

# ConsensusClusterPlus (Tutorial)

Matthew D. Wilkerson

October 29, 2025

## 1 Summary

**ConsensusClusterPlus** is a tool for unsupervised class discovery. This document provides a tutorial of how to use **ConsensusClusterPlus**.

## 2 Brief description of *Consensus Clustering*

*Consensus Clustering* [1] is a method that provides quantitative evidence for determining the number and membership of possible clusters within a dataset, such as microarray gene expression. This method has gained popularity in cancer genomics, where new molecular subclasses of disease have been discovered [3, 4]. The *Consensus Clustering* method involves subsampling from a set of items, such as microarrays, and determines clusterings of specified cluster counts ( $k$ ). Then, pairwise *consensus* values, the proportion that two items occupied the same cluster out of the number of times they occurred in the same subsample, are calculated and stored in a symmetrical *consensus matrix* for each  $k$ . The *consensus matrix* is summarized in several graphical displays that enable a user to decide upon a reasonable cluster number and membership. A web-based version of *Consensus Clustering* is publicly available [5]. For a formal description, see [1].

**ConsensusClusterPlus**[2] implements the *Consensus Clustering* method in *R* and extends it with new features and graphical outputs that can aid users in class discovery.

## 3 Tutorial

There are three main steps to use **ConsensusClusterPlus**: preparing input data, running the program, and generating cluster-consensus and item-consensus.

### 3.1 Preparing input data

The first step is to gather some data for cluster analysis. These data could be the result of an experiment such as a mRNA expression microarray or immunohistochemical staining intensities. The input data format is a matrix where columns are samples (items), rows are features and cells are numerical values. For this tutorial, we use the ALL gene expression data from the ALL library. You can see the matrix `d` is already in the proper format. The column and row names, which correspond to the sample and gene names, will be maintained in the output.

```
> library(ALL)
> data(ALL)
> d=exprs(ALL)
> d[1:5,1:5]
```

|           | 01005    | 01010    | 03002    | 04006    | 04007    |
|-----------|----------|----------|----------|----------|----------|
| 1000_at   | 7.597323 | 7.479445 | 7.567593 | 7.384684 | 7.905312 |
| 1001_at   | 5.046194 | 4.932537 | 4.799294 | 4.922627 | 4.844565 |
| 1002_f_at | 3.900466 | 4.208155 | 3.886169 | 4.206798 | 3.416923 |
| 1003_s_at | 5.903856 | 6.169024 | 5.860459 | 6.116890 | 5.687997 |
| 1004_at   | 5.925260 | 5.912780 | 5.893209 | 6.170245 | 5.615210 |

For the purpose of selecting the most informative genes for class detection, we reduce the dataset to the top 5,000 most variable genes, measured by median absolute deviation. The choice of 5,000 genes and MAD can be substituted with other statistical variability filters. Users can decide what type of filtering to use or to skip filtering. Another choice would be to supply weights for sampling genes see `weightsFeatures` in Additional Options.

```
> mads=apply(d,1,mad)
> d=d[rev(order(mads))[1:5000],]
```

If one wants to transform or normalize their data, they can easily do so using other Bioconductor methods or a simple statement. We chose to use the default settings of the agglomerative hierarchical clustering algorithm using Pearson correlation distance, so it is appropriate to gene median center `d` using this simple statement:

```
> d = sweep(d,1, apply(d,1,median,na.rm=T))
```

`d` is now ready for `ConsensusClusterPlus` analysis.

### 3.2 Running ConsensusClusterPlus

For this tutorial, we selected 80% item resampling (`pItem`), 80% gene resampling (`pFeature`), a maximum evaluated `k` of 6 so that cluster counts of 2,3,4,5,6 are evaluated (`maxK`), 50 resamplings (`reps`), agglomerative hierarchical clustering

algorithm (clusterAlg) upon 1- Pearson correlation distances (distance), gave our output a title (title), and opted to have graphical results written to png files. We also used a specific random seed so that this example is repeatable (seed).

