decibel

Release 1.0

Olga Ibañez-Solé and Alex M. Ascensión

Sep 19, 2022

CONTENTS

Measuring transcriptional noise in single-cell data. Check our manuscript Lack of evidence for increased transcriptional noise in aged tissues

CHAPTER ONE

INSTALLATION GUIDE

If you want to install the latest development version you can do it by cloning the repository:

```
git clone https://gitlab.com/olgaibanez/decibel.git
```

CHAPTER TWO

TUTORIAL

In order to use `decibel` you need a basic knowledge of how `scanpy` and `annData` objects work.

2.1 Computing transcriptional noise

In order to work with decibel you need to first run `scanpy` and load a dataset in an `annData` object.

```
adata = sc.read('path_to_file/filename.h5ad')
```

Then process data (normalization, log-transformation, QC filtering on cells and genes).

```
sc.pp.normalize_total(adata, target_sum=1e4)
sc.pp.log1p(adata)
sc.pp.filter_genes(adata, min_cells=3)
sc.pp.filter_cells(adata, min_genes=100)
```

Run PCA, feature selection (triku), batch effect correction (harmony) and dimensionality reduction (UMAP).

```
sc.pp.pca(adata)
sc.pp.neighbors(adata)
tk.tl.triku(adata)
sc.pp.pca(adata)
sce.pp.harmony_integrate(adata, 'batch')
sc.pp.neighbors(adata, use_rep='X_pca_harmony')
sc.tl.umap(adata)
```

Compute transcriptional noise as in Enge et al, (2017).

```
dcb.enge_transcriptional_noise(adata, 'batch')
```

CHAPTER

THREE

API

3.1 Decibel functions

`module.decibel.`**`enge_transcriptional_noise`**`(`*adata*, *batch*`)`

Compute the transcriptional noise as the biological variation over the technical variation. It can only be computed in datasets with ERCC spike-ins. The biological variation is computed as the correlation distance between each cell and the average gene expression of all the cells of the same cell type and the same batch. The technical variation is computed as the correlation distance between each cell and the mean ERCC spike-in expression of all the cell of the same cell type and batch. The transcriptional noise is computed as the biological variation over the technical variation.

Parameters

- **`adata`** (`annData`) – annData object with gene expression data. It must contain a slot with the batch identity of each cell in adata.obs
- **`batch`** (`str`) – batch label (Examples: 'donor', 'patient', 'mouse')

Returns

adata – annData object with gene expression data. Euclidean distances are stored in adata.obs['cordist_bio'], adata.obs['cordist_tech'] and adata.obs['noise']

Return type

`annData`

`module.decibel.`**`distance_to_celltype_mean`**`(`*adata*, *batch*`)`

Compute the distance between each cell and the mean expression of its cell type in the same batch (donor/mouse). It computes three distances: euclidean, correlation and manhattan.

Parameters

- **`adata`** (`annData`) – annData object with gene expression data. It must contain a slot with the batch identity of each cell in adata.obs
- **`batch`** (`str`) – batch label (Examples: 'donor', 'patient', 'mouse')

Returns

adata – annData object with gene expression data. Euclidean distances are stored in adata.obs['cor_dist'], adata.obs['euc_dist'] and adata.obs['man_dist']

Return type

`annData`

`module.decibel.`**`enge_euclidean_dist`**`(`*adata*`)`

Compute the Euclidean distance to the average expression across cell types using a set of invariant genes. The invariant genes are selected as follows: 1) Create equally sized bins of genes according to their mean expression

2) Discard the two most extreme bins (lowest and highest mean expression) 3) Select the 10% with the lowest coefficient of variation within each of the remaining bins

Parameters
`adata` (`annData`) – annData object with gene expression data.

Returns
adata – annData object with gene expression data. Euclidean distances are stored in adata.obs['euc_dist_tissue_invar']

Return type
`annData`

`module.decibel.`**`pairwise_euclidean_sample`**(*adata*, *cell_type*, *n*)

`module.decibel.`**`hernando_herraez`**(*adata*, *batch*)

Computes the correlation distance of each cell to the cell type median using the 500 most variably expressed genes.

