5-2015

OptCluster : an R package for determining the optimal clustering algorithm and optimal number of clusters.

Michael N. Sekula
University of Louisville

OPTCLUSTER: AN R PACKAGE FOR DETERMINING THE OPTIMAL CLUSTERING ALGORITHM AND OPTIMAL NUMBER OF CLUSTERS

By

Michael N. Sekula
B.A., Saginaw Valley State University, 2010

A Thesis
Submitted to the Faculty of the
School of Public Health and Information Sciences of the University of Louisville
in Partial Fulfillment of the Requirements
for the Degree of

Master of Science in Biostatistics: Decision Science

Department of Bioinformatics and Biostatistics
University of Louisville
Louisville, Kentucky

May 2015

OPTCLUSTER: AN R PACKAGE FOR DETERMINING THE OPTIMAL CLUSTERING ALGORITHM AND OPTIMAL NUMBER OF CLUSTERS

By

Michael N. Sekula
B.A., Saginaw Valley State University, 2010

A Thesis Approved on

April 9, 2015

by the following Thesis Committee:

ABSTRACT

OPTCLUSTER: AN R PACKAGE FOR DETERMINING THE OPTIMAL CLUSTERING ALGORITHM AND OPTIMAL NUMBER OF CLUSTERS

Michael N. Sekula

April 9, 2015

Determining the best clustering algorithm and ideal number of clusters for a particular dataset is a fundamental difficulty in unsupervised clustering analysis. In biological research, data generated from Next Generation Sequencing technology and microarray gene expression data are becoming more and more common, so new tools and resources are needed to group such high dimensional data using clustering analysis. Different clustering algorithms can group data very differently. Therefore, there is a need to determine the best groupings in a given dataset using the most suitable clustering algorithm for that data. This paper presents the R package **optCluster** as an efficient way for users to evaluate up to ten clustering algorithms, ultimately determining the optimal algorithm and optimal number of clusters for a given set of data. The selected clustering algorithms are evaluated by as many as nine validation measures classified as "biological", "internal", or "stability", and the final result is obtained through a weighted rank aggregation algorithm based on the calculated validation scores. Two examples using this package are presented, one with a microarray dataset and the other with an RNA-Seq dataset. These two examples highlight the capabilities the **optCluster** package and demonstrate its usefulness as a tool in cluster analysis.

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER I

INTRODUCTION

Research dealing with high dimensional data, such as microarray gene expression data, data generated from Next Generation Sequencing (NGS) technology, and mass spectrometry data, are commonplace in biomedical sciences. Just to summarize them in an unsupervised manner, cluster analysis plays an important role. The unsupervised technique of clustering organizes data by assigning similar observations together into the same group when "little" or "no" other information is known about the data. For example, not only do biologists need to expose underlying structures inside large microarray datasets, but they also need to do so in an optimal way that will create groups of genes with similar biological functions. However, the number of choices for clustering algorithms is vast and different algorithms can provide different results on the same data. Choosing the optimal clustering algorithm along with the optimal cluster size (number of clusters) for a given dataset becomes an overwhelming task. For this paper, the terms "cluster size" and "number of clusters" will be considered synonymous and will be used interchangeably.

The process of clustering can essentially be broken down into three steps: pre-processing, cluster analysis, and cluster validation (Handl et al., 2005). The first step, pre-processing, deals with transforming the dataset to improve the likelihood that similar observations will be grouped together. In the second step of the clustering process, parameters and clustering techniques are chosen and then applied to the data. Cluster validation, the third step, evaluates the performance of the selected clustering algorithms. This clustering process is cyclic and can repeat itself many times as different choices in any of the steps will result in different conclusions.