\* Note: In practice, a much higher reps is recommended such as 1,000 and a higher cluster count such as 20.

```
> library(ConsensusClusterPlus)
> title=tempdir()
> results = ConsensusClusterPlus(d,maxK=6, reps=50, pItem=0.8, pFeature=1,
+ title=title, clusterAlg="hc", distance="pearson", seed=1262118388.71279, plot="png")
```

The output of ConsensusClusterPlus is a list, in which the element of the list corresponds to results from the  $k$ th cluster, for instance, results[[2]] is the results result of  $k=2$ . The seed option specifies a random number seed and is used here for reproducibility of this tutorial. These list elements have the following elements:

```
> #consensusMatrix - the consensus matrix.
> #For .example, the top five rows and columns of results for k=2:
> results[[2]][["consensusMatrix"]][1:5,1:5]
```

|      | [,1]      | [,2]      | [,3]      | [,4]      | [,5]      |
|------|-----------|-----------|-----------|-----------|-----------|
| [1,] | 1.0000000 | 1.0000000 | 0.8947368 | 1.0000000 | 1.0000000 |
| [2,] | 1.0000000 | 1.0000000 | 0.9142857 | 1.0000000 | 1.0000000 |
| [3,] | 0.8947368 | 0.9142857 | 1.0000000 | 0.8857143 | 0.969697  |
| [4,] | 1.0000000 | 1.0000000 | 0.8857143 | 1.0000000 | 1.0000000 |
| [5,] | 1.0000000 | 1.0000000 | 0.9696970 | 1.0000000 | 1.0000000 |

```
> #consensusTree - hclust object
> results[[2]][["consensusTree"]]
```

Call:

```
hclust(d = as.dist(1 - fm), method = finalLinkage)
```

Cluster method : average

Number of objects: 128

```
> #consensusClass - the sample classifications
> results[[2]][["consensusClass"]][1:5]
```

```
01005 01010 03002 04006 04007
      1      1      1      1      1
```

```
>
```

```
> #ml - consensus matrix result
```

```
> #clrs - colors for cluster
```

See additional options section for further description of clustering algorithms and distance metrics.

### 3.3 Generating cluster and item consensus

After executing `ConsensusClusterPlus`, one can optionally calculate cluster-consensus and item-consensus results by:

```
> icl = calcICL(results,title=title,plot="png")
```

`calcICL` returns a list of two elements:

```
> icl[["clusterConsensus"]]
```

|       | k | cluster | clusterConsensus |
|-------|---|---------|------------------|
| [1,]  | 2 | 1       | 0.7681668        |
| [2,]  | 2 | 2       | 0.9788274        |
| [3,]  | 3 | 1       | 0.6176820        |
| [4,]  | 3 | 2       | 0.9190744        |
| [5,]  | 3 | 3       | 1.0000000        |
| [6,]  | 4 | 1       | 0.8446083        |
| [7,]  | 4 | 2       | 0.9067267        |
| [8,]  | 4 | 3       | 0.6612850        |
| [9,]  | 4 | 4       | 1.0000000        |
| [10,] | 5 | 1       | 0.8175802        |
| [11,] | 5 | 2       | 0.9066489        |
| [12,] | 5 | 3       | 0.6062040        |
| [13,] | 5 | 4       | 0.8154580        |
| [14,] | 5 | 5       | 1.0000000        |
| [15,] | 6 | 1       | 0.7511726        |
| [16,] | 6 | 2       | 0.8802040        |
| [17,] | 6 | 3       | 0.7410730        |
| [18,] | 6 | 4       | 0.8154580        |
| [19,] | 6 | 5       | 0.7390864        |
| [20,] | 6 | 6       | 1.0000000        |

```
> icl[["itemConsensus"]][1:5,]
```

|   | k | cluster | item  | itemConsensus |
|---|---|---------|-------|---------------|
| 1 | 2 | 1       | 28031 | 0.6173782     |
| 2 | 2 | 1       | 28023 | 0.5797202     |
| 3 | 2 | 1       | 43012 | 0.5961974     |
| 4 | 2 | 1       | 28042 | 0.5644619     |
| 5 | 2 | 1       | 28047 | 0.6259350     |