Parameters

- **adata** (`annData`) – annData object with gene expression data. Cell type annotations must be stored in adata.obs['cell_type']
- **cell_type** (`str`) – cell type label
- **batch** (`str`) – batch label (Examples: 'donor', 'patient', 'mouse')

Returns
adata – annData object with gene expression data. Euclidean distances are stored in adata.obs['cor_dist_median']

Return type
`annData`

`module.decibel.`**`distance_to_celltype_mean_invariant`**(*adata*, *batch*)

Compute the distance between each cell and the mean expression of its cell type in the same batch (donor/mouse), using a set of invariant genes as in Enge (2017). It computes three distances: euclidean, correlation and manhattan.

Parameters

- **adata** (`annData`) – annData object with gene expression data. It must contain a slot with the batch identity of each cell in adata.obs
- **batch** (`str`) – batch label (Examples: 'donor', 'patient', 'mouse')

Returns
adata – annData object with gene expression data. Euclidean distances are stored in adata.obs['cor_dist_invar'], adata.obs['euc_dist_invar'] and adata.obs['man_dist_invar']

Return type
`annData`

`module.decibel.`**`gcl`**(*adata*, *num_divisions*)

Following the original GCL.m script provided by the authors (https://github.com/guy531/gcl).

`module.decibel.`**`gcl_per_cell_type_and_batch`**(*adata*, *num_divisions*, *batch*)

Compute GCL for each cell type and batch in adata.obs['batch'].

Parameters

- **adata** (`annData`) – annData object with gene expression data. It must contain a slot with the batch identity of each cell in adata.obs
- **num_divisions** (*int*) – number of iterations to use in gcl()

Returns
output – Pandas dataframe with the GCL per cell type x batch x iteration

Return type
`pd.DataFrame`

`module.decibel.`**`rerun_preprocessing`**`(`*adata*, *batch_key*`)`

Re-runs preprocessing steps: filter lowly expressed genes, compute HVGs, run batch-effect corrected PCA (harmony), neighbors.

Parameters

- **adata** (`annData`) – annData object with gene expression data. It must contain a slot with the batch identity of each cell in adata.obs
- **batch_key** (`str`) – batch label to run batch-effect correction (Examples: ‘donor’, ‘patient’, ‘mouse’)

Returns
adata – updated annData object with gene expression data.

Return type
`annData`

`module.decibel.`**`scallop_pipeline`**`(`*adata*, *res_vals=None*`)`

Compute transcriptional noise as 1 - membership score (averaged over aa range of resolution values). It runs the whole Scallop pipeline: 1) Create separate annData object per condition (young/old, smoker/non-smoker) and cell type. 2) Re-run preprocessing on annData 3) Run Scallop over range of resolution values 4) Compute average membership score across resolutions 5) Transcriptional noise = 1 - mean membership score

Parameters

- **adata** (`annData`) – annData object with gene expression data. It must contain a slot with the batch identity of each cell in adata.obs
- **batch** (`str`) – batch label (Examples: ‘donor’, ‘patient’, ‘mouse’)

Returns
adata – annData object with gene expression data. Euclidean distances are stored in adata.obs[‘cor_dist_invar’], adata.obs[‘euc_dist_invar’] and adata.obs[‘man_dist_invar’]

Return type
`annData`

CHAPTER

FOUR

CHANGELOG

4.1 v1.0

Functions included in decibel:

- `` `enge_transcriptional_noise` ``
- `` `distance_to_celltype_mean` ``
- `` `enge_euclidean_dist` ``
- `` `pairwise_euclidean_sample` ``
- `` `hernando_herraez` ``
- `` `distance_to_celltype_mean_invariant` ``
- `` `gcl` ``
- `` `gcl_per_cell_type_and_batch` ``
- `` `rerun_preprocessing` ``
- `` `scallop_pipeline` ``

CHAPTER

FIVE

LICENSE

INDEX