clValid, an R package for cluster validation

Guy Brock, Vasyl Pihur, Susmita Datta, and Somnath Datta

Department of Bioinformatics and Biostatistics, University of Louisville

Note: A previous version of this manuscript was published in the *Journal of Statistical Software* (Brock et al., March 2008).

February 14, 2021

Contents

Abstract

The R package *clValid* contains functions for validating the results of a clustering analysis. There are three main types of cluster validation measures available, "internal", "stability", and "biological". The user can choose from nine clustering algorithms in existing R packages, including hierarchical, K-means, self-organizing maps (SOM),

and model based clustering. In addition, we provide a function to perform the self-organizing tree algorithm (SOTA) method of clustering. Any combination of validation measures and clustering methods can be requested in a single function call. This allows the user to simultaneously evaluate several clustering algorithms while varying the number of clusters, to help determine the most appropriate method and number of clusters for the dataset of interest. Additionally, the package can automatically make use of the biological information contained in the Gene Ontology (GO) database to calculate the biological validation measures, via the annotation packages available in Bioconductor. The function returns an object of S4 class *clValid*, which has summary, plot, print, and additional methods which allow the user to display the optimal validation scores and extract clustering results.

1 Introduction

Clustering is an unsupervised technique used to group together objects which are "close" to one another in a multidimensional feature space, usually for the purpose of uncovering some inherent structure which the data possesses. Clustering is commonly used in the analysis of high-throughput genomic data, with the aim of grouping together genes or proteins which have similar expression patterns and possibly share common biological pathways (DeRisi et al., 1997; Chu et al., 1998; Eisen et al., 1998; Bhattacherjee et al., 2007). A plethora of clustering algorithms currently exist, many of which have shown some promise in the analysis of genomic data (Herrero et al., 2001; McLachlan et al., 2002; Dembele and Kastner, 2003; Fu and Medico, 2007). Deciding which clustering method to use can therefore be a daunting task for the researcher conducting the experiment. An additional, related problem is determining the number of clusters that are most appropriate for the data. Ideally, the resulting clusters should not only have good statistical properties (compact, well-separated, connected, and stable), but also give results that are biologically relevant.

A variety of measures aimed at validating the results of a clustering analysis and determining which clustering algorithm performs the best for a particular experiment have been proposed (Kerr and Churchill, 2001; Yeung et al., 2001; Datta and Datta, 2003). This validation can be based solely on the internal properties of the data or on some external reference, and on the expression data alone or in conjunction with relevant biological information (Gibbons and Roth, 2002; Gat-Viks et al., 2003; Bolshakova et al., 2005; Datta and Datta, 2006). The article by Handl et al. (2005), in particular, gives an excellent overview of cluster validation with post-genomic data and

provides a synopsis of many of the available validation measures.

In this paper, we present an R package *clValid* which contains a variety of methods for validating the results from a clustering analysis. The main function is `clValid()`, and the available validation measures fall into the three general categories of "internal", "stability", and "biological". The user can simultaneously select multiple clustering algorithms, validation measures, and numbers of clusters in a single function call, to determine the most appropriate method and an optimal number of clusters for the dataset. Additionally, the *clValid* package makes use of the biological information contained in the Gene Ontology (GO) database via the annotation packages in Bioconductor, in order to automate the calculation of the biological validation measures. The package also contains a function for implementing the self-organizing tree algorithm (SOTA), which to our knowledge has not been previously available in R packages on CRAN. The function returns an object of S4 class *clValid*, which has a variety of methods available to plot and summarize the validation measures, display the optimal scores along with the corresponding cluster method and number of clusters, and extract the clustering results for a particular algorithm.

The rest of this paper is organized as follows. Section 2 contains a detailed description of the validation measures that are available. Section 3 describes the clustering algorithms which are available to use with the *clValid* package. Section 4 contain an example using mouse gene expression data from Bhattacherjee et al. (2007) that illustrates the use of the *clValid* package functions and objects. Finally, Section 5 discusses some additional validation software which is available, and some of the benefits our software provides in comparison.

2 Validation Measures

The *clValid* package offers three types of cluster validation, "internal", "stability", and "biological". Internal validation measures take only the dataset and the clustering partition as input and use intrinsic information in the data to assess the quality of the clustering. The stability measures are a special version of internal measures. They evaluate the consistency of a clustering result by comparing it with the clusters obtained after each column is removed, one at a time. Biological validation evaluates the ability of a clustering algorithm to produce biologically meaningful clusters. We have measures to investigate both the biological homogeneity and stability of the clustering results.

2.1 Internal measures

For internal validation, we selected measures that reflect the compactness, connectedness, and separation of the cluster partitions. Connectedness relates to what extent observations are placed in the same cluster as their nearest neighbors in the data space, and is here measured by the connectivity (Handl et al., 2005). Compactness assesses cluster homogeneity, usually by looking at the intra-cluster variance, while separation quantifies the degree of separation between clusters (usually by measuring the distance between cluster centroids). Since compactness and separation demonstrate opposing trends (compactness increases with the number of clusters but separation decreases), popular methods combine the two measures into a single score. The Dunn Index (Dunn, 1974) and Silhouette Width (Rousseeuw, 1987) are both examples of non-linear combinations of the compactness and separation, and with the connectivity comprise the three internal measures available in *clValid*. The details of each measure are given below, and for a good overview of internal measures in general see Handl et al. (2005).

Connectivity

Let N denote the total number of observations (rows) in a dataset and M denote the total number of columns, which are assumed to be numeric (e.g., a collection of samples, time points, etc.). Define $nn_{i(j)}$ as the jth nearest neighbor of observation i, and let $x_{i,nn_{i(j)}}$ be zero if i and j are in the same cluster and $1/j$ otherwise. Then, for a particular clustering partition $\mathcal{C} = \{C_1, \ldots, C_K\}$ of the N observations into K disjoint clusters, the connectivity is defined as

$$Conn(\mathcal{C}) = \sum_{i=1}^{N} \sum_{j=1}^{L} x_{i,nn_{i(j)}} \ ,$$

where L is a parameter giving the number of nearest neighbors to use. The connectivity has a value between zero and ∞ and should be minimized.

Silhouette Width

The Silhouette Width is the average of each observation's Silhouette value. The Silhouette value measures the degree of confidence in the clustering assignment of a particular observation, with well-clustered observations having values near 1 and poorly clustered observations having values near -1. For

observation i, it is defined as

$$S(i) = \frac{b_i - a_i}{max(b_i, a_i)},$$

where a_i is the average distance between i and all other observations in the same cluster, and b_i is the average distance between i and the observations in the "nearest neighboring cluster", i.e.

$$b_i = \min_{C_k \in \mathcal{C} \setminus C(i)} \sum_{j \in C_k} \frac{dist(i,j)}{n(C_k)},$$

where $C(i)$ is the cluster containing observation i, $dist(i,j)$ is the distance (e.g. Euclidean, Manhattan) between observations i and j, and $n(C)$ is the cardinality of cluster C. The Silhouette Width thus lies in the interval $[-1, 1]$, and should be maximized. For more information, see the help page for the `silhouette()` function in package *cluster* (Rousseeuw et al., 2006).

Dunn Index

The Dunn Index is the ratio of the smallest distance between observations not in the same cluster to the largest intra-cluster distance. It is computed as

$$D(\mathcal{C}) = \frac{\min_{C_k, C_l \in \mathcal{C}, C_k \neq C_l} \left(\min_{i \in C_k, j \in C_l} dist(i,j) \right)}{\max_{C_m \in \mathcal{C}} diam(C_m)},$$

where $diam(C_m)$ is the maximum distance between observations in cluster C_m. The Dunn Index has a value between zero and ∞, and should be maximized.

2.2 Stability measures

The stability measures compare the results from clustering based on the full data to clustering based on removing each column, one at a time. These measures work especially well if the data are highly correlated, which is often the case in high-throughput genomic data. The included measures are the average proportion of non-overlap (APN), the average distance (AD), the average distance between means (ADM), and the figure of merit (FOM) (Datta and Datta, 2003; Yeung et al., 2001). In all cases the average is taken over all the deleted columns, and all measures should be minimized.

Average Proportion of Non-overlap (APN)

The APN measures the average proportion of observations not placed in the same cluster by clustering based on the full data and clustering based on the data with a single column removed. Let $C^{i,0}$ represent the cluster containing observation i using the original clustering (based on all available data), and $C^{i,\ell}$ represent the cluster containing observation i where the clustering is based on the dataset with column ℓ removed. Then, with the total number of clusters set to K, the APN measure is defined as

$$APN(K) = \frac{1}{MN} \sum_{i=1}^{N} \sum_{\ell=1}^{M} \left(1 - \frac{n(C^{i,\ell} \cap C^{i,0})}{n(C^{i,0})} \right) .$$

The APN is in the interval $[0, 1]$, with values close to zero corresponding with highly consistent clustering results.

