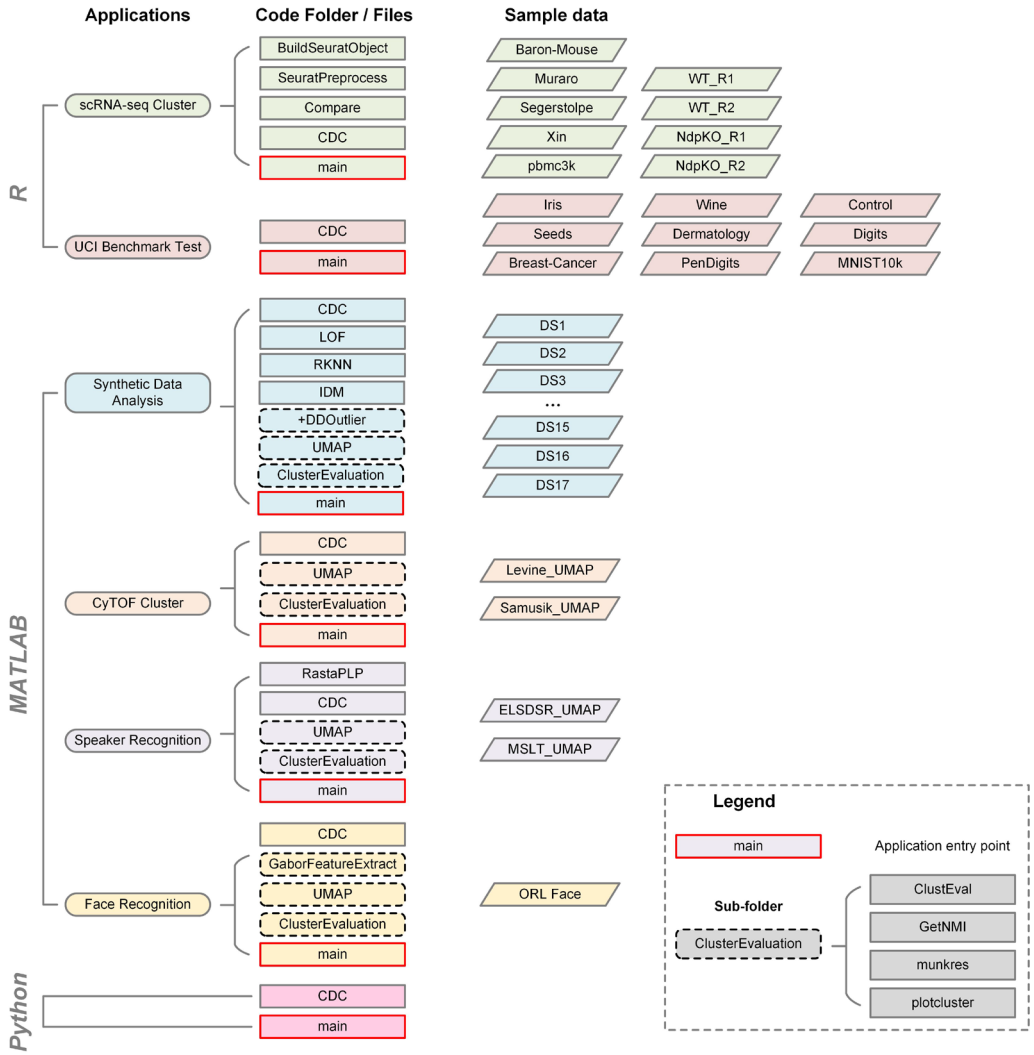
Tutorial

Introduction

We propose a novel Clustering algorithm by measuring Direction Centrality (CDC) locally. It adopts a density-independent metric based on the distribution of K-nearest neighbors (KNNs) to distinguish between internal and boundary points. The boundary points generate enclosed cages to bind the connections of internal points, thereby preventing cross-cluster connections and separating weakly-connected clusters.

This is a toolkit for CDC cluster analysis on various applications, including 'scRNA-seq Cluster', 'UCI Benchmark Test', 'Synthetic Data Analysis', 'CyTOF Cluster', 'Speaker Recognition', 'Face Recognition'. They are implemented using *MATLAB*, *R* and *Python* languages. Each application is assembled in a separate folder. You can use the application by running the 'main' file in its root directory. The framework of this toolkit is illustrated below:



Although this figure shows the developing language for each assembled application, it not necessarily

indicates that a certain application can be only conducted using the illustrated languages. Actually, we provide the core code of CDC in *MATLAB*, *R* and *Python* exclusively. So that, you can select any of the three languages to customize the applications (e.g., conducting scRNA-seq cluster in *MATLAB*) or create your own applications (e.g., analyzing spatiotemporal point patterns for geographical objects and events, such as lakes and crimes, using *Python*). To achieve that, you need to find or program the supplementary codes for data preprocessing, loading, format transformation and etc. Otherwise, you can just simply use the preprocessed data in other languages and tools as the inputs.