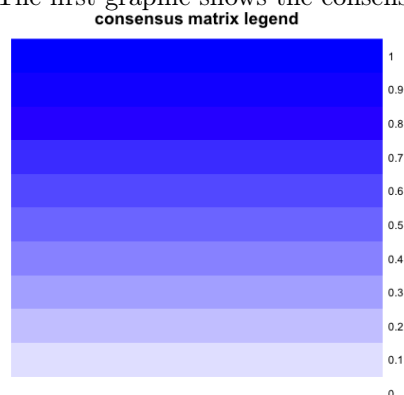
## 4 Graphic Output Description

The output of `ConsensusClusterPlus` consists of graphics, which are written to the screen, 'pdf' file, or 'png' files depending on the plot option; and numerical data which can be optionally written to a CSV file depending on the writeTable

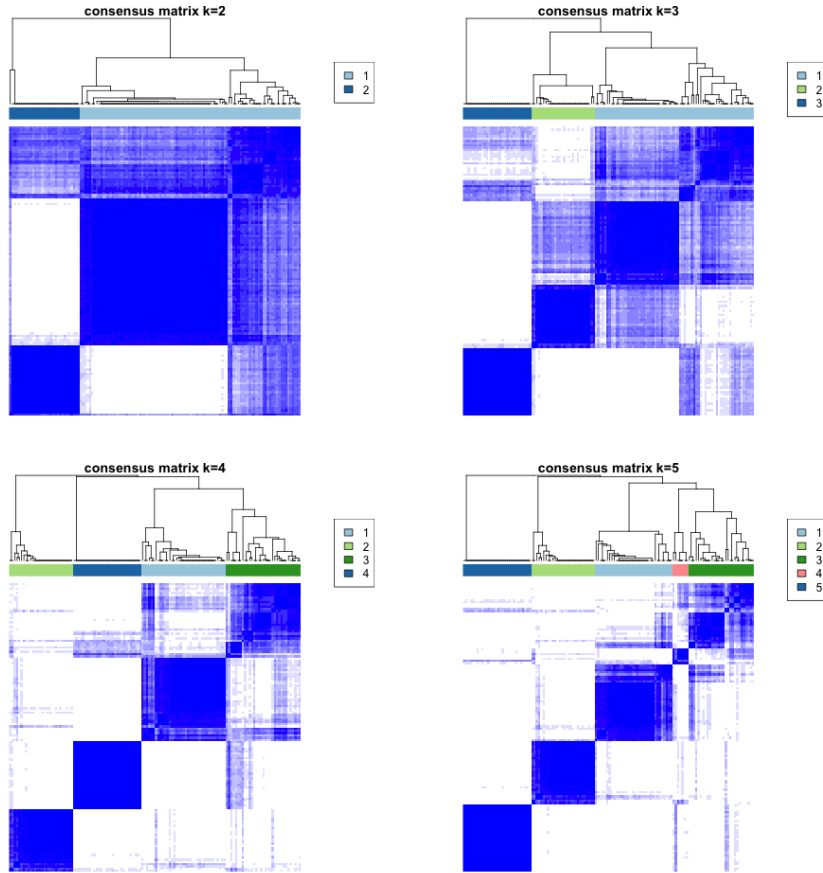
option. For large datasets, graphical displays can be quite large and plotting the consensus dendrogram above the consensus matrices may not be possible. If your dataset is large, the plot option 'pngBMP' which does not produce the consensus matrix dendrogram and uses the bitmap function rather png. Bitmap is often available natively on linux systems but can potentially be installed on other systems.

## 4.1 Consensus Matrices

The first graphic shows the consensus color legend.