Average Distance (AD)

The AD measure computes the average distance between observations placed in the same cluster by clustering based on the full data and clustering based on the data with a single column removed. It is defined as

$$AD(K) = \frac{1}{MN} \sum_{i=1}^{N} \sum_{\ell=1}^{M} \frac{1}{n(C^{i,0}) n(C^{i,\ell})} \left[\sum_{i \in C^{i,0}, j \in C^{i,\ell}} dist(i, j) \right] .$$

The AD has a value between zero and ∞, and smaller values are preferred.

Average Distance between Means (ADM)

The ADM measure computes the average distance between cluster centers for observations placed in the same cluster by clustering based on the full data and clustering based on the data with a single column removed. It is defined as

$$ADM(K) = \frac{1}{MN} \sum_{i=1}^{N} \sum_{\ell=1}^{M} dist(\bar{x}_{C^{i,\ell}}, \bar{x}_{C^{i,0}}) ,$$

where $\bar{x}_{C^{i,0}}$ is the mean of the observations in the cluster which contain observation i, when clustering is based on the full data, and $\bar{x}_{C^{i,\ell}}$ is similarly defined. Currently, ADM only uses the Euclidean distance. It also has a value between zero and ∞, and again smaller values are prefered.

Figure of Merit (FOM)

The FOM measures the average intra-cluster variance of the observations in the deleted column, where the clustering is based on the remaining (undeleted) samples. This estimates the mean error using predictions based on the cluster averages. For a particular left-out column ℓ, the FOM is

$$FOM(\ell, K) = \sqrt{\frac{1}{N} \sum_{k=1}^{K} \sum_{i \in C_k(\ell)} dist(x_{i,\ell}, \bar{x}_{C_k(\ell)})},$$

where $x_{i,\ell}$ is the value of the ith observation in the ℓth column in cluster $C_k(\ell)$, and $\bar{x}_{C_k(\ell)}$ is the average of cluster $C_k(\ell)$. Currently, the only distance available for FOM is Euclidean. The FOM is multiplied by an adjustment factor $\sqrt{\frac{N}{N-K}}$, to alleviate the tendency to decrease as the number of clusters increases. The final score is averaged over all the removed columns, and has a value between zero and ∞, with smaller values equaling better performance.

2.3 Biological

Biological validation evaluates the ability of a clustering algorithm to produce biologically meaningful clusters. A typical application of biological validation is in microarray data, where observations correspond to genes (where "genes" could be open reading frames (ORFs), express sequence tags (ESTs), serial analysis of gene expression (SAGE) tags, etc.). There are two measures available, the biological homogeneity index (BHI) and biological stability index (BSI), both originally presented in Datta and Datta (2006).

Biological Homogeneity Index (BHI)

As its name implies, the BHI measures how homogeneous the clusters are biologically. Let $\mathcal{B} = \{B_1, \ldots, B_F\}$ be a set of F functional classes, not necessarily disjoint, and let $B(i)$ be the functional class containing gene i (with possibly more than one functional class containing i). Similarly, we define $B(j)$ as the function class containing gene j, and assign the indicator function $I(B(i) = B(j))$ the value 1 if $B(i)$ and $B(j)$ match (any one match is sufficient in the case of membership to multiple functional classes), and 0 otherwise. Intuitively, we hope that genes placed in the same statistical cluster also belong to the same functional classes. Then, for a given statistical clustering partition $\mathcal{C} = \{C_1, \ldots, C_K\}$ and set of biological classes $\mathcal{B}$,

the BHI is defined as

$$BHI(\mathcal{C}, \mathcal{B}) = \frac{1}{K} \sum_{k=1}^{K} \frac{1}{n_k(n_k - 1)} \sum_{i \neq j \in C_k} I\left(B(i) = B(j)\right) .$$

Here $n_k = n(C_k \cap B)$ is the number of annotated genes in statistical cluster C_k. Note that if $n_k = 1$ or 0, i.e. there is only one or no annotated genes in statistical cluster C_k, then a value of zero is added for that cluster. The BHI is in the range $[0, 1]$, with larger values corresponding to more biologically homogeneous clusters.

Biological Stability Index (BSI)

The BSI is similar to the other stability measures, and inspects the consistency of clustering for genes with similar biological functionality. Each sample is removed one at a time, and the cluster membership for genes with similar functional annotation is compared with the cluster membership using all available samples. The BSI is defined as

$$BSI(\mathcal{C}, \mathcal{B}) = \frac{1}{F} \sum_{k=1}^{F} \frac{1}{n(B_k)(n(B_k) - 1)M} \sum_{\ell=1}^{M} \sum_{i \neq j \in B_k} \frac{n(C^{i,0} \cap C^{j,\ell})}{n(C^{i,0})} ,$$

where F is the total number of functional classes, $C^{i,0}$ is the statistical cluster containing observation i based on all the data, and $C^{j,\ell}$ is the statistical cluster containing observation j when column ℓ is removed. Note that if $n(B_k) = 0$ or 1, i.e. there are less than two genes belonging to functional class B_k, then a value of zero is taken for that class. The BSI is in the range $[0, 1]$, with larger values corresponding to more stable clusters of the functionally annotated genes.

3 Clustering Algorithms

The R statistical computing project (R Development Core Team, 2006) has a wide variety of clustering algorithms available in the base distribution and various add-on packages. We make use of nine algorithms from the base distribution and add-on packages *cluster* (Rousseeuw et al., 2006; Kaufman and Rousseeuw, 1990), *kohonen* (Wehrens, 2006), and *mclust* (Fraley and Raftery, 2007; Fraley and A. E. Raftery, 2003), and in addition provide a function for implementing SOTA in the *clValid* package. A brief description of each clustering method and its availability is given below.

UPGMA

Unweighted Pair Group Method with Arithmetic Mean is probably the most frequently used clustering algorithm (Kaufman and Rousseeuw, 1990). It is an agglomerative, hierarchical clustering algorithm that yields a dendogram which can be cut at a chosen height to produce the desired number of clusters. Each observation is initially placed in its own cluster, and the clusters are successively joined together in order of their "closeness". The closeness of any two clusters is determined by a dissimilarity matrix, and can be based on a variety of agglomeration methods. UPGMA is included with the base distribution of R in function `hclust()`, and is also implemented in the `agnes()` function in package *cluster*.

K-means

K-means is an iterative method which minimizes the within-class sum of squares for a given number of clusters (Hartigan and Wong, 1979). The algorithm starts with an initial guess for the cluster centers, and each observation is placed in the cluster to which it is closest. The cluster centers are then updated, and the entire process is repeated until the cluster centers no longer move. Often another clustering algorithm (e.g., UPGMA) is run initially to determine starting points for the cluster centers. K-means is implemented in the function `kmeans()`, included with the base distribution of R.

Diana

Diana is a divisive hierarchical algorithm that initially starts with all observations in a single cluster, and successively divides the clusters until each cluster contains a single observation. Along with SOTA, Diana is one of a few representatives of the divisive hierarchical approach to clustering. Diana is available in function `diana()` in package *cluster*.

PAM

Partitioning around medoids (PAM) is similar to K-means, but is considered more robust because it admits the use of other dissimilarities besides Euclidean distance. Like K-means, the number of clusters is fixed in advance, and an initial set of cluster centers is required to start the algorithm. PAM is available in the *cluster* package as function `pam()`.

Clara

Clara is a sampling-based algorithm which implements PAM on a number of sub-datasets (Kaufman and Rousseeuw, 1990). This allows for faster running times when a number of observations is relatively large. Clara is also available in package *cluster* as function `clara()`.

Fanny

This algorithm performs fuzzy clustering, where each observation can have partial membership in each cluster (Kaufman and Rousseeuw, 1990). Thus, each observation has a vector which gives the partial membership to each of the clusters. A hard cluster can be produced by assigning each observation to the cluster where it has the highest membership. Fanny is available in the *cluster* package (function `fanny()`).

SOM

Self-organizing maps (Kohonen, 1997) is an unsupervised learning technique that is popular among computational biologists and machine learning researchers. SOM is based on neural networks, and is highly regarded for its ability to map and visualize high-dimensional data in two dimensions. SOM is available as the `som()` function in package *kohonen*.

