Cell society Tomography v1.0
26th,Jan,2018 by An Chengrui in Zhejiang University

1 Introduction and requirement

1.1 Introduction

Cell society Tomography (CsT) is an overlapped cluster analyze of single cell RNA-seq or any other high-dimensional data fitting Poisson's distribution. This tool considers parameters with different variation separately rather than normalize them into unique variation, which like CT scanning every slice of human body. All the potential cluster will be recorded as probability graph for father biological testimony.

1.2 Running Environment

CsT v1.0 can only run in MATLAB environment, no matter in Windows and Linux. But Windows is recomanded because it will show the waitbar with this version. Linux version with waitbar will be released soon and

Because it needs lots of calculation, a server or workstation with CPUs with more than 15 threads and more than 32GB RAM is recommanded for small samples. If your data with more than 2000, please use super calculation center or cloud serve to deal with the task.

1.3 Suitable data

1:single cell RNA-seq or any other high-dimensional data fitting Poisson's distribution
2:the feature number of samples fit to normal distribution
3:Deep sequencing (average feature number is more than 4000)

1.4 Package installation

Download all files in https://github.com/purewaltzan/Cell-society-Tomography/. Set path to this folder in MATLAB

2 User guide

2.1 Data prepareretion

CsT support .csv, .tsv, .xls or any other format with constant seperator whose string is coded by ASCII and number is coded by ASCII, double, single or Int.

The table must be according to the formation below (for exmple .csv):

labelname1,label1.1,label1.2,...
labelname2,label2.1,label2.2,...
...
gene,cellname1,cellname2,...
genename1,data1.1,data1.2,...
genename2,data2.1,data2.2,...
....

Where block of 'gene' is very important for CsT to seperate labels and data. No capital here. Table also can start with 'gene' if there is no labels for every cells.

2.2 Parameter initiation

```
> CsTc = CsT_InitialCsTc(filename,species);
```

2.3 Import data

```
> CsTc = CsT_dataimport(CsTc);
```

After import data, table of gene expressions and counts will be stored in `CsTc.data`, gene list will be stored in `CsTc.gene`, name of samples will be stored in `CsTc.Cellid`, labels of samples will be stored in `CsTc.label`

2.4 Read gene annotation

```
> CsTc = CsT_Readannotation(CsTc);
```

CsT insert a program to help researchers select genes according to their annotations. The annotation contains symbols, transcriptionIDs, genomeIDs, descriptions of genes from Ensmbl database.

After CsT reads the annotation, some parameters are generated to control the normalization. They are:

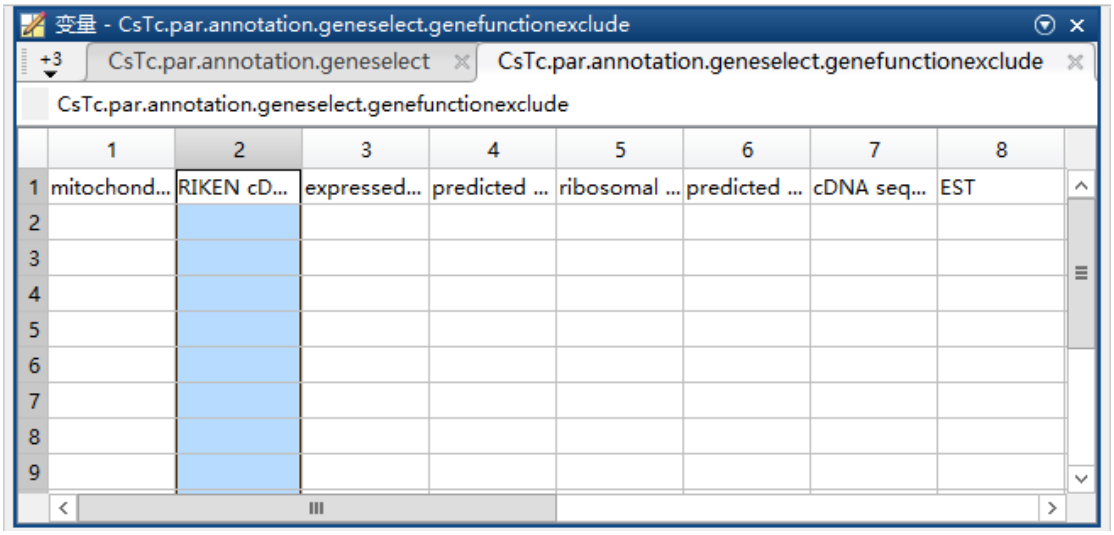
`CsTc.par.annotation.geneselect.transcripttype`, to specify the transcript type of interesting genes.

`CsTc.par.annotation.geneselect.source`, to specify the source of interesting genes.

`CsTc.par.annotation.geneselect.genefunctionexclude`, to specify the genes with the function should be excluded.

There are two ways to change these parameters.

The first way is to delete the corresponding block in variable window directly.



The second way is to delete the corresponding contends with command, such as:

```
>CsTc.par.annotation.geneselect.transcripttype=CsTc.par.annotation.geneselect.transcripttype(31);
>CsTc.par.annotation.geneselect.source=CsTc.par.annotation.geneselect.source(3);
```

2.5 Data Normalization

Firstly, normalization parameters must be specified.

```
> CsTc.par.Normalize.maxcounts=inf; %%Numeric, default inf
> CsTc.par.Normalize.mincounts=-inf; %%Numeric, default -inf
> CsTc.par.Normalize.maxgenes=inf; %%Numeric, default inf
> CsTc.par.Normalize.mingenes=-inf; %%Numeric, default -inf
> CsTc.par.Normalize.Type='CPM' %%String('CPM','TPM'). default 'CPM'
> CsTc.par.Normalize.maskr=[]; %%[a,b], select genes whose CPM or TPM larger than a in more than b cells. Default, [2,2]
> CsTc.par.Normalize.transform='Null'; %%log transform of expression matrix. ('Null','log2','log10'), default 'Null'
```

After set normalization parameters, run command:

```
> CsTc = CsT_Normalization(CsTc);
```

If another normalization methods are needed, original counts matrix is stored in `CsTc.data`, counts matrix of filtered gene only with annotation is stored in `CsTc.main.data`, counts matrix of final filtered gene is stored in `CsTc.main.N.counts`.