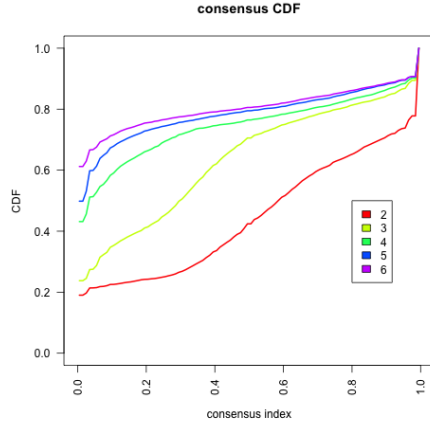


The remaining graphics are heatmaps of the consensus matrices for  $k = 2, 3, 4, 5$  [1]. The consensus matrices have items as both rows and columns, which are microarrays in this example, and where consensus values range from 0 (never clustered together) to 1 (always clustered together) marked by white to dark blue. The consensus matrices are ordered by the consensus clustering which is depicted as a dendrogram atop the heatmap. To aid analysis, the cluster memberships are marked by colored rectangles between the dendrogram and heatmap according to a legend within the graphic. This enables a user to compare a clusters' member count in the context of their consensus.



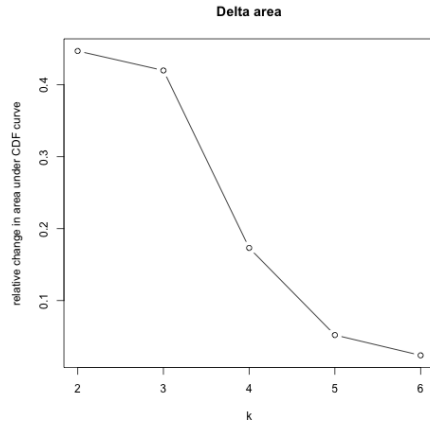
## 4.2 Consensus Cumulative Distribution Function (CDF) Plot

This graphic shows the cumulative distribution functions [1] of the consensus matrix for each  $k$  (indicated by colors), estimated by a histogram of 100 bins. This figure allows a user to determine at what number of clusters,  $k$ , the CDF reaches an approximate maximum, thus consensus and cluster confidence is at a maximum at this  $k$ . See [1] for further details interpretation.



### 4.3 Delta Area Plot

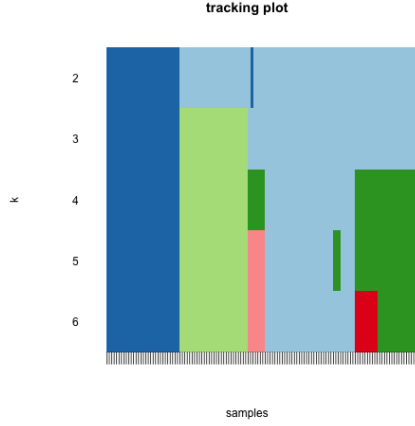
This graphic shows the relative change in area under the CDF curve [1] comparing  $k$  and  $k - 1$ . For  $k = 2$ , there is no  $k - 1$ , so the total area under the curve rather than the relative increase is plotted. This plot allows a user to determine the relative increase in consensus and determine  $k$  at which there is no appreciable increase. See [1] for interpretation.



### 4.4 Tracking Plot

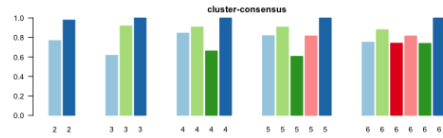
This graphic shows the cluster assignment of items (columns) for each  $k$  (rows) by color. The colors correspond to the colors of the consensus matrix class assignments. Hatch marks below the plot indicate items/samples. This plot provides a view of item cluster membership across different  $k$  and enables a user

to track the history of clusters relative to earlier clusters. Items that change clusters often (changing colors within a column) are indicative of unstable membership. Clusters with an abundance of unstable members suggest an unstable cluster.