Model based clustering

Under this approach, a statistical model consisting of a finite mixture of Gaussian distributions is fit to the data (Fraley and Raftery, 2001). Each mixture component represents a cluster, and the mixture components and group memberships are estimated using maximum likelihood (EM algorithm). The function `Mclust()` in package *mclust* implements model based clustering.

SOTA

Self-organizing tree algorithm (SOTA) is an unsupervised network with a divisive hierarchical binary tree structure. It was originally proposed by Dopazo and Carazo (1997) for phylogenetic reconstruction, and later applied to cluster microarray gene expression data in (Herrero et al., 2001). It uses a fast algorithm and hence is suitable for clustering a large number of objects. SOTA is included with the *clValid* package as function `sota()`.

4 Example - Mouse Mesenchymal Cells

To illustrate the cluster validation measures in package *clValid*, we use data from an Affymetrix microarray experiment comparing gene expression of mesenchymal cells from two distinct lineages, neural crest and mesoderm-derived. The dataset consists of 147 genes and ESTs which were determined to be significantly differentially expressed between the two cell lineages, with at least a 1.5 fold increase or decrease in expression. There are three samples for each of the neural crest and mesoderm-derived cells, so the expression matrix has dimension 147×6. In addition, the genes were grouped into the functional classes according to their biological description, with categories ECM/receptors (16), growth/differentiation (16), kinases/phosphatases (7), metabolism (8), stress-induced (6), transcription factors (28), and miscellaneous (25). The biological function of 10 genes was unknown, and 31 of the "genes" were ESTs. For further description of the dataset and the experiments the reader is referred to Bhattacherjee et al. (2007).

We begin by loading the dataset:

```
R> data(mouse)
```

his dataset has the typical format found in microarray data, with the rows as genes (variables) and the columns as the samples. Although this is a transposition of the data structure used for more conventional statistics (rows are samples, columns are variables), in both cases the typical goal is to cluster the rows based on the columns (although, in microarray data analysis the samples are also commonly clustered). Hence, the `clValid` function assumes that the rows of the input matrix are the intended items to be clustered.

We want to evaluate the results from a clustering analysis, using the clustering algorithms UPGMA, PAM, and K-means. Since the genes fall into one of two groups, up or down-regulated in the neural crest vs. mesoderm-derived tissue, the numbers of clusters is varied from 2 to 6. The distance metric (both for the applicable clustering methods and validation measures) is set to "euclidean"; other available options are "correlation" and "manhattan". The agglomeration method for hierarchical clustering is set to "average". We illustrate each category of validation measure separately, but it should be noted that the user can request all three types of validation measures at once (which would also be more computationally efficient).

4.1 Internal Validation

The internal validation measures are the connectivity, Silhouette Width, and Dunn Index. The neighborhood size for the connectivity is set to 10 by default, the `neighbSize` argument can be used to change this. Note that the clustering method "agnes" was omitted, since this also performs hierarchical clustering and would be redundant with the "hierarchical" method.

```
R> express <- mouse[,c("M1","M2","M3","NC1","NC2","NC3")]
R> rownames(express) <- mouse$ID
R> intern <- clValid(express, 2:6, clMethods=c("hierarchical","kmeans","pam"),
+                    validation="internal")
```

To view the results of the analysis, print, plot, and summary methods are available for the `clValid` object `intern`. The summary statement will display all the validation measures in a table, and also give the clustering method and number of clusters corresponding to the optimal score for each measure.

```
R> summary(intern)

Clustering Methods:
 hierarchical kmeans pam

Cluster sizes:
 2 3 4 5 6

Validation Measures:
                                 2       3       4       5       6

hierarchical Connectivity   5.3270 14.2528 20.7520 27.0726 30.6194
             Dunn           0.1291  0.0788  0.0857  0.0899  0.0899
             Silhouette     0.5133  0.4195  0.3700  0.3343  0.3233
kmeans       Connectivity  13.2548 17.6651 37.3980 43.2655 50.6095
             Dunn           0.0464  0.0873  0.0777  0.0815  0.0703
             Silhouette     0.4571  0.4182  0.3615  0.3367  0.3207
pam          Connectivity  18.7917 27.9651 30.9302 44.9671 32.9667
             Dunn           0.0391  0.0597  0.0510  0.0761  0.0816
             Silhouette     0.4271  0.3489  0.3563  0.3530  0.4152

Optimal Scores:
```

```
             Score  Method       Clusters
Connectivity 5.3270 hierarchical 2
Dunn         0.1291 hierarchical 2
Silhouette   0.5133 hierarchical 2
```

Hierarchical clustering with two clusters performs the best in each case. The validation measures can also be displayed graphically using the `plot()` method. Plots for individual measures can be requested using the `measures` argument. A legend is also included with each plot. The default location of the legend is the top right corner of each plot, this can be changed using the `legendLoc` argument. Here, we combine all three plots into a single figure and so suppress the legends in each individual plot.

```
R> op <- par(no.readonly=TRUE)
R> par(mfrow=c(2,2),mar=c(4,4,3,1))
R> plot(intern, legend=FALSE)
R> plot(nClusters(intern),measures(intern,"Dunn")[,,1],type="n",axes=F,
+         xlab="",ylab="")
R> legend("center", clusterMethods(intern), col=1:9, lty=1:9, pch=paste(1:9))
R> par(op)
```

The plots of the connectivity, Dunn Index, and Silhouette Width are given in Figure 1. Recall that the connectivity should be minimized, while both the Dunn Index and the Silhouette Width should be maximized. Thus, it appears that hierarchical clustering (UPGMA) outperforms the other clustering algorithms under each validation measure. For hierarchical clustering the optimal number of clusters is clearly two. For PAM, a case could be made for using six clusters.

4.2 Stability Validation

The stability measures include the APN, AD, ADM, and FOM. The measures should be minimized in each case. Stability validation requires more time than internal validation, since clustering needs to be redone for each of the datasets with a single column removed.

```
R> stab <- clValid(express, 2:6, clMethods=c("hierarchical","kmeans","pam"),
+                 validation="stability")
```

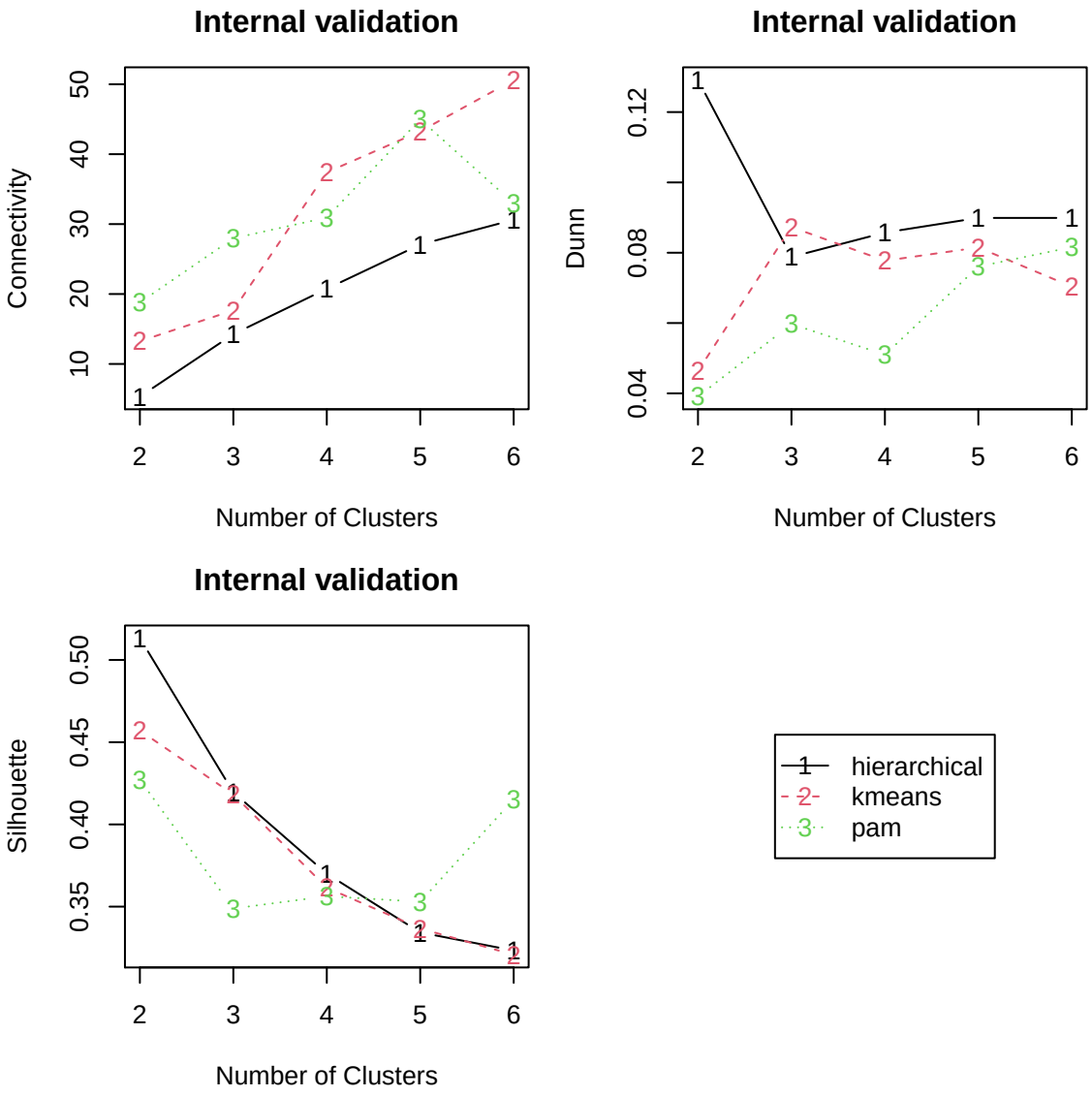

Figure 1: Plots of the connectivity measure, the Dunn Index, and the Silhouette Width.