scRNA-seq Cluster

Workflow

This application supports the CDC cluster analysis for scRNA-seq datasets and is implemented in R language. Seurat is used to preprocess the scRNA-seq and its standard pipeline can be seen in https://satijalab.org/seurat/index.html. The entire workflow of CDC cluster analysis contains the following four steps:

- Step 1: Convert the raw data to standard Seurat object.
- Step 2: Preprocess the data using standard Seurat pipeline, including *normalization, HVG selection, scaling and PCA&UMAP embedding*.
- Step 3: Perform CDC on the embedded data.
- Step 4: Visualize and evaluate clustering results, as well as compare performances with integrated baselines (*Seurat, SNN-Walktrap, SNN-Louvain, K-means*).

Depends

- R (≥4.1.0)
- (*optional*) RStudio (≥1.3.1093)

Directory

Please click Tools->Global Options->General, change the working directory to that of our toolkit folder.

Imports

argparse (≥2.0.4), assertthat (≥0.2.1), BiocGenerics (≥0.40.0), BiocSingular (≥1.10.0), ClusterR (≥1.2.5), dotCall64 (≥1.0.1), fields (≥12.5), GenomeInfoDb (≥1.30.1), GenomicRanges (≥1.46.1), geometry (≥0.4.5), ggplot2 (≥3.3.5), grid (≥4.1.0), gtools (≥3.9.2), IRanges (≥2.28.0), MatrixGenerics (≥1.6.0), mclust (≥5.4.7), parallel (≥4.1.0), prodlim (≥2019.11.13), RcppHungarian (≥0.1), readr (≥1.4.0), reshape2 (≥1.4.4), S4Vectors (≥0.30.0), scran (≥1.22.1), scuttle (≥1.4.0), Seurat (≥4.0.5), SingleCellExperiment (≥1.16.0), spam (≥2.7.0), stats4 (≥4.1.0), SummarizedExperiment (≥1.24.0), uwot (≥0.1.10)

Noted: all R packages can be installed from the *CRAN repository* or *Bioconductor*. You can also use the following R scripts to install them all at once:

```
## Please click Tools->Global Options->Packages, change CRAN repository to a near mirror. Then, execute the following code:
## Install packages from CRAN.
install.packages(c("argparse", "assertthat", "ClusterR", "dotCall64", "fields", "geometry", "ggplot2", "gtools", "mclust", "prodlim",
```

```
"RcppHungarian", "readr", "reshape2", "Seurat", "spam", "uwot"))
## Determine whether the package "BiocManager" exists, if not, install this package.
if (!require("BiocManager", quietly = TRUE))
    install.packages("BiocManager")
## Install packages from Bioconductor.
BiocManager::install(c("BiocGenerics", "BiocSingular", "GenomeInfoDb", "GenomicRanges", "IRanges", "MatrixGenerics",
"S4Vectors", "scran", "scuttle", "SingleCellExperiment", "SummarizedExperiment"), force = TRUE, update = TRUE, ask = FALSE)
```

Data Format

This application supports two data formats currently, 'csv' and '10X'. Data in 'csv' format must contain a Matrix.csv file at least, which organizes the cell read counts as a matrix as follow:

	X0610007P14Rik	X0610009B22Rik	X0610009E02Rik	...
mouse1_lib1.final_cell_0001	0	0	0	...
mouse1_lib1.final_cell_0002	2	0	0	...
mouse1_lib1.final_cell_0003	0	0	1	...
...	...	...	...	...

The first column records all cell names (IDs) or barcodes, that means each row represents the expression level in all genes of a cell. While, the first row records all gene names (IDs) and each column denotes the expression level in a gene of all cells.

Data in standard '10X' format need to contain three files, 'barcodes.tsv', 'genes.tsv', and 'matrix.MTX' at least. 'barcodes.tsv' includes the all barcodes or cell names (IDs), and 'genes.tsv' includes all gene names (IDs). 'matrix.MTX' stores all non-zero elements and organizes the matrix in a vector of [No_gene, No_cell, count].

barcodes.tsv	genes.tsv	matrix.MTX
AAACATACAACCAC-1	ENSG00000243485 MIR1302-10	32738 2700 2286884
AAACATTGAGCTAC-1	ENSG00000237613 FAM138A	32709 1 4
AAACATTGATCAGC-1	ENSG00000186092 OR4F5	32707 1 1
...	...	...

It should be noted that all three filenames must be consistent with the above. In addition, the 'Labels.csv' is also required if you want to evaluate the final clustering accuracy quantitatively, otherwise the ARI score cannot be calculated successfully. The 'Labels.csv' has only one column and the column name must be 'x'.

x
beta
ductal
delta
schwann
...

