is the number of clusters, which the method can estimate or is defined by the user as an int or an int list.

Parameters

- **`sc`** (*`SingleCell`*) – The SingleCell dataset containing gene expression data to be clustered.

- **k** (*int, list or str, optional*) – This is an optional input for the number of clusters in the dataset. It is either an int, a Python list of ints or a str. If it is an int, the method will cluster the data into k groups. If it is a Python list, the method will cluster the data into the number of groups specified in the list and store the cluster information for all the values in the list. If it is the string 'gap', the method will use gap statistic to estimate the number of clusters and cluster the data into the number of groups estimated.
- **k_max** (*int, optional*) – The upper limit of the number of clusters when using gap statistic to estimate the number of clusters. Ignored if k is not 'gap'. Default 10.
- **normalization** (*str, optional*) – The normalization method to use when normalizing the gene expression data. By default the normalized data is not saved in the SingleCell object after using the Cluster function. If you want to save the normalized counts in the SingleCell object, you can use the FeatureNormalize() function. Type help(FeatureNormalize) for more information. Options for the normalization are 'mean' for z-score normalization (default), 'l2' for L2 normalization, 'cosine' for Cosine normalization and None for no normalization.
- **q** (*list, optional*) – This parameter is an int list which specifies the number of top features to select. The clustering algorithm selects each q_i top features in this list, perofrms clustering, computes the clustering score,and then determines the optimal number of features which give best clustering. By default q = [1, 2, . . . , 5% of d], where d is the number of features.

Returns

- *SingleCell* – The SingleCell object with the cluster information stored in the celldata assay.
- *int* – The number of clusters in the dataset, k. This output is useful when k is not known and estimated by the algorithm.

Raises ValueError – If k_max is < 2. If k > n and < 2, where n is the number of samples in the dataset. If unknown input types/class is detected.

`feats.feats_clustering.`**`ClusteringScore`**(*X*, *labels*)

`feats.feats_clustering.`**`SelectTopNScores`**(*clustering_score*, *n*)

1.1.4 feats.feats_detectoutliers module

`feats.feats_detectoutliers.`**`DetectOutliers`**(*sc*, *cluster_label*, *red_dim=2*, *outlier_prob_thres=0.0001*)

This function implements the outlier detection scheme of FEATS.

Parameters

- **sc** (*SingleCell*) – The SingleCell object which contains the data and metadata of genes and cells
- **cluster_label** (*str*) – The name of the column in celldata assay of sc which stores the cluster labels of the cells
- **red_dim** (*int, optional*) – The reduced dimentionality in which the outliers are computed. Default 2.
- **outlier_prob_thres** (*float*) – The probability threshold for samples to be classified as outliers. Default 10^-4.

Returns The single cell object containing the outlier analysis information in the celldata assay. It contains the following columns in the celldata assay with the outlier information:

'FEATS_Outliers' - A column with the value True if the respective cell is an outlier, False otherwise. 'FEATS_Msd' - The computed Mahalanobis squared distance for the respective cells. 'FEATS_Outlier_Score' - The outlier score for the respective cells. 'FEATS_Oos' - A column with the value True if the respective cell was not used by the Minimum Covariance Determinant (MCD) algorithm in computing the robust mean and covariance matrix.

Return type SingleCell

1.1.5 feats.feats_filtering module

`feats.feats_filtering.`**`CellFilter`**(*sc*, *min_genes*, *max_genes*, *expr_thres=0*)

Filters the cells in which the genes are lowly and/or highly expressed. Returns the sliced SingleCell object with the cells selected through the filtering criteria. This functions skips filtering and warns the user if the filtering criteria filters out all the samples/cells.

Parameters

- **sc** (*SingleCell*) – The SingleCell object containing gene expression data.
- **min_genes** (*int*) – The minimum number of genes in the cell in which the genes must be expressed with the expression threshold. The value should be >= 0 and <= d, where d is the number of genes the the dataset.
- **max_genes** (*int*) – The maximum number of genes in the cell in which the gene must be expressed with the expression threshold. The value should be >= 0 and <= d, where d is the number of genes the the dataset.
- **expr_thres** (*int, optional*) – The expression threshold. Default 0. The gene is considered not expressed if the expression count is <= this threshold.

Returns **sc** – A sliced SingleCell object containing the filtered cells.