Instead of viewing all the validation measures via the `summary()` method, we can instead just view the optimal values using the `optimalScores()` method.

```
R> optimalScores(stab)

        Score       Method Clusters
APN 0.0478101 hierarchical        2
AD  1.5271789          pam        6
ADM 0.1400795          pam        6
FOM 0.5158032          pam        6
```

For the APN measures, hierarchical clustering with two clusters again gives the best score. However, for the other three measures PAM with six clusters has the best score. It is illustrative to graphically visualize each of the validation measures. The plot of the FOM measure is very similar to the AD measure, so we have omitted it from the figure.

```
R> par(mfrow=c(2,2),mar=c(4,4,3,1))
R> plot(stab, measure=c("APN","AD","ADM"),legend=FALSE)
R> plot(nClusters(stab),measures(stab,"APN")[,,1],type="n",axes=F,
+         xlab="",ylab="")
R> legend("center", clusterMethods(stab), col=1:9, lty=1:9, pch=paste(1:9))
R> par(op)
```

The plots of the APN, AD, and ADM are given in Figure 2. The APN measure shows an interesting trend, in that it initially increases from two to four clusters but subsequently decreases afterwards. Though hierarchical clustering with two clusters has the best score, PAM with six clusters is a close second. The AD and FOM measures tend to decrease as the number of clusters increases. Here PAM with six clusters has the best overall score, though the other algorithms have similar scores. For the ADM measure PAM with six clusters again has the best score, though the other methods outperform PAM for smaller numbers of clusters.

4.3 Biological Validation

There are two options for biological validation using the BHI and BSI measures. The first option is to explicitly specify the functional clustering of the genes. This requires the user to predetermine the functional classes of the genes, e.g. using an annotation software package like FatiGO (Al-Shahrour

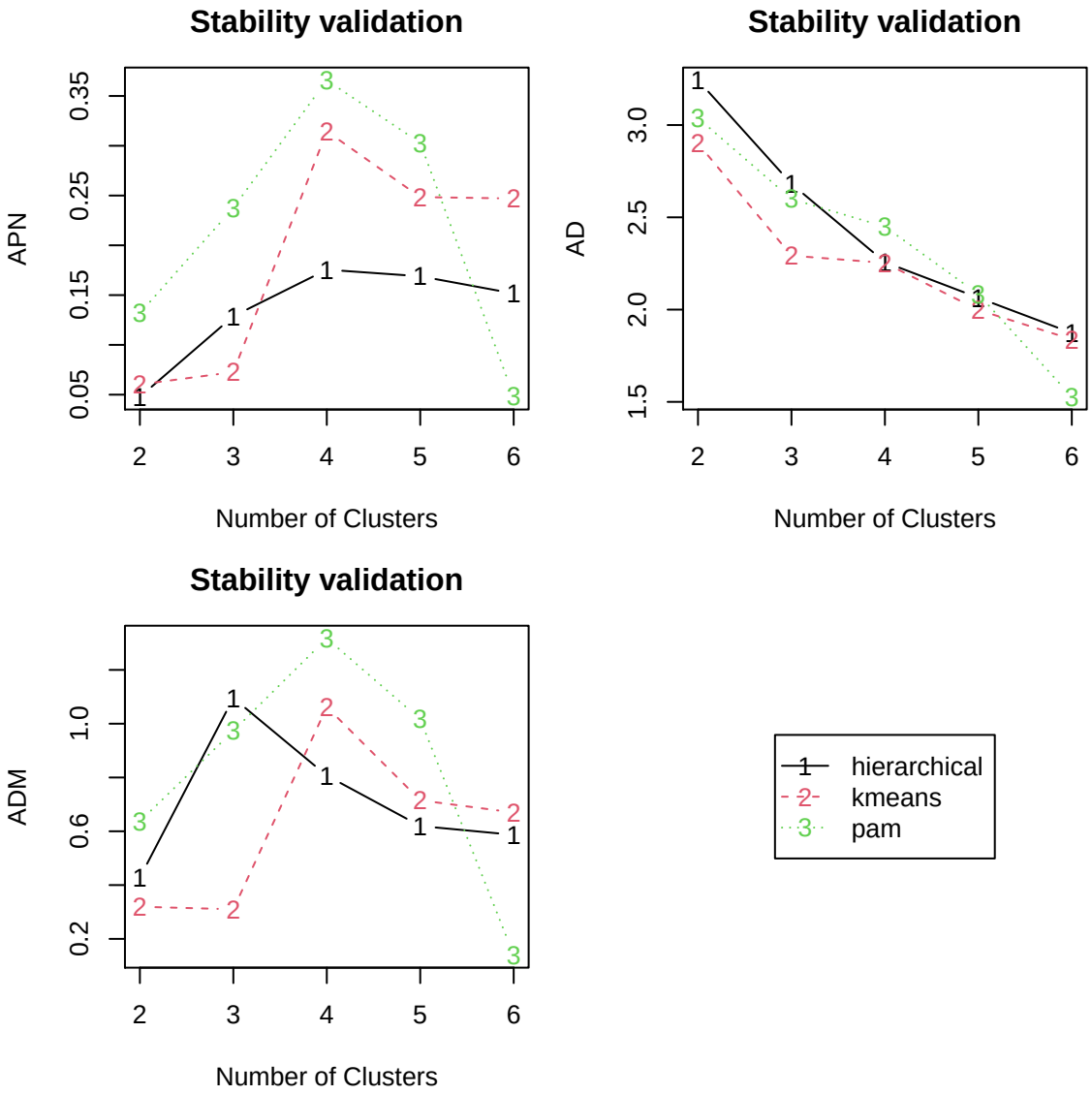

Figure 2: Plot of the APN, AD, and APN measures.

et al., 2004) or FunCat (Ruepp et al., 2004). functional clustering of the genes can be specified via either a named list or logical matrix. In "list" format, each item in the list is a vector giving genes belonging to a particular biological class. In "matrix" format, each column is a logical vector indicating which genes belong to the biological class. `clValid` will convert the biological annotation to matrix format internally if initially given in list format.

The functional categorization of the genes in the dataset `mouse` were previously determined in Bhattacherjee et al. (2007), so these will be used initially to define the functional classes.

```
R> fc <- tapply(rownames(express),mouse$FC, c)
R> fc <- fc[!names(fc)%in%c("EST","Unknown")]
R> bio <- clValid(express, 2:6, clMethods=c("hierarchical","kmeans","pam"),
+                validation="biological", annotation=fc)
```

The user also has the option of reading in the biological annotation from an external file. The required format is comma separated, with the first column indicating the biological functional category, and the remaining columns containing the gene identifiers for those genes belonging to that category. Comma separated files are easily created from within the Excel program, and using 'CSV (Comma delimited)' for the 'Save as type' in the 'Save As' window. The file "fc.csv" is included as an illustration, the format of the first two lines is:

```
ECM_Receptors, 1452671_s_at, 1423110_at, 1439381_x_at, 1450857_a_at
Growth_Differentiation, 1448995_at, 1448147_at, 1421180_at, 1416855_at
```

To read in the external biological annotation file, use the function `readAnnotationFile`.

```
R> fc2 <- readAnnotationFile("fc.csv")
R> ## bio.fc2 <- clValid(express, 2:6, clMethods=c("hierarchical","kmeans","pam"),
R> ##                    validation="biological", annotation=fc2)
R> ## all.equal(measures(bio), measures(bio.fc2))
```

Recall that both the BHI and BSI should be maximized. The optimal values for each measure are given below.

```
R> optimalScores(bio)
```

```
        Score       Method Clusters
BHI 0.2025793       kmeans        5
BSI 0.6755826 hierarchical        2
```

K-means clustering with five clusters has the best value of the BHI, while for the BSI hierarchical clustering with two clusters again does well. Plots of the measures are given in Figures 3 and 4.