Because the output of Normalization is CsTc.main.N.data, it can be replace if you want to employ another normalization methods. In this case, please make sure the dimension of `CsTc.main.N.data`, `CsTc.main.N.gene`, `CsTc.main.N.Cellid`, and `CsTc.main.N.label` are the same.

2.6 CsT parameter initiation

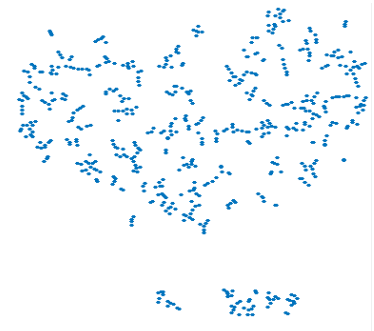
Because annotation occupies abundance of memory and parameters should be set for further analysis, CsT parameter must be initiated secondly with the command:

```
> CsTc = CsT_Deletefields1(CsTc);
```

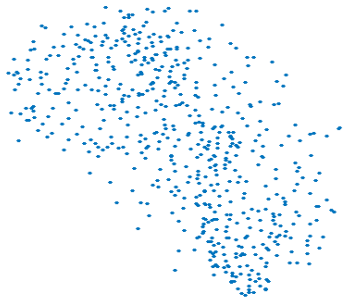
Perplex and initiate dimension are the key parameter may affect the results. Before CsT analyse, you should makesure t-sne work well.

```
> tsner = tsne(CsTc.data',[],2, CsTc.par.slice.ini_dim,…
CsTc.par.slice.perplex);
> scatter(tsner(:,1),tsner(:,2));
```

If the points are too scattered like the figure blow, increase the `CsTc.par.slice.ini_dim`, or `CsTc.par.slice.perplex`.



If the points cannot divided into different clusters like the figure blow, decrease the `CsTc.par.slice.ini_dim`, or `CsTc.par.slice.perplex`.



They can be changed with command:

```
> CsTc.par.slice.display.ini_num=30; %%Numeric, default 30
> CsTc.par.slice.display.perplex=20; %%Numeric, default 20
```

2.7 CsT analyze

It will take very long time

```
> CsTc = CsT_Slicetsnetransform(CsTc);
> CsTc = CsT_Innerslicedistence(CsTc);
> CsTc = CsT_Interslicedistence(CsTc);
> CsTc.par.slice.display.dimension=3; %%Numeric,(2,3), default 2, set the dimension of scatter plot
> CsTc = CsT_Reducedim(CsTc);
```

2.8 Represented slice detection

Because there are lots of slices in CsT analyze results, only some distinguished slices are needed to show the potential key clusters.

```
> CsTc = CsT_Clusterlayer(CsTc);
```

Some exist marker gene can be used to test whether the detection is perfect. You can use the command below to show the distribution of this marker.

```
> CsT_Showgene(CsTc,Marker); %% Marker is a string existed in CsTc.gene such as 'Acan'
```

If this marker is not gather in all slices, increase `CsTc.par.slice.sliceclusternum`. The number of selected slices may be equal to `CsTc.par.slice.sliceclusternum + 1` in some situation. Small `CsTc.par.slice.sliceclusternum` is recommended because it will simplify the results of CsT

2.9 Cluster cells in every slice

Firstly, the number of selected slices should be assured with the command.

```
> numel(CsTc.slice.display)
```

Then cluster cells in every slice one by one. Three cluster algorithms have been built in CsT, which are Hierarchy Cluster, K-means and Density peaks cluster (recommanded).

For a specific slice n, the commands are:

```
> CM=CsT_initialCM
```

```
> CM.methods='DBCA'; %% 'kmeans' for Kmeans, 'HCluster' for Hierarchy Cluster,'DBCA' for Density peaks cluster
> CM.number=8;
> CM.DBCA.dc=3;
> CsTc = CsT_Cellcluster(CsTc,sliceID,CM);
```

After cluster, the cluster result will be showed in window. If it is not satisfied, change `CM.number` and `CM.DBCA.dc` and execute `CsT_Cellcluster` again.

The principles of setting cluster parameter described below according to their importance.

1. Make sure all isolated clusters are discriminated
2. Decrease the cluster number as you can
3. Make sure all the turbulence relevant to markergenes are discriminated

2.10 Find the markers for every cluster

```
> CsTc.par.display.markernum=10; %% number of marker genes for every cluster
> load mappp %% load colormap for Heatmap, (default: Yellow/Blue)
> CsTc.par.display.colormap=mappp; %% Initiate colormap, strings represented colormap is supported ('jet','hsv', et.)
> CsTc = CsT_Markergene(CsTc);
```

2.11 Reconstruct cell society

```
> CsTc.par.graph.minlinkage=0.5; %% threshold to display the linkage between clusters
> CsTc.par.graph.degree=[10,3]; %% filter clusters with its profile of its degree
> CsTc.par.graph.outputname='graph'; %% filename of output
> CsTc = CsT_Graph(CsTc);
```

The output file is an .xlsx file whose format is suitable for CytoScape.