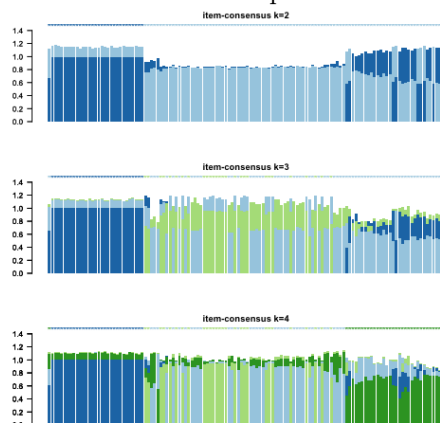
## 4.5 Cluster-Consensus Plot

This graphic shows the *cluster-consensus* value of clusters at each  $k$ . This is the mean of all pairwise consensus values between a cluster's members. Cluster is indicated by color following the same color scheme as the cluster matrices and tracking plots. The bars are grouped by  $k$  which is marked on the horizontal axis. High values indicate a cluster has high stability and low values indicate a cluster has low stability. This plot enables a user to view the mean cluster-consensus among clusters at a given  $k$  and compare values of clusters across different  $k$  via the color scheme.



## 4.6 Item-Consensus Plot

*Item-consensus* values are the mean consensus of an item with all items in a particular cluster. An item has  $k$  item-consensus values corresponding to each cluster at a particular  $k$ . These values are depicted in barplots for each  $k$ . Samples are stacked bars. *Item-consensus* values are indicated by the heights of the colored portion of the bars, whose color corresponds to the common color scheme. Bars' rectangles are ordered by increasing value from bottom to top. The asterisks at the top indicate the consensus cluster for each item.



This plot provides a view of item-consensus across all other clusters at a given  $k$ . This enables a user to see if a sample is a very "pure" member of a cluster or if it shares high consensus to multiple clusters (large rectangles in a column of multiple colors), suggesting that it is an unstable or "unpure" member. These values could be used to select "core" samples similar to [4] that are highly representative of a cluster. Further, this plot can aid cluster number decisions. For instance, if a cluster consists mainly of members with very "unpure" items, then this evidence could be used to support a maximum cluster number at 1 below this  $k$  or this evidence could support that this cluster is an outlier cluster. Decisions such as these are best to be made by the user in conjunction with other evidence such as consensus matrices, tracking plots, etc.

## 4.7 Additional details on options for ConsensusClusterPlus function

- **d** This option specifies the data to be used in ConsensusClusterPlus. This is typically a matrix of numerical expression values, of which an example is provided in the Running ConsensusClusterPlus section of this document. When provided with a data matrix as **d**, ConsensusClusterPlus recalculates a distance matrix during each iteration. This recalculation is required if feature resampling is specified (**pFeature** less than 1). However with very

large datasets (1,000's of items) and no feature resampling, this process can be time consuming and unnecessary. Alternatively, a pre-computed distance matrix can be provided as `d`, resulting in faster computation. An example of using a `dist` object as input follow below.

```
> #example of providing a custom distance matrix as input:
> #dt = as.dist(1-cor(d,method="pearson"))
> #ConsensusClusterPlus(dt,maxK=4, reps=100, pItem=0.8, pFeature=1, title="example2", distance="pearson")
```

- **distance** This option describes the distance metric to be used. A character value of one of the following metrics is accepted: `pearson` for (1 - Pearson correlation), `spearman` for (1 - Spearman correlation), `euclidean`, `binary`, `maximum`, `canberra`, `minkowski`. Alternatively a custom distance function can be supplied for this argument, which accepts a numerical matrix (items as rows and features as columns) as input and returns a `dist` object.

```
> #example of providing a custom distance function:
> #myDistFunc = function(x){ dist(x,method="manhattan")}
> #ConsensusClusterPlus(d,maxK=4, reps=100, pItem=0.8, pFeature=1, title="example3", distance="myDistFunc")
```

