User Manual for KCC—a MATLAB package for K-means-based Consensus Clustering

Hao Lin[1], Hongfu Liu[2], Junjie Wu[3], Hong Li[4], Stephan Günnemann[5]

1. INTRODUCTION

This user manual systematically presents the usage of the MATLAB package on K-means-based Consensus Clustering (KCC) accompanying the following paper:

> Hao Lin, Hongfu Liu, Junjie Wu, Hong Li, and Stephan Günnemann. 2022. Algorithm xxxx: KCC: A MATLAB Package for K-means-based Consensus Clustering. *ACM Trans. Math. Softw.*

The package was developed and tested in MATLAB R2012a under Linux. To those without access to MATLAB and those who prefer to use free open source software, we also investigate the usage of KCC with OCTAVE, and find it is also compatible with OCTAVE without additional efforts. OCTAVE version?.

For package installation, you need to first unpack the compressed archive into your current directory. It consists of a *source code* folder Matlab, a folder userManual with this comprehensive *user manual*, and a *license* file LICENSE indicating that the package is distributed under GNU GENERAL PUBLIC LICENSE (Version 3). Then under the Matlab folder, you need to add one of its subfolder, i.e., the Src folder, to the MATLAB path. The directory structure of the Matlab folder is described as follows.

- Src (core functions for conducting KCC)
  - BasicCluster_RFS.m (function to generate BPs with RFS)
  - BasicCluster_RPS.m (function to generate BPs with RPS)
  - Preprocess.m (function to prepare for consensus clustering)
  - KCC.m (consensus function)
  - exMeasure.m (function to compute validity scores for clustering results)
  - load_sparse.m (auxiliary function to load input text data as a sparse matrix)
  - hungarian.m (auxiliary function for cluster label assignment)
  - BasicCluster_RPS_missing.m (auxiliary function to generate IBPs with strategy-I)
  - addmissing.m (auxiliary function to generate IBPs using strategy-II)
  - distance_* (distance functions)
  - gClusterDistribution.m (auxiliary function to calculate cluster distribution for BPs)
  - Ucompute.m, Ucompute_miss.m (auxiliary function for utility calculation)
  - gCentroid.m, gCentroid_miss.m (auxiliary function for centroid update)
  - sCentroid.m, sCentroid_miss.m (auxiliary function for centroid initialization)
- Drivers (illustrative examples)
  - data (input data for illustration)
  - demo.m (function for KCC with different utility functions)

[1]Department of Informatics, Technical University of Munich, Germany, linh@in.tum.de.
[2]Michtom School of Computer Science, Brandeis University, USA, hongfuliu@brandeis.edu.
[3]School of Economics and Management, Beihang University, China, wujj@buaa.edu.cn.
[4]School of Economics and Management, Beihang University, China, hong_lee@buaa.edu.cn.
[5]Department of Informatics, Technical University of Munich, Germany, guennemann@in.tum.de.

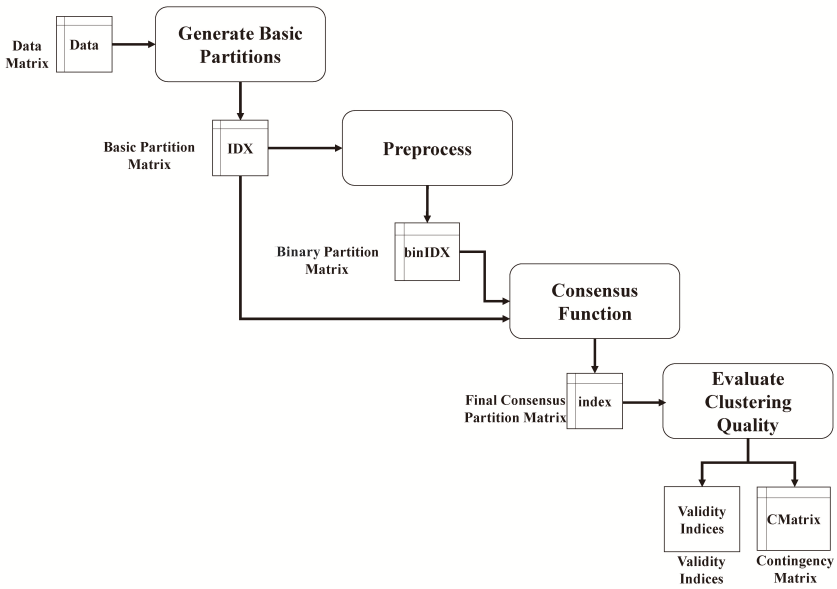

Fig. 1. Typical program flow of KCC package.

demoIBPI.m (function for KCC with IBPs generated by strategy-I)
demoIBPII.m (function for KCC with IBPs generated by strategy-II)
demoNumberBP.m (function for KCC with varying number of BPs)
demoStrategyBP.m (function for KCC with RFS strategy for BP generation)

Figure 1 illustrates a typical program flow of using the KCC package, which includes data preparation, basic partitions generation, consensus clustering preprocessing, consensus function, and clustering quality evaluation. We can see from Figure 1 that, in a typical flow for consensus clustering, a real-world matrix Data is first input to generate a basic partition matrix IDX. The basic partition matrix is then input to a Preprocess function to produce the sparse representation of $\mathcal{X}^{(b)}$, i.e., the binary matrix binIDX. The binary matrix binIDX, along with the basic partition matrix IDX is the input of the final consensus clustering via a K-means heuristic, also known as consensus function, which produces a consensus partition matrix index. Lastly, the clustering quality is evaluated with an exMeasure function, which outputs several external validity indices and a contingency matrix. We will take a detailed look at the important functions and fields in the Guides in Section 2.

2. GUIDES

Syntax:

```
IDX = BasicCluster_RFS(Data,r,K,dist,nFeature
```

Input parameters:

`Data` : an `n` × `p` matrix of data, whose rows correspond to `n` observations, and columns correspond to `p` featu

`r` :

`K` :

`dist` : sqEuclidean namely the squared Euclidean distance, and each centroid is the mean of the data points i

`nFeature` : sampled uniformly at random without replacement from the integers 1 to p, which forms the inc

Output parameters:

`IDX` : a n × r cluster labels matrix for n data points in r basic partitions