The other option for biological validation is to use the annotation packages available in Bioconductor (Gentleman et al., 2004). This option uses the annotation packages to map the genes to their corresponding GO terms. There are three main ontologies, cellular component ("CC"), biological process ("BP"), and molecular function ("MF"), which can be selected via the `GOcategory` argument. The user must download, at a minimum, the *Biobase*, *annotate*, and *GO* packages from Bioconductor (http://www.bioconductor.org/), then load them during the R session. In addition, any specific annotation packages that are required will need to be downloaded (e.g., experiments using the Affymetrix GeneChip hgu95av2 would require the *hgu95av2* package). Once the appropriate annotation packages are downloaded, they can be specified in the function call via the `annotation` argument. The `goTermFreq` argument is used to select a threshold, so that only GO terms with a frequency in the dataset above the threshold are used to determine the functional classes.

To illustrate, the identifiers in the dataset `mouse` are from the Affymetrix Murine Genome 430a GeneChip Array, with corresponding annotation package *moe430a.db* available from Bioconductor. We leave the `goTermFreq` argument at its default level of 0.05, and use all available GO categories (`GOcategory="all"`) for annotation.

```
R> if(require("Biobase", quietly = TRUE) && require("annotate", quietly = TRUE) &&
+    require("GO.db", quietly = TRUE) && require("moe430a.db", quietly = TRUE)) {
+   ## Need to know which affy chip was used in experiment
+   ## affymetrix murine genome 430a genechip arrays
+   bio2 <- clValid(express, 2:6, clMethods=c("hierarchical","kmeans","pam"),
+                   validation="biological",annotation="moe430a.db",GOcategory="all")
+ }
R>

R> if(exists("bio2")) optimalScores(bio2)

        Score       Method Clusters
BHI 0.3205595 hierarchical        2
BSI 0.7559907 hierarchical        2
```

```
R> plot(bio, measure="BHI", legendLoc="topleft")
```

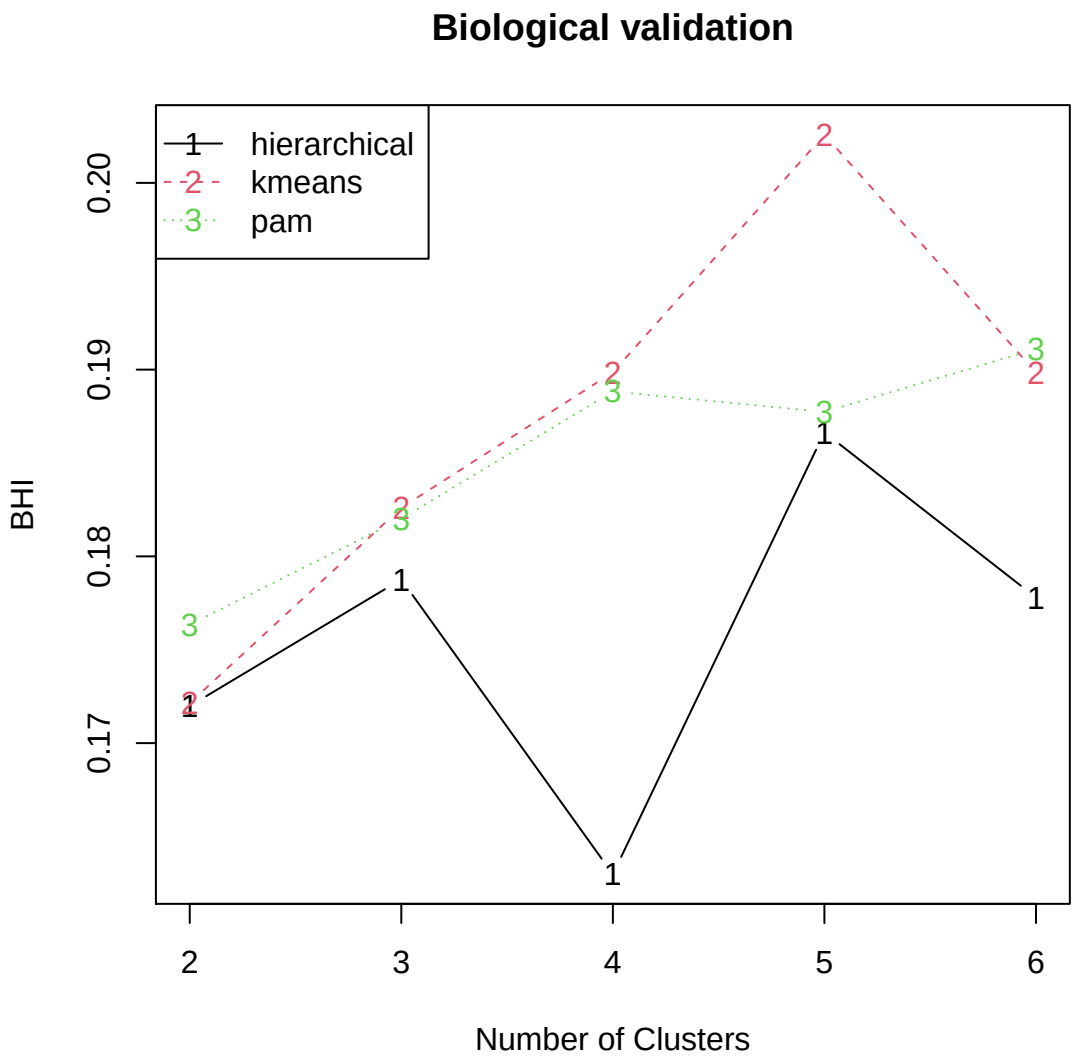

Figure 3: Plot of the BHI measure, using predetermined functional classes.

```
R> plot(bio, measure="BSI")
```

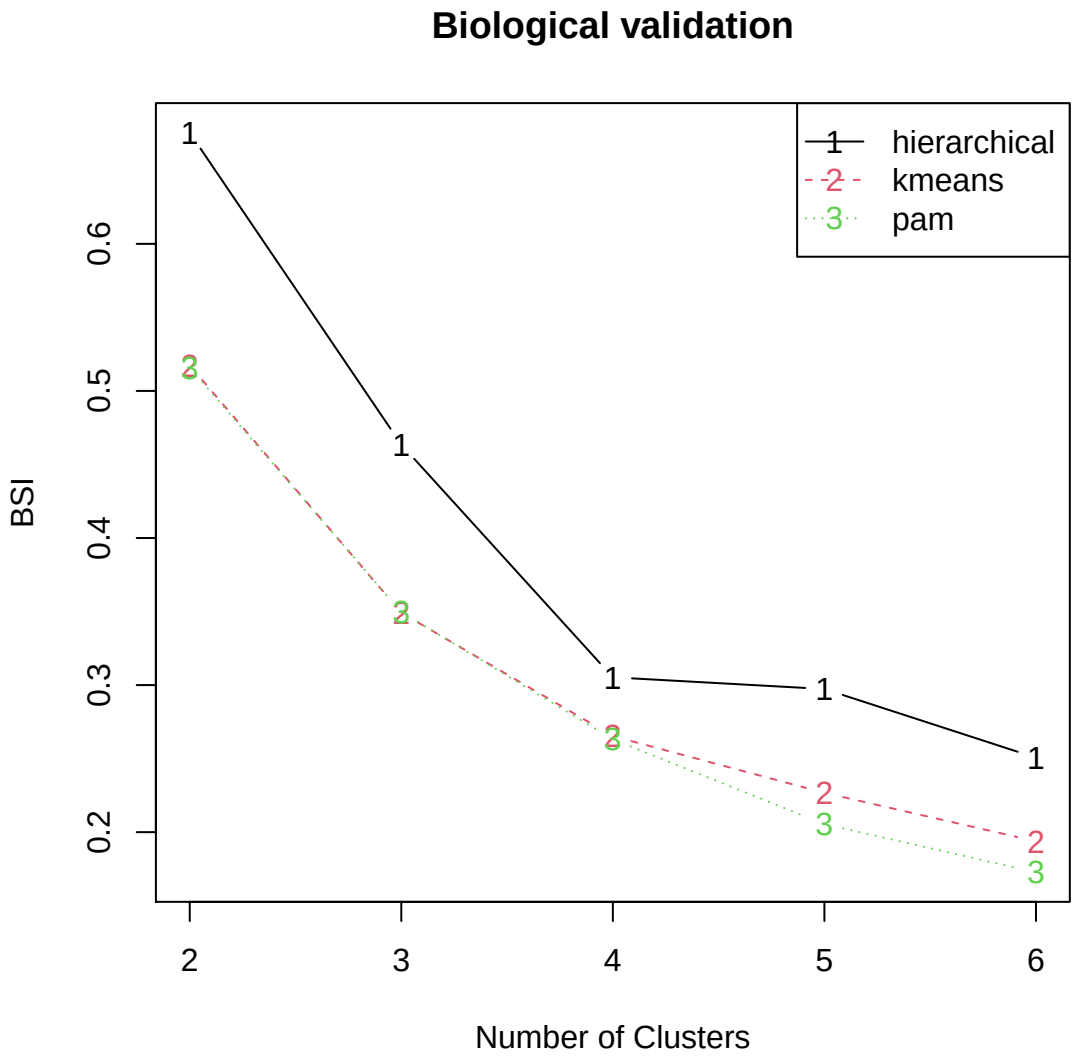

Figure 4: Plot of the BSI measure, using predetermined functional classes.