In this toolkit, we provide **9** scRNA-seq sample datasets, 'Baron-Mouse', 'Muraro', 'Segerstolpe', 'Xin', 'pbmc3k', 'WT_R1', 'WT_R1', 'NdpKO_R1', and 'NdpKO_R2'. The first four datasets are in 'csv' format, while the last five are in '10X' format. They all contain the true cell label files (denoted as '1', otherwise '0').

Dataset	Format	Label
'Baron-Mouse'	'csv'	'1'
'Muraro'	'csv'	'1'
'Segerstolpe'	'csv'	'1'
'Xin'	'csv'	'1'
'pbmc3k'	'10X'	'1'
'WT_R1'	'10X'	'1'
'WT_R2'	'10X'	'1'
'NdpKO_R1'	'10X'	'1'
'NdpKO_R2'	'10X'	'1'

Taking the file volume into account, we only attach the above sample datasets to this toolkit. The rest datasets and their labels used in our paper can be downloaded from the following links.

Dataset	Link
Baron-Human (BH), AMB, TM	https://zenodo.org/record/2877646#.YhCJs99ByUm
ALM	https://portal.brain-map.org/atlases-and-data/rnaseq/mouse-v1-and-alm-smart-seq
VISp	https://portal.brain-map.org/atlases-and-data/rnaseq/mouse-v1-and-alm-smart-seq
MIHPF	https://portal.brain-map.org/atlases-and-data/rnaseq/mouse-whole-cortex-and-hippocampus-10x

Arguments

The arguments of CDC function in R

dat_mat	A n×m data matrix, containing n samples with m features
k	k of KNN (Default: 30, Recommended: 30~50)
ratio	A percentile ratio of internal points (Default: 0.9, Recommended: 0.75~0.95)
embeding_method	The embedding method (Default: "None", Recommended: "None" or "UMAP")
k_UMAP	The number of neighboring points for UMAP (Default: 30, Recommended: 5~50)
npc	The dimension of UMAP space to embed into (Default: 2, Recommended: 2~5)
norm	Whether to normalize the data or not (Default: FALSE, Recommended: FALSE or TRUE)

How to Run

- **Specify the data file name**

Open the 'main.R' file in the root directory of 'scRNA-seq Cluster' file. First, please specify the name and format of the scRNA-seq data and determine to read to label file or not. You must name and organize the files as the description in **Data Format**. ***To be noted, the sample datasets have been compressed into .zip files due the data size limit of GitHub. Before using them, please decompress them into***

the corresponding data folders named by the datasets.

```
## Specify the filename, format and labels.
## --Arguments--
## --Arguments--
##       filename: filename of the scRNA-seq dataset (We provide nine sample datasets in this application)
##       format: data format of the scRNA-seq dataset (We currently support two data formats, csv and 10X)
##       labels:    whether the file contains the true label of cells ('1' is Yes, '0' is No)
filename = 'Baron-Mouse
format = 'csv'
labels = '1'
```

- **Build Seurat object**

Then, start to read the data files and convert the raw data to the standard Seurat object

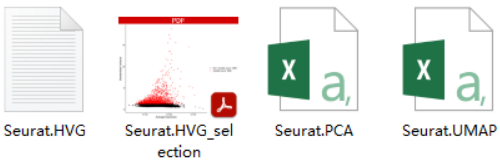
```
## Generate Seurat Object from the raw data
source('BuildSeuratObject.R')
SeuratData <- BuildSeuratObject(filename, format, labels)
```

- **Preprocess**

After that, please preprocess the data using standard Seurat pipeline, including Normalization, HVG Selection, Scaling and PCA&UMAP Embedding. The details can be seen at Seurat Website. By default, the UMAP dimension is specified as 2, and we recommend you to set it as 2~5 for subsequent CDC clustering.

```
## Preprocess scRNA-seq using standard Seurat pipeline
## --Arguments--
##      n_components: The dimension of UMAP space to embed into (Recommended: 2~5)
source('SeuratPreprocess.R')
n_components = 2
SeuratData <- SeuratPreprocess(SeuratData, filename, n_components)
```

In this step, the toolkit will write the intermediate results of preprocessing into four files: 'Seurat.HVG', 'Seurat.HVG_selection', 'Seurat.PCA', 'Seurat.UMAP' that store the selected HVGs, and the dimension reduction results.