- **clusterAlg** This option specifies the type of clustering algorithm to use: `"hc"` for hierarchical clustering, `"pam"` for partitioning around medoids, `"km"` for kmeans. Alternatively, one can supply their own clustering function, which should accept a distance matrix and a cluster number as its arguments and returns vector of cluster assignments having the same order as the distance matrix columns. For example, this simple function executes divisive clustering using the `diana` function from the `cluster` package and returns the expected object. The last line shows an example of how this could be used.

```
> #library(cluster)
> #dianaHook = function(this_dist,k){
>   #tmp = diana(this_dist,diss=TRUE)
>   #assignment = cutree(tmp,k)
>   #return(assignment)
> #}
> #ConsensusClusterPlus(d,clusterAlg="dianaHook",distance="pearson",...)
```

- **update on kmeans options** `"km"` option performs kmeans clustering directly on a data matrix, with items and features resampled.
- **innerLinkage** This option specifies the linkage method to use in iterative agglomerative hierarchical clustering. Not applicable to other cluster algorithms.
- **finalLinkage** This option specifies the linkage method to use in the final agglomerative hierarchical clustering.

- **distance** This option specifies the distance metric to use: "pearson" for 1-Pearson correlation coefficient, "spearman" for 1-Spearman correlation coefficient, "euclidean" for Euclidean distance.
- **tmyPal** character vector of ordered colors to use for consensus matrix. If not specified, a series of white to blue colors is used.
- **writeTable** boolean. If TRUE, write consensus matrices, ICL, and log to file.
- **weightsFeature** numerical vector of weights for sampling features. See help for further details.
- **weightsItem** numerical vector of weights for sampling items. See help for further details.
- **verbose** boolean. If TRUE, print messages to the screen to indicate progress. This is useful for large datasets.

## References

- [1] Monti, S., Tamayo, P., Mesirov, J., Golub, T. (2003) Consensus Clustering: A Resampling-Based Method for Class Discovery and Visualization of Gene Expression Microarray Data. *Machine Learning*, 52, 91–118.
- [2] Wilkerson, M.D., Hayes, D.N. (2010). ConsensusClusterPlus: a class discovery tool with confidence assessments and item tracking. *Bioinformatics*, 2010 Jun 15;26(12):1572–3.
- [3] Hayes, D.N, Monti, S., Parmigiani, G. et al. (2006) Gene Expression Profiling Reveals Reproducible Human Lung Adenocarcinoma Subtypes in Multiple Independent Patient Cohorts. *Journal of Clinical Oncology*, 24 (31) 5079–5090.
- [4] Verhaak, R., Hoadley, K., et al. (2010) Integrated genomic analysis identifies clinically relevant subtypes of glioblastoma characterized by abnormalities in PDGFRA, IDH1, EGFR and NF1. *Cancer Cell*. 17,1-13.
- [5] <http://www.broadinstitute.org/cancer/software/genepattern/>

## 4.8 Changes

- Version 1.0.1. Item-consensus calculation was corrected. Consensus matrix heat maps are now guaranteed to correspond to the scale.
- Version 1.5.1. Version 1.0.1 changes were re-incorporated into Bioc 2.9, 2.8. Version 1.0.1 was part of Bioc 2.6, but not part of Bioc 2.7.

- Version 1.11.1. For large datasets, the input data (d) was modified to also accept a distance matrix which reduces computation time, and plotBMP was added a plot type so that large consensus matrices can be plotted. Internal data structures were modified to increase speed. Distance metric options expanded ("maximum", "manhattan", "canberra", "binary", "minkowski" from dist) and custom distance function option added. Partitioning Around Medoids clustering (from cluster package) was added as a clustering algorithm. Kmeans invocation was changed to run on the data matrix by default. Kmeans invocation on a distance matrix is now possible by kmDIST.
- Version Version 1.35.0 Added CITATION file, updated references, and man pages.
- Version 1.51.1 Brief R code update for compatibility with R 4.0. Deprecated kmDIST clustering option.