Further plots can be obtained by running

```
R> if(exists("bio2")) plot(bio2, measure="BHI", legendLoc="topleft")
R> if(exists("bio2")) plot(bio2, measure="BSI")
```

The user can also limit the level of evidence used to determine GO annotation using the `dropEvidence` argument. In particular, the "IEA" evidence code is a relatively weak association based only on electronic information, and users may wish to omit this evidence when determining the functional annotation classes.

```
R> if(require("Biobase", quietly = TRUE) && require("annotate", quietly = TRUE) &&
+    require("GO.db", quietly = TRUE) && require("moe430a.db", quietly = TRUE)) {
+   ## Need to know which affy chip was used in experiment
+   ## affymetrix murine genome 430a genechip arrays
+   bio2DE <- clValid(express, 2:6, clMethods=c("hierarchical","kmeans","pam"),
+                     validation="biological",annotation="moe430a.db",GOcategory="all",
+                     dropEvidence="IEA")
+   optimalScores(bio2DE)
+ }
```

4.4 Rank Aggregation

As we saw in all three examples, the order of clustering algorithms on each validation measure is rarely the same. Rank aggregation is helpful in reconciling the ranks and producing a "super"-list, which determines the overall winner and also ranks all the clustering algorithms based on their performance as determined by all the validation measures simultaneously. This idea was introduced in the clustering context by Pihur et al. (2007), and the R package *RankAggreg* has functions for performing rank aggregation. The package and associated functions are described in fuller detail in Pihur et al. (2009).

To illustrate, we cluster the `mouse` microarray data using the hierarchical, K-means, and PAM algorithms with four to six clusters. Both internal and stability measures are used for validation.

```
R> result <- clValid(express, 4:6, clMethods=c("hierarchical","kmeans","pam"),
+                    validation=c("internal","stability"))
R> res <- getRanksWeights(result)
```

The `getRanksWeights` function extracts the validation measures and order of the clustering algorithms for each validation measure to use as input

for RankAggreg. The validation measures are used for calculating weighted distances.

The top three ranking algorithms for each measure are given below:

```
R> print(res$ranks[,1:3], quote=FALSE)
```

```
             1              2              3
APN          pam-6          hierarchical-6 hierarchical-5
AD           pam-6          kmeans-6       hierarchical-6
ADM          pam-6          hierarchical-6 hierarchical-5
FOM          pam-6          kmeans-6       hierarchical-6
Connectivity hierarchical-4 hierarchical-5 hierarchical-6
Dunn         hierarchical-5 hierarchical-6 hierarchical-4
Silhouette   pam-6          hierarchical-4 kmeans-4
```

We see that PAM with six clusters performs best on four of the six measures, so picking an overall winner is relatively straightforward in this case. However, in many cases there is no clearly best performing algorithm. Also, it would be rather difficult to give the overall ordered list, and we may be interested in, say, the top three performing algorithms instead of restricting ourselves to a single set of results. This can be accomplished using the RankAggreg function, which searches for the "master" list which minimizes the distance between itself and the individual lists for each validation measure. Two distances are available, the Spearman's footrule distance and Kendall's tau distance. The two available search algorithms are a cross-entropy Monte Carlo algorithm or a genetic algorithm (there is also a "brute force" search method which does an exhaustive search, but this is not recommended except for very small lists of five or fewer elements). Here, we perform rank aggregation using the default cross-entropy method with weighted Spearman's footrule to produce a "top five" overall list.

```
R> if(require("RankAggreg")) {
+   CEWS <- RankAggreg(x=res$ranks, k=5, weights=res$weights, seed=123, verbose=FALSE
+   CEWS
+ }
```

```
The optimal list is:
        pam-6 hierarchical-6 hierarchical-5 hierarchical-4 kmeans-6

  Algorithm:   CE
  Distance:    Spearman
  Score:       2.679264
```

The top results are PAM with six clusters, then hierarchical clustering with six and five clusters. Convergence was achieved in 15 iterations, with a minimum objective function score of 2.679264. Since the search is stochastic using a different seed may produce a different result, but repeating the search using several different seeds gave the same result.

To get a visual representation of the results, a convenient *plot* function is provided. It takes the object returned by the *RankAggreg* function as its first argument and outputs three side-by-side plots with useful information on the convergence properties and the final ranking.

4.5 Further Analysis

Hierarchical clustering consistently performs well for many of the validation measures. The clustering results from any method can be extracted from a `clValid()` object for further analysis, using the `clusters()` method. Here, we extract the results from hierarchical clustering, to plot the dendogram and view the observations that are grouped together at the various levels of the topology. The dendrogram is plotted in Figure 6, with the genes belonging to the "Growth/Differentiation" (GD) and "Transcription factor" (TF) functional classes labeled. The genes belonging to the top two clusters are cross-classified with their functional annotation given in the dataset. Of potential interest, the second cluster contains no genes in the "EST" or "Miscellaneous" categories. Further inspection of the results is left to a subject matter expert.

```
R> hc <- clusters(bio,"hierarchical")

R> mfc <- factor(mouse$FC, labels=c("Re","EST","GD","KP","Met","Mis","St","TF","U"))
R> tf.gd <- ifelse(mfc%in%c("GD","TF"),levels(mfc)[mfc],"")
R> plot(hc, labels=tf.gd, cex=0.7, hang=-1, main="Mouse Cluster Dendrogram")


R> two <- cutree(hc,2)
R> xtabs(~mouse$FC + two)

                        two
mouse$FC                  1  2
  ECM/Receptors          12  4
  EST                    31  0
  Growth/Differentiation 12  4
  Kinases/Phosphatases    4  3
```

```
R> plot(CEWS)
```