- **Cluster the cells**

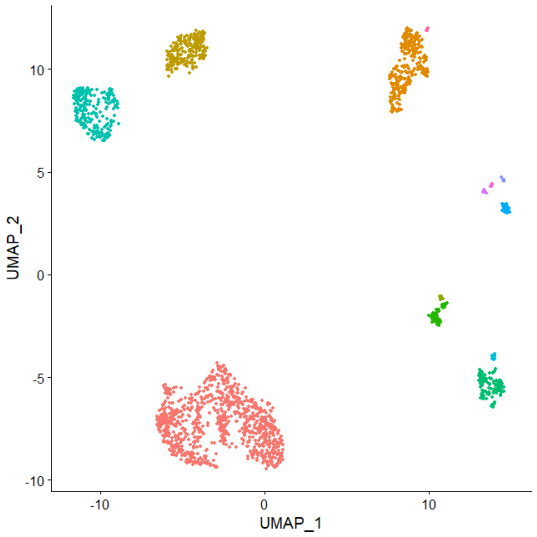
This toolkit adopts CDC to cluster the cells, which has two parameters, *k* and *ratio*. We input the embedded

UMAP data to CDC, and the clusters can be found using the Idents() function. We specify the default values as 30 and 0.9 for *k* and *ratio* respectively. You can also adjust them for searching best results in exploratory analysis. It should be noted that default parameter settings are empirical and can only produce a rational result, but cannot guarantee optimal clustering results on all datasets. In order to obtain optimal clustering result, tuning the parameter settings and traversing parameter space are needed.

```
## Cluster the cells using CDC algorithm
## --Arguments--
##       k: k of KNN (Default: 30, Recommended: 5~50)
##       ratio: percentile ratio of internal points (Default: 0.9, Recommended: 0.70~0.95)
source('CDC.R')
k = 30
ratio = 0.9
Idents(SeuratData) <- CDC(SeuratData@reductions[["umap"]]@cell.embeddings, k, ratio)
```

- **Visualize and evaluate**

We provide code to visualize the clustering results using 2D UMAP data. You can set the point size in the pt.size attribute. The true cell labels can be found in SeuratData@meta.data[["Cluster"]].



ARI = 0.964 (UMAP_Dim = 2, k = 30, ratio = 0.9)

- **Compare with baselines**

We also integrate four clustering baselines for comparison, Seurat, SNN-Walktrap, SNN-Louvain, K-means. Seurat was performed on the 20~50 PCs; while SNN-Walktrap, SNN-Louvain, and K-means are conducted both on 50 PCs and low-dimensional UMAPs (depends on the argument UMAP_Dim). You can also add other clustering algorithms in 'Compare.R' file, and customize the parameter space.

```
## Compare the clustering performance with Seurat, SNN-Walktrap, SNN-Louvain, K-means
## Noted: Comparison is usually time consuming, if you just want to run CDC algorithm and don't want to compare with other
```

```
## algorithms by traversing different settings in the parameter spaces, please just comments out this part of the code.
## --Arguments--
##       seurat_dim: dimension of reduction to use as input in Seurat (Default:  seq(20, 50, 5))
##       seurat_resolution: resolution of Louvain algorithm in Seurat (Default: seq(0.1, 1, 0.1))
##       snnwalk_k: k of SNN graph in SNN-Walktrap (Default: seq(5, 30, 5))
##       snnlouv_k: k of SNN graph in SNN-Louvain (Default: seq(5, 30, 5))
##       kmeans_k: k of K-means (Default: seq(2, 50))
##       CDC_k: k of KNN in CDC (Default: seq(30, 50, 10))
##       CDC_ratio: ratio of CDC (Default: seq(0.75, 0.95, 0.02))
source('Compare.R')
box_res <- Compare(SeuratData, filename, seurat_dim, seurat_resolution, snnwalk_k, snnlouv_k, kmeans_k, CDC_k, CDC_ratio)
boxplot(box_res, ylab='ARI', col=c('#FF69B4', '#4682B4', '#3CB371', '#FF7F50','#008B8B','#FFD700','#FF0000','#1E90FF'))
```