Return type SingleCell

`feats.feats_filtering.`**`GeneFilter`**(*sc*, *min_cells*, *max_cells*, *expr_thres=0*)

Filters the lowly and/or highly expressed genes stored in the SingleCell object based on the expression counts. Returns the sliced SingleCell object with the genes selected through the filtering criteria. This functions skips filtering and warns the user if the filtering criteria filters out all the genes.

Parameters

- **sc** (*SingleCell*) – The SingleCell object containing gene expression data.
- **min_cells** (*int*) – The minimum number of cells in which the gene must be expressed with the expression threshold. The value should be >= 0 and <= n, where n is the number of cells the the dataset.
- **max_cells** (*int*) – The maximum number of cells in which the gene must be expressed with the expression threshold. The value should be >= 0 and <= n, where n is the number of cells the the dataset.
- **expr_thres** (*int, optional*) – The expression threshold. Default 0. The gene is considered not expressed if the expression count is <= this threshold.

Returns **sc** – A sliced SingleCell object containing the filtered genes.

Return type SingleCell

`feats.feats_filtering.`**`HVGFilter`**(*sc*, *num_genes*, *name_by='gene_names'*)

Filters/Selects the Highly Variable Genes in the SingleCell object by computing the dispersion of each gene. Returns a sliced SingleCell object containing the top Highly Variable Genes. Stores additional information such as dispersion ('FEATS_dispersion), mean ('FEATS_mean') and variance ('FEATS_variance') in the genedata assay of SingleCell object.

Parameters

- **`sc`** (*`SingleCell`*) – The SingleCell object containing gene expression data.
- **`num_genes`** (*`int`*) – The number of Highly Variable Genes to select.
- **`name_by`** (*`str`*) – The name of the column in SingleCell object that stores the gene names. This is used to print the top Highly Variable Genes.

Returns **sc** – A sliced SingleCell object containing the top Highly Variable Genes.

Return type SingleCell

`feats.feats_filtering.`**`LogFilter`**(*sc*)

Computes and stores the log-transformed gene expression data stored in SingleCell object. Here Log2 is performed after adding a pseudocount of 1.

Parameters **`sc`** (*`SingleCell`*) – The SingleCell object containing gene expression data.

Returns **sc** – The SingleCell object containing the log-transfprmed gene expression data.

Return type SingleCell

1.1.6 feats.feats_gapstatistics module

`feats.feats_gapstatistics.`**`ComputeWk`**(*X*, *labels*, *classes*)

X - (d x n) data matrix, where n is samples and d is dimentionality lables - n dimentional vector which are class labels

`feats.feats_gapstatistics.`**`GapStatistics`**(*sc_obj*, *k_max*, *B*)

Computes the gap statistic and estimates the number of clusters in the gene expression dataset contained in SingleCell object.

Parameters

- **`sc_obj`** (*`SingleCell`*) – The SingleCell object containing gene expression data and the metadata.
- **`k_max`** (*`int`*) – The upper limit of the number of clusters.
- **`B`** (*`int`*) – The number of reference datasets to generate to compute the gap quantities.

Returns

- **est_clusters** (*int*) – The estimate of the number of clusters in the dataset.
- **Gap_k** (*list*) – The gap statistic quantity for gap. The list contains gap values for each k, where k = 1, 2, . . . , k_max.
- **s_k** (*list*) – The gap statistic quantity for standard deviation. The list contains the standard deviation for each k, where k = 1, 2, . . . , k_max.
- **W_k** (*list*) – A gap statistic quantity for each k, where k = 1, 2, . . . , k_max.
- **w_bar** (*list*) – A gap statistic quantity for each k, where k = 1, 2, . . . , k_max.

1.1.7 feats.feats_transformations module

`feats.feats_transformations.`**`GramMatrix`**(*K*)

`feats.feats_transformations.`**`PCA`**(*sc*, *n_comp=1*, *dist_or_kernel='linear'*)

Computes and stores the Principal Components of the gene expression data stored in the SingleCell object.

Parameters

- **sc** (*SingleCell*) – The SingleCell object containing gene expression data.
- **n_comp** (*int, optional*) – The number of Principal Components to compute. Default 1.
- **dist_or_kernel** (*str, optional*) – The distance metric or the kernel to compute. If a distance metric is passed, it computes the pairwise distance between the cells and then converts the distance metrics to kernels. If a kernel is passed, it computes the kernel. Valid values are 'linear' (default) for linear kernel, 'spearmans' for Spearmans distance, 'euclidean' for Euclidean distance and 'pearsons' for Pearsons distance.

Returns **sc** – The SingleCell object containing the dimensionnality reduced gene expression data. The reduced dimensionality is n_comp. The gene names are removed and the features in the reduced space are named 'PC1', 'PC2' and so on.