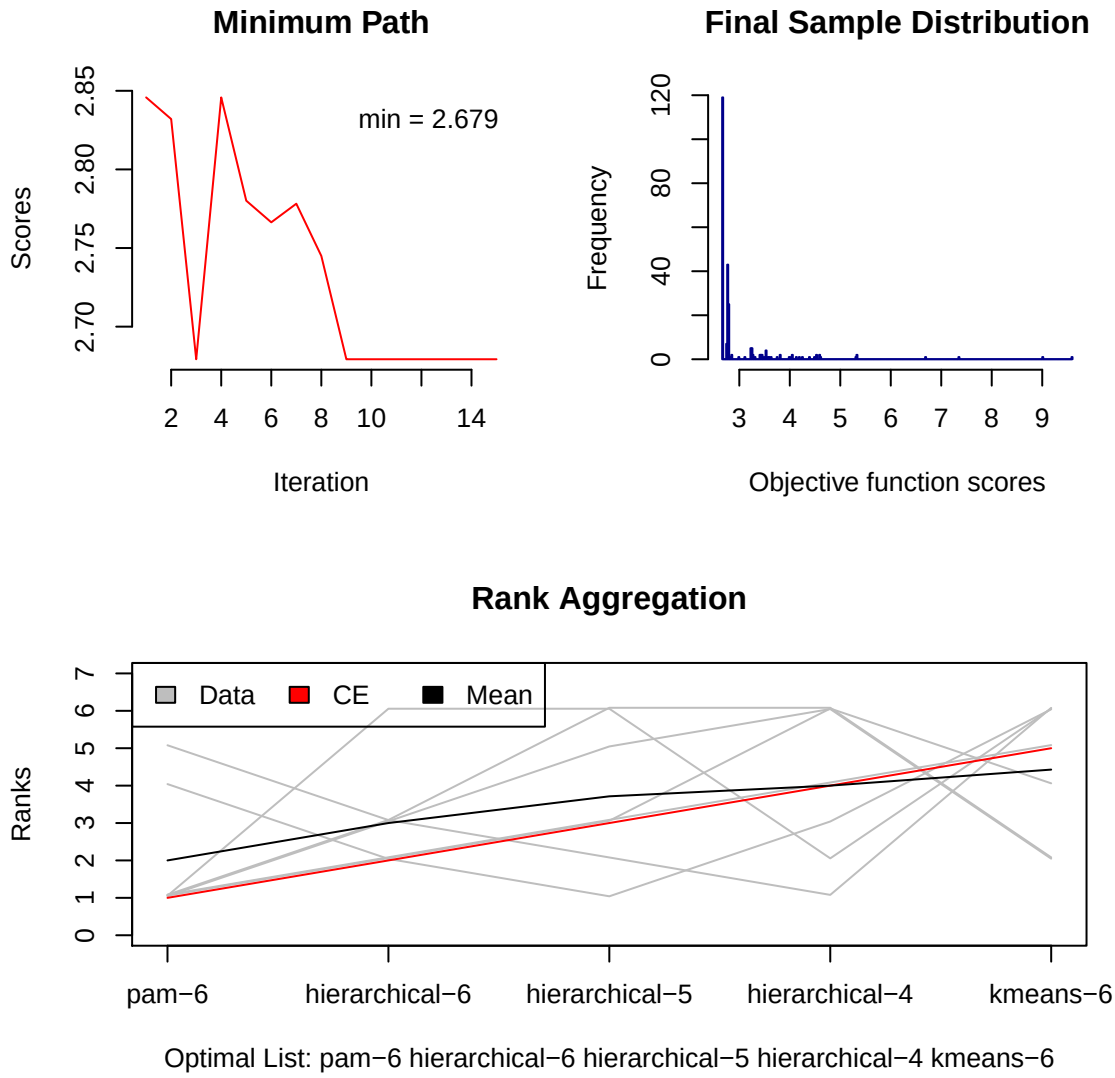

Figure 5: Visual Representation of the aggregation results through the *plot()* function. The first plot in the top row shows the path of minimum values of the objective function over time. The global minimum is shown in the top right corner. The histogram of the objective function scores at the last iteration is displayed in the second plot. Looking at these two plots, one can get a general idea about the rate of convergence and the distribution of candidate lists at the last iteration. The third plot at the bottom shows the individual lists and the obtained solution along with optional average ranking.

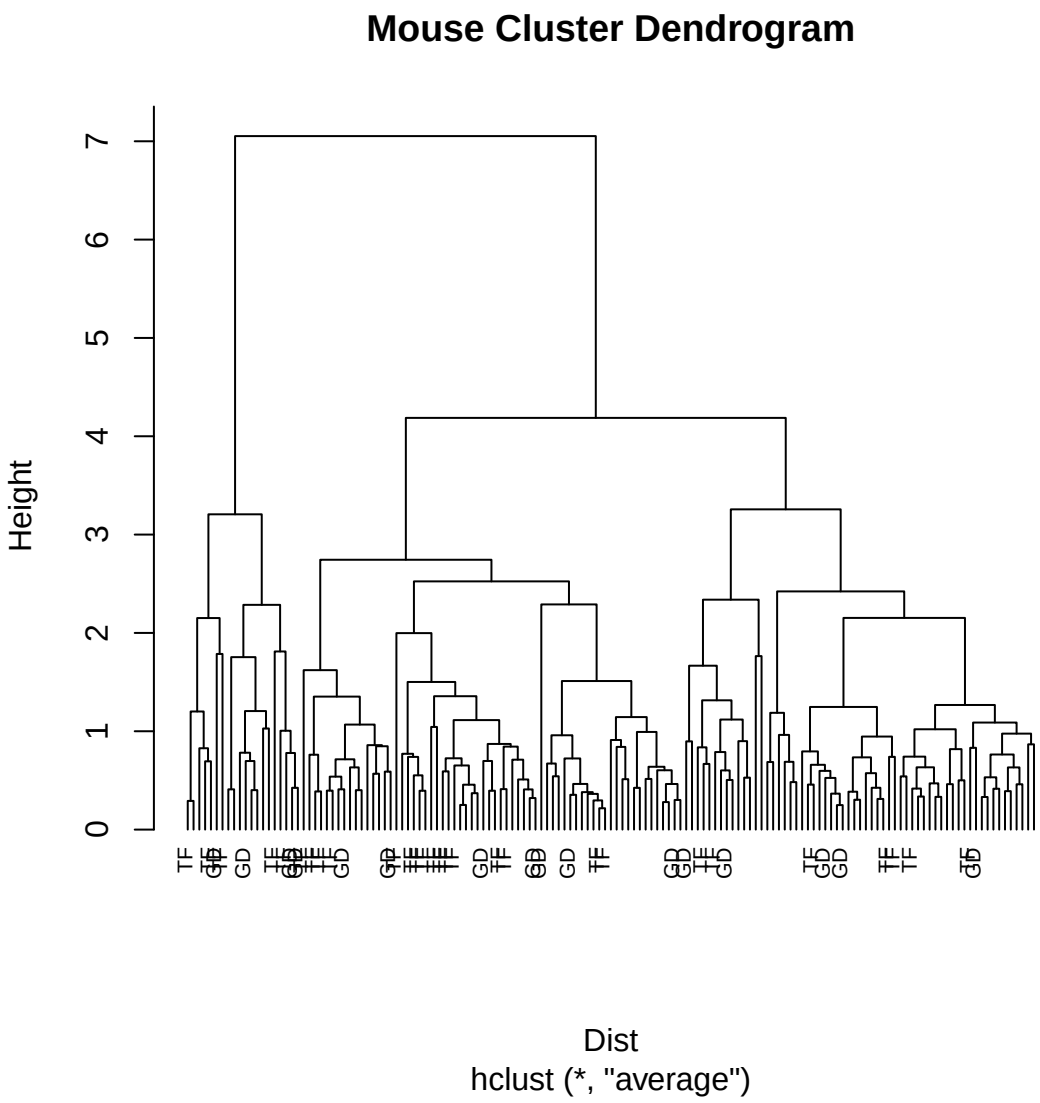

Figure 6: Plot of the dendogram for hierarchical clustering.

```
Metabolism              7  1
Miscellaneous          25  0
Stress-induced          4  2
Transcription factor   23  5
Unknown                 9  1
```

5 Discussion

We have developed an R package, *clValid*, which contains measures for validating the results from a clustering procedure. We categorize the measures into three distinct types, "internal", "stability", and "biological", and provide plot, summary, and additional methods for viewing and summarizing the validation scores and extracting the clustering results for further analysis. In addition to the object-oriented nature of the language, implementing the validation measures within the R statistical programming framework provides the additional advantage in that it can interface with numerous clustering algorithms in existing R packages, and accommodate further algorithms as they are developed and coded into R libraries. Currently, `clValid()` accepts up to ten different clustering methods. This permits the user to simultaneously vary the number of clusters and the clustering algorithms to decide how best to group the observations in her/his dataset. Lastly, the package makes use of the annotation packages available in Bioconductor to calculate the biological validation measures, so that the information contained in the GO database can be used to assist in the cluster validation process.

The illustration for the *clValid* package we have given here focuses on clustering genes, but it is common in microarray analysis to cluster both genes and samples to create a "heatmap". Though the "biological" validation measures are specifically designed for validation of clustering genes, the other measures could also be used with clustering of samples in a microarray experiment. Also, for microarray data, it is a good idea to limit the number of genes being clustered to a small subset ($100 \sim 600$) of the thousands of expression measures routinely available on a microarray, both for computational and visualization purposes. Typically, some initial pre-selection of the genes based on t-statistics, p-values, or expression ratios is performed.