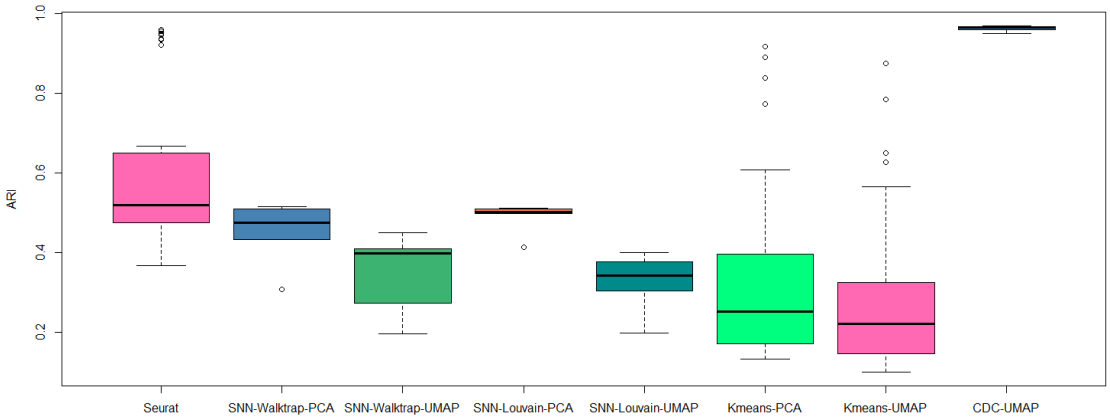
During the comparing analysis, the console will constantly update the running status and ARI scores.

```
-------------------------------------------
Running Seurat algorithm...
-------------------------------------------
The number of times Seurat ran: 70
Overall runtime of Seurat: 17.27s
Average runtime of Seurat: 0.247s
Max/Average ARI of Seurat: 0.959/0.565
-------------------------------------------
Running SNN-Walktrap algorithm...
-------------------------------------------
The number of times SNN-Walktrap ran: 12
Overall runtime of SNN-Walktrap: 10.7s
Average runtime of SNN-Walktrap: 0.892s
Max/Average ARI of SNN-Walktrap-PCA: 0.516/0.453
Max/Average ARI of SNN-Walktrap-UMAP: 0.45/0.354
-------------------------------------------
Running SNN-Louvain algorithm...
-------------------------------------------
The number of times SNN-Louvain ran: 12
Overall runtime of SNN-Louvain: 1.92s
Average runtime of SNN-Louvain: 0.16s
Max/Average ARI of SNN-Louvain-PCA: 0.512/0.49
Max/Average ARI of SNN-Louvain-UMAP: 0.401/0.327
-------------------------------------------
Running K-means algorithm...
-------------------------------------------
The number of times K-means ran: 98
Overall runtime of K-means: 0.93s
Average runtime of K-means: 0.009s
Max/Average ARI of Kmeans-PCA: 0.91/0.325
Max/Average ARI of Kmeans-UMAP: 0.911/0.267
-------------------------------------------
Running CDC algorithm...
-------------------------------------------
The number of times CDC ran: 33
Overall runtime of CDC: 37s
Average runtime of CDC: 1.121s
Max/Average/Default ARI of CDC-UMAP: 0.968/0.962/0.964
-------------------------------------------
Comparison complete!
Overall elapsed time: 67.82s
```

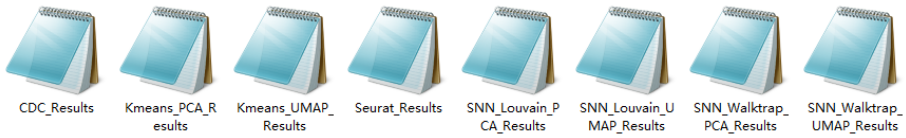
Noted: the UMAP results are slightly different in Windows and macOS operating systems, which is probably because UMAP is a stochastic algorithm and makes use of randomness to speed up

approximation steps. It results in slightly different clustering accuracies (The average ARI scores of our method on dataset Baron-Mouse on Windows and macOS are 0.968 and 0.963 respectively).

The ARI scores of all clustering algorithms will be output in box_res, and a box plot will be generated to view the comparing result.



In addition, all clustering results will be written into eights .tsv files and stored in the same file.



UCI Benchmark Test

This application supports for cluster analysis on high-dimensional UCI datasets, such as 'Iris', 'Wine', 'Control', 'Seeds', 'Dermatology', 'Digits', 'PenDigits', 'MNIST10K', 'Breast-Cancer'. These datasets can be used as standard benchmarks to test and compare the performance of the clustering algorithms.

Depends

- R (⩾4.1.0)
- (*optional*) RStudio (⩾1.3.1093)

Imports

- argparse (⩾2.0.4), assertthat (⩾0.2.1), ClusterR (⩾1.2.5), dotCall64 (⩾1.0.1), fields (⩾12.5), geometry (⩾0.4.5), grid (⩾4.1.0), gtools (⩾3.9.2), mclust (⩾5.4.7), prodlim (⩾2019.11.13), RcppHungarian (⩾0.1), readr (⩾1.4.0), reshape2 (⩾1.4.4), spam (⩾2.7.0), uwot (⩾0.1.10)

How to Run

- **Read the UCI benchmarks**

Open the main.R file in the root directory of the folder 'UCI Benchmark Test'. The UCI benchmarks we used are in format of .txt and .csv. For each benchmark, the features and labels have been integrated in

the same file, where each row represents an instance and the last column denotes the labels.

```
path <- getwd()
setwd(path)
## Read UCI data (Iris, Seeds, Wine and Digits are txt files, others are in csv format)
data <- read.table('./UCI benchmarks/Iris.txt', header = FALSE, sep = '\t')
# data <- read.csv('./UCI benchmarks/Dermatology.csv',header = FALSE)
dat_mat <- as.matrix(data[, 1:(ncol(data)-1)])
dat_mat <- dat_mat[,colSums(dat_mat)>0]
dat_label <- unlist(data[,ncol(data)])
```