Return type SingleCell

`feats.feats_transformations.`**`dist_to_kernel`**(*D*)

`feats.feats_transformations.`**`euclidean`**(*X*)

X is a (n x d) matrix where rows are samples Returns D which is a (n x n) matrix of distances between samples

`feats.feats_transformations.`**`linear_kernel`**(*X*)

X is a (n x d) matrix where rows are samples Returns K which is a (n x n) kernel matrix

`feats.feats_transformations.`**`pearsons`**(*X*)

X is a (n x d) matrix where rows are samples Returns D which is a (n x n) matrix of distances between samples

`feats.feats_transformations.`**`spearmans`**(*X*)

X is a (n x d) matrix where rows are samples Returns D which is a (n x n) matrix of distances between samples

1.1.8 feats.feats_utils module

`feats.feats_utils.`**`FeatureNormalize`**(*sc*, *norm*)

Computes and stores the normalized gene expression data stored in SingleCell object.

Parameters

- **sc** (*SingleCell*) – The SingleCell object containing gene expression data.
- **norm** (*str*) – The normalization to perform. Accepted values are 'l2', 'mean', 'norm6' and 'cosine'.

Returns **sc** – The SingleCell object containing the normalized gene expression data.

Return type SingleCell

1.1.9 Module contents

FEATS

FEATS is a new Python tool for performing the following downstream analysis on single-cell RNA-seq datasets: 1. Clustering 2. Estimating the number of clusters 3. Outlier detection 4. Batch correction and integration of data from multiple experiments

See https://github.com/edwinv87/feats for more information and documentation.

`feats.`**`CellFilter`**(*sc*, *min_genes*, *max_genes*, *expr_thres=0*)

Filters the cells in which the genes are lowly and/or highly expressed. Returns the sliced SingleCell object with the cells selected through the filtering criteria. This functions skips filtering and warns the user if the filtering criteria filters out all the samples/cells.

Parameters

- **sc** (*SingleCell*) – The SingleCell object containing gene expression data.
- **min_genes** (*int*) – The minimum number of genes in the cell in which the genes must be expressed with the expression threshold. The value should be >= 0 and <= d, where d is the number of genes the the dataset.
- **max_genes** (*int*) – The maximum number of genes in the cell in which the gene must be expressed with the expression threshold. The value should be >= 0 and <= d, where d is the number of genes the the dataset.
- **expr_thres** (*int, optional*) – The expression threshold. Default 0. The gene is considered not expressed if the expression count is <= this threshold.

Returns sc – A sliced SingleCell object containing the filtered cells.

Return type SingleCell

`feats.`**`Cluster`**(*sc*, *k='gap'*, *k_max=10*, *normalization='mean'*, *q=None*)

Clusters gene expression data stored in SingleCell object. Stores the cluster information (labels) in the cell-data assay of the SingleCell object. The column name under which it stores the cluster information is 'FEATS_k_Clusters', where k is the number of clusters, which the method can estimate or is defined by the user as an int or an int list.

Parameters

- **sc** (*SingleCell*) – The SingleCell dataset containing gene expression data to be clustered.
- **k** (*int, list or str, optional*) – This is an optional input for the number of clusters in the dataset. It is either an int, a Python list of ints or a str. If it is an int, the method will cluster the data into k groups. If it is a Python list, the method will cluster the data into the number of groups specified in the list and store the cluster information for all the values in the list. If it is the string 'gap', the method will use gap statistic to estimate the number of clusters and cluster the data into the number of groups estimated.
- **k_max** (*int, optional*) – The upper limit of the number of clusters when using gap statistic to estimate the number of clusters. Ignored if k is not 'gap'. Default 10.
- **normalization** (*str, optional*) – The normalization method to use when normalizing the gene expression data. By default the normalized data is not saved in the SingleCell object after using the Cluster function. If you want to save the normalized counts in the SingleCell object, you can use the FeatureNormalize() function. Type help(FeatureNormalize) for more information. Options for the normalization are 'mean' for z-score normalization

(default), 'l2' for L2 normalization, 'cosine' for Cosine normalization and None for no normalization.

- **q** (*list, optional*) – This parameter is an int list which specifies the number of top features to select. The clustering algorithm selects each q_i top features in this list, perofrms clustering, computes the clustering score,and then determines the optimal number of features which give best clustering. By default q = [1, 2, . . . , 5% of d], where d is the number of features.