There are several R packages that also perform cluster validation and are available from CRAN or Bioconductor. Examples include the `clustIndex()` function in package *cclust* (Dimitriadou, 2006), which performs 14 different validation measures in three classes, `cluster.stats()` and `clusterboot()` in package *fpc* (Hennig, 2006), the *clusterRepro* (Kapp and Tib-

shirani, 2006) and *clusterSim* (Walesiak and Dudek, 2007) packages, and the *clusterStab* (MacDonald et al., 2006) package from Bioconductor. The `cl_validity()` function in package *clue* (Hornik, September 2005b) does validation for both paritioning methods ("dissimilarity accounted for") and hierarchical methods ("variance accounted for"), and function `fclustIndex()` in package *e1071* (Dimitriadou et al., 2006) has several fuzzy cluster validation measures. However, to our knowledge none of these packages offers biological validation or the unique stability measures which we present here. Handl et al. (2005) provides C++ code for the validation measures which they discuss, and the `Caat` tool available in the GEPAS software suite offers a web-based interface for visualizing and validating (using the Silhouette Width) cluster results. However, neither of these two tools are as flexible for interfacing with the variety of clustering algorithms that are available in the R language, or can automatically access the annotation information which is available in Bioconductor. Hence, the *clValid* package is a valuable addition to the growing collection of cluster validation software available for researchers.

References

F. Al-Shahrour, R. Diaz-Uriarte, and J. Dopazo. FatiGo: a web tool for finding significant associations of Gene Ontology terms with groups of genes. *Bioinformatics*, 20(4):578–80, 2004.

V. Bhattacherjee, P. Mukhopadhyay, S. Singh, C. Johnson, J. T. Philipose, C. P. Warner, R. M. Greene, and M. M. Pisano. Neural crest and mesoderm lineage-dependent gene expression in orofacial development. *Differentiation*, 2007.

N. Bolshakova, F. Azuaje, and P. Cunningham. A knowledge-driven approach to cluster validity assessment. *Bioinformatics*, 21(10):2546–7, 2005.

G. Brock, V. Pihur, S. Datta, and S. Datta. clValid: An R package for cluster validation. *Journal of Statistical Software*, 25(4), March 2008. URL `http://www.jstatsoft.org/v25/i04`.

S. Chu, J. DeRisi, M. Eisen, J. Mulholland, D. Botstein, P. O. Brown, and I. Herskowitz. The transcriptional program of sporulation in budding yeast. *Science*, 282(5389):699–705, 1998.

S. Datta and S. Datta. Comparisons and validation of statistical clustering techniques for microarray gene expression data. *Bioinformatics*, 19(4):459–66, 2003.

S. Datta and S. Datta. Methods for evaluating clustering algorithms for gene expression data using a reference set of functional classes. *BMC Bioinformatics*, 7: 397, 2006.

D. Dembele and P. Kastner. Fuzzy C-means method for clustering microarray data. *Bioinformatics*, 19(8):973–80, 2003.

J. L. DeRisi, V. R. Iyer, and P. O. Brown. Exploring the metabolic and genetic control of gene expression on a genomic scale. *Science*, 278(5338):680–6, 1997.

E. Dimitriadou. *cclust: Convex Clustering Methods and Clustering Indexes*, 2006. URL `http://www.R-project.org`. R package version 0.6-13.

E. Dimitriadou, K. Hornik, F. Leisch, D. Meyer, and A. Weingessel. *e1071: Misc Functions of the Department of Statistics (e1071), TU Wien*, 2006. URL `http://www.R-project.org`. R package version 1.5-16.

J. Dopazo and J. M. Carazo. Phylogenetic reconstruction using a growing neural network that adopts the topology of a phylogenetic tree. *Journal of Molecular Evolution*, pages 226–233, 1997.

J. C. Dunn. Well separated clusters and fuzzy partitions. *Journal on Cybernetics*, 4:95–104, 1974.

M. B. Eisen, P. T. Spellman, P. O. Brown, and D. Botstein. Cluster analysis and display of genome-wide expression patterns. *Proc Natl Acad Sci U S A*, 95(25): 14863–8, 1998.

C. Fraley and A. E. A. E. Raftery. Enhanced model-based clustering, density estimation, and discriminant analysis software: MCLUST. *Journal of Classification*, 20(2):263–286, 2003.

C. Fraley and A. E. Raftery. Model-based clustering, discriminant analysis, and density estimation. *Journal of the American Statistical Association*, 17:126–136, 2001.

C. Fraley and A. E. Raftery. *mclust: Model-Based Clustering / Normal Mixture Modeling*, 2007. URL `http://www.stat.washington.edu/mclust`. R package version 3.1-1.

L. Fu and E. Medico. FLAME, a novel fuzzy clustering method for the analysis of DNA microarray data. *BMC Bioinformatics*, 8:3, 2007.

I. Gat-Viks, R. Sharan, and R. Shamir. Scoring clustering solutions by their biological relevance. *Bioinformatics*, 19(18):2381–9, 2003.

R. C. Gentleman, V. J. Carey, D. M. Bates, B. Bolstad, M. Dettling, S. Dudoit, B. Ellis, L. Gautier, Y. Ge, J. Gentry, K. Hornik, T. Hothorn, W. Huber, S. Iacus, R. Irizarry, F. L. C. Li, M. Maechler, A. J. Rossini, G. Sawitzki, C. Smith, G. Smyth, L. Tierney, J. Y. H. Yang, and J. Zhang. Bioconductor: Open software development for computational biology and bioinformatics. *Genome Biology*, 5: R80, 2004. URL `http://genomebiology.com/2004/5/10/R80`.

F. D. Gibbons and F. P. Roth. Judging the quality of gene expression-based clustering methods using gene annotation. *Genome Res*, 12(10):1574–81, 2002.

J. Handl, J. Knowles, and D. B. Kell. Computational cluster validation in post-genomic data analysis. *Bioinformatics*, 21(15):3201–12, 2005.

J. A. Hartigan and M. A. Wong. A k-means clustering algorithm. *Applied Statistics*, 28:100–108, 1979.

C. Hennig. *fpc: Fixed point clusters, clusterwise regression and discriminant plots*, 2006. URL `http://www.R-project.org`. R package version 1.2-2.

J. Herrero, A. Valencia, and J. Dopazo. A hierarchical unsupervised growing neural network for clustering gene expression patterns. *Bioinformatics*, 17(2):126–36, 2001.

K. Hornik. A CLUE for CLUster Ensembles. *Journal of Statistical Software*, 14 (12), September 2005b. URL `http://www.jstatsoft.org/v14/i12/`.

A. Kapp and R. Tibshirani. *clusterRepro: Reproducibility of gene expression clusters*, 2006. URL `http://www.R-project.org`. R package version 0.5-1.

L. Kaufman and P. J. Rousseeuw. *Finding Groups in Data. An Introduction to Cluster Analysis*. Wiley, New York, 1990.

M. K. Kerr and G. A. Churchill. Bootstrapping cluster analysis: assessing the reliability of conclusions from microarray experiments. *Proc Natl Acad Sci U S A*, 98(16):8961–5, 2001.

T. Kohonen. *Self-Organizing Maps*. Springer-Verlag, second edition, 1997.

J. MacDonald, D. Ghosh, and M. Smolkin. *clusterStab: Compute cluster stability scores for microarray data*, 2006. URL `http://www.R-project.org`. R package version 1.6.0.

G. J. McLachlan, R. W. Bean, and D. Peel. A mixture model-based approach to the clustering of microarray expression data. *Bioinformatics*, 18(3):413–22, 2002.

V. Pihur, S. Datta, and S. Datta. Weighted rank aggregation of cluster validation measures: a monte carlo cross-entropy approach. *Bioinformatics*, 23(13):1607–15, 2007.

V. Pihur, S. Datta, and S. Datta. Rankaggreg, an r package for weighted rank aggregation. *BMC Bioinformatics*, 10(62), 2009. URL `http://www.biomedcentral.com/1471-2105/10/62`.

R Development Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2006. URL `http://www.R-project.org`. ISBN 3-900051-07-0.

P. Rousseeuw, A. Struyf, M. Hubert, and M. Maechler. *cluster: Cluster Analysis Extended Rousseeuw et al.*, 2006. URL `http://www.R-project.org`. R package version 1.11.4.

P. J. Rousseeuw. Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. *Journal of Computational and Applied Mathematics*, 20:53–65, 1987.

A. Ruepp, A. Zollner, D. Maier, K. Albermann, J. Hani, M. Mokrejs, I. Tetko, U. Guldener, G. Mannhaupt, M. Munsterkotter, and H. W. Mewes. The FunCat, a functional annotation scheme for systematic classification of proteins from whole genomes. *Nucleic Acids Res*, 32(18):5539–45, 2004.

M. Walesiak and A. Dudek. *clusterSim: Searching for optimal clustering procedure for a data set*, 2007. URL `http://www.R-project.org`. R package version 0.30-7.

R. Wehrens. *kohonen: Supervised and unsupervised self-organising maps*, 2006. URL `http://www.R-project.org`. R package version 1.1.1.

K. Y. Yeung, D. R. Haynor, and W. L. Ruzzo. Validating clustering for gene expression data. *Bioinformatics*, 17(4):309–18, 2001.