- **Normalize the data and reduce the dimensions**

We normalize the data using max-min method and use UMAP to embed the normalized data to low-dimensional space.

```
## Normalize the data
norm_dat_mat <- dat_mat
for (i in 1:ncol(dat_mat)){
  if ((max(dat_mat[,i])-min(dat_mat[,i]))>0){
    norm_dat_mat[,i]<-(dat_mat[,i]-min(dat_mat[,i]))/(max(dat_mat[,i])-min(dat_mat[,i]))
  }
}
## UMAP Embedding
## --Arguments--
##       n_neighbors: The number of neighboring points for UMAP (Recommended: 5~50)
##       n_components: The dimension of the space to embed into using UMAP (Recommended: 2~5)
n_neighbors = 25
n_components = 4
set.seed(142)
umap_dat <- uwot::umap(norm_dat_mat, n_neighbors, n_components)
```

- **Clustering**

Then, you can conduct CDC algorithm to cluster the benchmark.

```
## Run CDC algorithm
## --Arguments--
##       k: k of KNN (Default: 30, Recommended: 5~50)
##       ratio: percentile ratio of internal points (Default: 0.9, Recommended: 0.70~0.95)
k = 11
ratio = 0.9
source('CDC.R')
res <- CDC(umap_dat, k, ratio)
clus <- as.integer(res)
```

- **Evaluate**

Finally, you can calculate four evaluation metrics, ARI, ACC, NMI and JI, to assess the clustering accuracy of CDC algorithm.

```
## Calculate the validity metrics (ARI, ACC, NMI, JI)
metrics <- matrix(0, nrow=1, ncol=4)
metrics[1,1] <- mclust::adjustedRandIndex(clus, dat_label)
metrics[1,2] <- ACC(dat_label, clus)
metrics[1,3] <- ClusterR::external_validation(dat_label, clus, method="nmi")
metrics[1,4] <- ClusterR::external_validation(dat_label,clus,method="jaccard_index")
cat(paste('ARI: ', round(metrics[1,1], 4),' ACC: ', round(metrics[1,2], 4),' NMI: ',round(metrics[1,3], 4),' JI: ', round(metrics[1,4], 4)))
```

The evaluation metrics will be output in the console.

```
ARI:  0.9038  ACC:  0.9667  NMI:  0.8851  JI:  0.8787
```

Synthetic Data Analysis

This application supports for cluster analysis on synthetic datasets. It contains two main files, 'main1.m' and 'main2.m'. The first handles noise-free datasets, and the second integrates noise elimination methods, *LOF*, *RKNN* and *IDM*. We provide 17 synthetic 2D datasets with different shapes of clusters in this application, where DS10-DS13 contain noise points. These datasets can help users to understand the capabilities of the different clustering algorithms under representative 2D data distributions.

Depends

- MATLAB 2020b

How to Run

- **Setup the parameters**

Open the 'main1.m' file in the root directory of 'Synthetic Data' file. Specify the two input parameters of CDC algorithm.

```
%% Specify the parameters
%    k_num: k of KNN
%    ratio: percentile ratio of internal points (Default: 0.9, Recommended: 0.70~0.95)
k_num = 30;
ratio = 0.7;
```

- **Read the data**

Input the file path of the data and use textread() function to read it. The first two columns are the two features, and the third is the true labels.

```
%% Input the data and labels
```

```
data = textread('SyntheticDatasets/DS1.txt');
[n, m] = size(data);
X = data(:,1:2);
label = data(:,3);
```

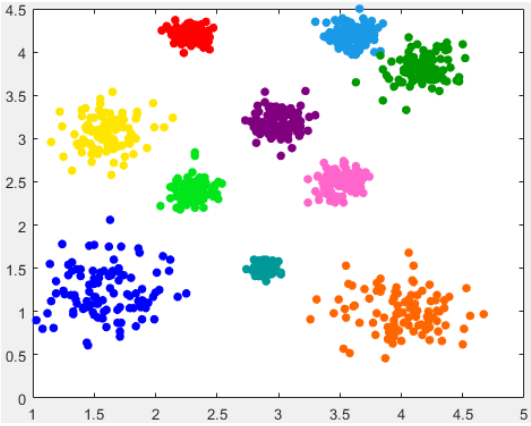
- **Clustering**

```
%% Perform CDC algorithm
cluster = CDC(X, k_num, ratio);
```

- **Evaluate and visualize**

```
%% Evaluate the clustering accuracy and plot the result
addpath ClusterEvaluation
[Accuracy, NMI, ARI, Fscore, JI, RI] = ClustEval(label, cluster);
plotcluster(X, cluster);
```

The plotcluster() function will generate a scatter plot to view the clustering results.