Returns

- *SingleCell* – The SingleCell object with the cluster information stored in the celldata assay.
- *int* – The number of clusters in the dataset, k. This output is useful when k is not known and estimated by the algorithm.

Raises **ValueError** – If k_max is < 2. If k > n and < 2, where n is the number of samples in the dataset. If unknown input types/class is detected.

feats.**CorrectBatches**(*batches*, *correct_order*, *k=20*, *sigma=10*, *svd_dim=0*, *adj_var=True*)

Implements the Mutual Nearest Neighbour (MNN) approach to correcting batch effects in the integrated and merged datasets.

Parameters

- **batches** (*SingleCell*) – An integrated and merged SingleCell object (dataset) to correct. This dataset should have a 'batch' column in the celldata dataframe with batch information.
- **correct_order** (*list*) – A Python list by the batch number or name representing the order in which to correct the batches. The first batch in the list is the reference batch.
- **k** (*int, optional*) – The number of Mutual Nearest Neighbours to compute between the datasets to be corrected. default (20).
- **signa** (*float, optional*) – A parameter controlling the width of the Gaussian kernel smoothing function.
- **svd_dim** (*int, optional*) – The dimensionality of U when computing the SVD of data, where U, S, V^T = svd(X). default (0).
- **adj_var** (*bool, optional*) – Whether or not to adjust the variance of the computed batch vectors. True (default) if adjust the variance, False if not.

Returns The corrected dataset.

Return type SingleCell

Raises **ValueError** – If the length of 'correct_order' is less than 2, and if no batch information is found in the input parameter 'batches'.

feats.**DetectOutliers**(*sc*, *cluster_label*, *red_dim=2*, *outlier_prob_thres=0.0001*)

This function implements the outlier detection scheme of FEATS.

Parameters

- **sc** (*SingleCell*) – The SingleCell object which contains the data and metadata of genes and cells
- **cluster_label** (*str*) – The name of the column in celldata assay of sc which stores the cluster labels of the cells
- **red_dim** (*int, optional*) – The reduced dimentionality in which the outliers are computed. Default 2.

- **outlier_prob_thres** (*float*) – The probability threshold for samples to be classified as outliers. Default 10^-4.

Returns The single cell object containing the outlier analysis information in the celldata assay. It contains the following columns in the celldata assay with the outlier information: 'FEATS_Outliers' - A column with the value True if the respective cell is an outlier, False otherwise. 'FEATS_Msd' - The computed Mahalanobis squared distance for the respective cells. 'FEATS_Outlier_Score' - The outlier score for the respective cells. 'FEATS_Oos' - A column with the value True if the respective cell was not used by the Minimum Covariance Determinant (MCD) algorithm in computing the robust mean and covariance matrix.

Return type SingleCell

`feats.`**`FeatureNormalize`**(*sc*, *norm*)

Computes and stores the normalized gene expression data stored in SingleCell object.

Parameters

- **sc** (*SingleCell*) – The SingleCell object containing gene expression data.
- **norm** (*str*) – The normalization to perform. Accepted values are 'l2', 'mean', 'norm6' and 'cosine'.

Returns **sc** – The SingleCell object containing the normalized gene expression data.

Return type SingleCell

`feats.`**`GapStatistics`**(*sc_obj*, *k_max*, *B*)

Computes the gap statistic and estimates the number of clusters in the gene expression dataset contained in SingleCell object.

Parameters

- **sc_obj** (*SingleCell*) – The SingleCell object containing gene expression data and the metadata.
- **k_max** (*int*) – The upper limit of the number of clusters.
- **B** (*int*) – The number of reference datasets to generate to compute the gap quantities.

Returns

- **est_clusters** (*int*) – The estimate of the number of clusters in the dataset.
- **Gap_k** (*list*) – The gap statistic quantity for gap. The list contains gap values for each k, where k = 1, 2, . . . , k_max.
- **s_k** (*list*) – The gap statistic quantity for standard deviation. The list contains the standard deviation for each k, where k = 1, 2, . . . , k_max.
- **W_k** (*list*) – A gap statistic quantity for each k, where k = 1, 2, . . . , k_max.
- **w_bar** (*list*) – A gap statistic quantity for each k, where k = 1, 2, . . . , k_max.

`feats.`**`GeneFilter`**(*sc*, *min_cells*, *max_cells*, *expr_thres=0*)

Filters the lowly and/or highly expressed genes stored in the SingleCell object based on the expression counts. Returns the sliced SingleCell object with the genes selected through the filtering criteria. This functions skips filtering and warns the user if the filtering criteria filters out all the genes.