You can also input high-dimensional data in this application. We have integrated UMAP for embedding the data into space of low-dimensions. Before performing CDC, you should normalize the high-dimensional data and use UMAP to embed to a low-dimensional space (recommended: 2D~5D).

```
%% Normalize data and UMAP embedding
for i = 1 : length(X(1, :))
    if ((max(X(:, i))-min(X(:, i)))>0)
        X(:, i) = (X(:, i)-min(X(:, i)))/(max(X(:, i))-min(X(:, i)));
    end
end
addpath UMAP/umap
[X, ~, ~, ~] = run_umap(X, 'n_components', 2, 'min_dist', 0.1, 'n_neighbors', 20);
```

CyTOF Cluster

This application supports for cell type identification on CyTOF datasets. We provide the preprocessed and embedded CyTOF datasets in this application, 'Levine_UMAP.txt' and 'Samusik_UMAP.txt'. The original datasets can be downloaded in *FlowRepository* and the preprocessing steps can be found in *Weber et al., 2016*. We provided the embedded data using UMAP in this application. If you want to process the original CyTOF datasets, you can use UMAP to embed the CyTOF data into low-dimensional space (recommended: 2D~5D), and then input UMAPs into CDC algorithm. We have integrated the UMAP function in this application, and UMAP can also be implemented in R (uwot package), and Java environments.

Depends

- MATLAB 2020b

How to Run

Open the 'main.m' file in the root directory of the folder 'CyTOF Cluster'. Specify the filename, the columns of the features and label for the dataset to be analyzed. Then, specify the two parameters of CDC and conduct clustering. After that, remove the cells without labels from the results. Finally, evaluate the clustering accuracy using the integrated six evaluation metrics, Accuracy, ARI, F1-score, NMI, RI and JI, and visualize the results.

```
%% Input the data and labels
data = textread('Data/Samusik_UMAP.txt');
[n, m] = size(data);
X = data(:,1:2);
label = data(:,3);

%% Perform CDC algorithm
% ---Recommended Arguments---
% Levine_UMAP: k_num = 60; ratio = 0.9667;
% Samusik_UMAP: k_num = 35; ratio = 0.556;
k_num = 60;
ratio = 0.9667;
cluster = CDC(X, k_num, ratio);

%% Remove the cells without labels
nan_id = isnan(label);
X(nan_id, :) = [];
cluster(nan_id) = [];
label(nan_id) = [];

%% Evaluate the clustering accuracy and plot the result
addpath ClusterEvaluation
[Accuracy, NMI, ARI, Fscore, JI, RI] = ClustEval(label, cluster);
```

```
plotcluster(X(1:100:length(X),:), cluster(1:100:length(X),:));
```

Speaker Recognition

This application supports speaker recognition from the voice corpuses. It contains two corpuses, English Language Speech Database for Speaker Recognition (*ELSDSR*) and the Microsoft Speech Language Translation (*MSLT*) corpus datasets. ELSDSR contains English speech data collected from 22 speakers composed of 12 males and 10 females. The speech was recorded in a noise-free chamber. The volunteers read the same sentences in a declarative voice. MSLT corpus consists of tens of thousands of conversational utterances in multiple languages. The audio data were captured from the conversations held by the authorized volunteers in communication software. The contents of conversations were not predefined and the environments were random. A quantity of modal particles and emotional utterances were contained in the conversations. We selected 200 segmented audio files of the utterances of 20 speakers in Chinese including 12 males and 8 females. The voice corpuses are stored in WAV format, and we adopt RelAtive SpecTrAl-Perceptual Linear Prediction (RASTA-PLP) to extract the cepstral features from the raw waveform data. We provided the embedded data using UMAP in this application, 'ELSDSR_UMAP.txt' and 'MSLT_UMAP.txt'. If you want to process the original RASTA-PLP features, you can use UMAP to embed them into low-dimensional space (recommended: 2D~5D), and then input UMAPs into CDC algorithm. We have integrated the UMAP function in this application, and UMAP can also be implemented in R (uwot package), and Java environments.

Depends

- MATLAB 2020b

How to Run

Open the 'main.m' file in the root directory of the folder 'Speaker Recognition'. Specify the filename, the columns of the features and label for the dataset to be analyzed. Then, specify the two parameters of CDC and conduct clustering. After that, evaluate the clustering accuracy using the integrated six evaluation metrics, Accuracy, ARI, F1-score, NMI, RI and JI, and visualize the results.

```
%% Input the embedded data using UMAP
data = textread('Data/ELSDSR_UMAP.txt');
[n, m] = size(data);
X = data(:, 1:2);
label = data(:, 3);


%% Perform CDC algorithm
% Recommended Arguments
% ELSDSR_UMAP: k_num = 5; ratio = 0.7;
% MSLT_UMAP: k_num = 7; ratio = 0.53;
k_num = 5;
ratio = 0.7;
cluster = CDC(X, k_num, ratio);
```

```
%% Evaluate the clustering accuracy and plot the result
addpath ClusterEvaluation
[Accuracy, NMI, ARI, Fscore, JI, RI] = ClustEval(label, cluster);
plotcluster(X, cluster);
```

Face Recognition

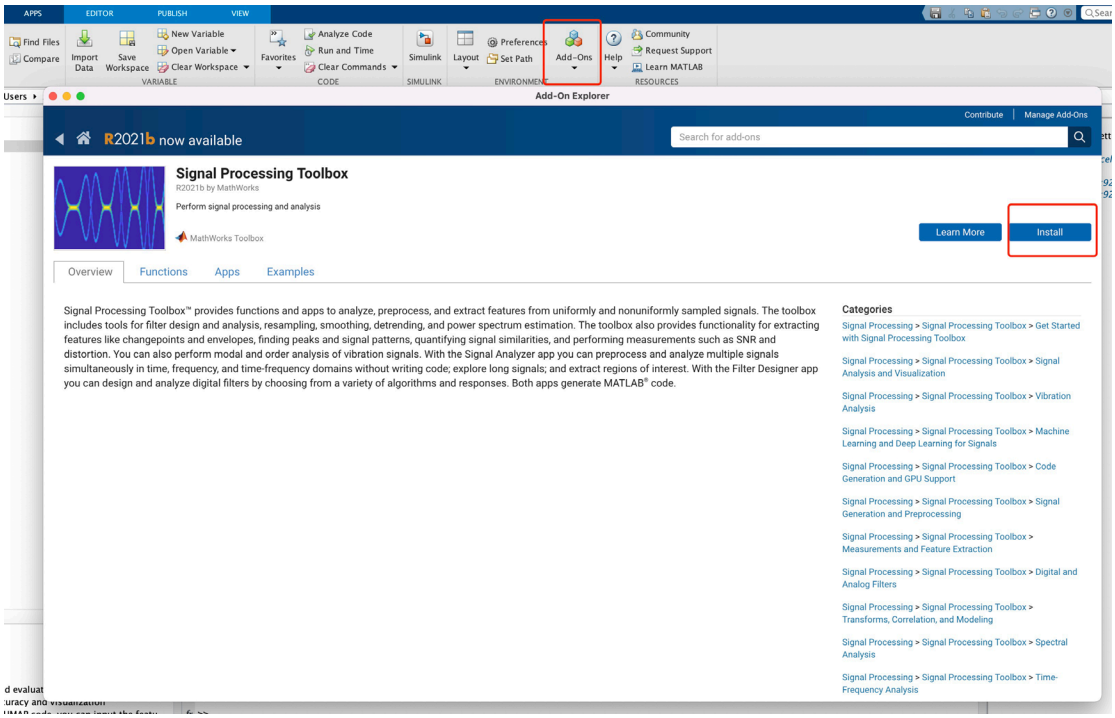
CDC can be applied to face recognition. This application provides the ORL face dataset. It contains 40 distinct individuals and each one has 10 images with different lighting, accessories, and facial expressions. All 400 face images are grayscale with 92×112 pixels in size. We extracted Gabor features as the input of clustering.

Depends

- MATLAB 2020b
- Signal Processing Toolbox

How to Run

Click 'Add-Ons' and install the 'Signal Processing Toolbox'.



Then, open the 'main.m' file in the root directory of 'Face Recognition' file. Specify the number of images and the image size in pixels, and then load these images. Then, extract Gabor feature for each image, and assign true labels. After that, specify the two parameters of CDC and conduct clustering. Finally, evaluate the clustering accuracy using the integrated six evaluation metrics, Accuracy, ARI, F1-score, NMI, RI and JI, and visualize the results.

```
%% Input ORL face images
```

```
n = 400;
imgs = zeros(112,92,n);
for i=1:40
    for j=1:10
        num1 = num2str(i);
        num2 = num2str(j);
        num = num2str(j+10*(i-1));
        file = ['Olivetti/',num1,'/',num2,'.pgm'];
        img = imread(file);
        imgs(:,:,j+10*(i-1)) = img;
    end
end

%% Extract Gabor feature
gabor = [];
for i = 1:n
    addpath GaborFeatureExtract
    gaborArray = gaborFilterBank(5,8,39,39);    % Generates the Gabor filter bank
    gabor = [gabor; (gaborFeatures(imgs(:,:,i),gaborArray,15,15))'];
end

%% Generate true labels
label = zeros(n,1);
for i=1:n
    label(i) = ceil(i/10);
end

%% Normalize the gabor features
for i=1:length(gabor(1,:))
    if((max(gabor(:,i))-min(gabor(:,i)))>0)
        gabor(:,i) = (gabor(:,i)-min(gabor(:,i)))/(max(gabor(:,i))-min(gabor(:,i)));
    end
end

%% Perform UMAP embedding
addpath UMAP/umap
[reduction, ~, ~, ~]=run_umap(pca(gabor,80),'n_components',2,'min_dist',0.1,'n_neighbors',8);

%% Perform CDC clustering
k_num = 5;
ratio = 0.7;
cluster = CDC(reduction, k_num, ratio);

%% Evaluate the clustering accuracy and plot the result
```

```
addpath ClusterEvaluation
[Accuracy, NMI, ARI, Fscore, JI, RI] = ClustEval(label, cluster);
plotcluster(reduction, label);
```