Parameters

- **sc** (*SingleCell*) – The SingleCell object containing gene expression data.

- **min_cells** (*int*) – The minimum number of cells in which the gene must be expressed with the expression threshold. The value should be >= 0 and <= n, where n is the number of cells the the dataset.
- **max_cells** (*int*) – The maximum number of cells in which the gene must be expressed with the expression threshold. The value should be >= 0 and <= n, where n is the number of cells the the dataset.
- **expr_thres** (*int, optional*) – The expression threshold. Default 0. The gene is considered not expressed if the expression count is <= this threshold.

Returns **sc** – A sliced SingleCell object containing the filtered genes.

Return type SingleCell

`feats.`**`HVGFilter`**(*sc*, *num_genes*, *name_by='gene_names'*)

Filters/Selects the Highly Variable Genes in the SingleCell object by computing the dispersion of each gene. Returns a sliced SingleCell object containing the top Highly Variable Genes. Stores additional information such as dispersion ('FEATS_dispersion), mean ('FEATS_mean') and variance ('FEATS_variance') in the genedata assay of SingleCell object.

Parameters

- **sc** (*SingleCell*) – The SingleCell object containing gene expression data.
- **num_genes** (*int*) – The number of Highly Variable Genes to select.
- **name_by** (*str*) – The name of the column in SingleCell object that stores the gene names. This is used to print the top Highly Variable Genes.

Returns **sc** – A sliced SingleCell object containing the top Highly Variable Genes.

Return type SingleCell

`feats.`**`IntegrateBatches`**(*batches*, *name_by*)

Integrates a list of SingleCell objects storing single-cell data. This function first finds common genes across all the datasets in the list and then merges all the datasets together in one SingleCell object. If no common genes are found between the datasets, then an error is generated with a message.

Parameters

- **batches** (*list*) – A Python list of SingleCell objects (datasets) to integrate.
- **name_by** (*list*) – A list of str representing column names in the SingleCell objects where gene names are stored. The order in this list is same as the order of SingleCell objects in the batches list.

Returns A merged dataset where the number of rows (d) is equal to the number of common genes in all the datasets and the number columns (n) is the sum of the number of cells in all the datasets.

Return type SingleCell

Raises **`ValueError`** – If no common genes are found between the SingleCell datasets in the batches list.

`feats.`**`LogFilter`**(*sc*)

Computes and stores the log-transformed gene expression data stored in SingleCell object. Here Log2 is performed after adding a pseudocount of 1.

Parameters **sc** (*SingleCell*) – The SingleCell object containing gene expression data.

Returns **sc** – The SingleCell object containing the log-transfprmed gene expression data.

Return type SingleCell

`feats.`**`MergeBatches`**(*batches*)

Merges a list of SingleCell objects storing single-cell data. This function assumes that the datasets are already integrated and the number and type of genes in all the datasets are the same. If this is not the case, then an error is generated.

Parameters **`batches`** (*`list`*) – A Python list of SingleCell objects (datasets) to merge. It is assumed that the SingleCell datasets are already integrated.

Returns A merged dataset where the number of rows (d) is equal to the number genes the datasets (which is the same for all the datasets) and the number columns (n) is the sum of the number of cells in all the datasets.

Return type SingleCell

Raises **`ValueError`** – If the number and type of genes are not the same in all the datasets.

`feats.`**`PCA`**(*sc*, *n_comp=1*, *dist_or_kernel='linear'*)

Computes and stores the Principal Components of the gene expression data stored in the SingleCell object.

Parameters

- **`sc`** (*`SingleCell`*) – The SingleCell object containing gene expression data.
- **`n_comp`** (*`int, optional`*) – The number of Principal Components to compute. Default 1.
- **`dist_or_kernel`** (*`str, optional`*) – The distance metric or the kernel to compute. If a distance metric is passed, it computes the pairwise distance between the cells and then converts the distance metrics to kernels. If a kernel is passed, it computes the kernel. Valid values are ‘linear’ (default) for linear kernel, ‘spearmans’ for Spearmans distance, ‘euclidean’ for Euclidean distance and ‘pearsons’ for Pearsons distance.

Returns **sc** – The SingleCell object containing the dimensionnality reduced gene expression data. The reduced dimensionality is n_comp. The gene names are removed and the features in the reduced space are named ‘PC1’, ‘PC2’ and so on.

Return type SingleCell

CHAPTER TWO

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

INDEX