Cluster validation has become an increasingly important step in determining the most appropriate clustering algorithm given a dataset, especially when working with high dimensional data such as microarray data or NGS data. Validation measures serve as guidance to choosing the appropriate clustering algorithm for a dataset by providing performance evaluations based on some particular criteria such as compactness, separation, or biological homogeneity. Internal validation and external validation are the two major classes of cluster validation measures (Handl et al., 2005). The main difference between these two categories is whether or not the measurement utilizes additional information outside of the data in its validation technique. In many cases, there is “little” to “no” information known about the data so internal validation is the only option.

Handl et al. (2005) recommends using multiple validation measures to compare clustering algorithms while in the process of determining the “best” clustering algorithm. The inherent problem with using multiple validation measures is that an algorithm that performs well with one measure may perform poorly with another. When a researcher is comparing a large number of clustering algorithms and using multiple validation measures, the results become muddled and determining the optimal clustering algorithm visually from a plot (based on the validation scores for different number of clusters) becomes unclear.

There has been some recent research in the literature dealing with cluster analysis. Several of these works attempted to identify the types of clustering methods and validation measures that perform the best in a given situation. In 2006, Thalamuthu et al. compared six clustering algorithms commonly used for microarray analysis. For both simulated and real data, it was determined that tight clustering (Tseng & Wong, 2005) and model-based clustering (Fraley & Raftery, 2002) were the top performing algorithms, while SOM (Kohonen, 2001) and hierarchical clustering (Anderberg, 1973; Sneath & Sokal, 1973) had the worst performances. Rendón et al. (2011) used the clustering algorithms K-means (Hartigan & Wong, 1979) and Bisecting K-means (Theodoridis & Koutroumbas, 2006) to compare internal and external validation measures. The purpose of this study was to determine which type of validation measure was better at correctly identifying the true number of clusters within a dataset. Using thirteen different datasets, internal validation indices were concluded to be more accurate. An extensive study was performed by Arbelaitz et al. (2013) to assess the performance of thirty unique validation measures. While it was determined that there was not a single validation measure that outperformed the rest in every situation, the silhouette index (Rousseeuw, 1987) was noted as a high performer for many of situations evaluated.

Many of the recent clustering algorithms found in the literature have been developed for use in the cluster analysis of high dimensional data. Using a proposed Poisson dissimilarity matrix, Witten (2011) introduced a hierarchical algorithm for clustering RNA-Seq data. Other algorithms have been presented as improvements to the K-means algorithm in order to increase its performance when clustering genes and gene expression data (Wu, 2008; Lam & Tsang, 2012; Nazeer et al., 2013). Mavridis et al.

proposed a partitioning-based clustering algorithm called PFClust (Parameter Free Clustering) in 2013. This unique algorithm clusters data and determines an ideal number of clusters without requiring the user to specify any parameters. In 2014, Si et al. described several clustering algorithms based on probability models for RNA-Seq data. These new algorithms were able to provide better clustering results than the commonly used methods of hierarchical clustering, K-means, and SOM for both simulated and real data.

R Packages

A popular statistical resource for researchers in the field of biomedical sciences is the open source R software environment (R Core Team, 2014). Because this software is open sourced, new packages extending the statistical capabilities of R are developed and become readily accessible to all users through repositories such as the Comprehensive R Archive Network (CRAN) and Bioconductor (Gentleman et al., 2004). A variety of R packages providing tools for cluster analysis can be found at these repositories. Many of these packages offer functions that calculate cluster validation measures, with some popular examples including **clValid** (Brock et al., 2011), **clv** (Nieweglowski, 2013), **cclust** (Dimitriadou 2014), **clusterSim** (Walesiak & Dudek, 2014), and **fpc** (Hennig, 2015). The **RankAggreg** package (Pihur et al., 2009) can take ranked lists of clustering algorithms and combine them into an overall optimal list with the "best" clustering algorithm placed in the first position. The package **NbClust** (Charrad et al., 2014) provides two clustering algorithms and thirty cluster validation measures to determine the relevant number of clusters in a dataset. The "best" choice for number of clusters is determined by a majority rule. The **COMMUNAL** package (Chen et al., 2015) determines the optimal k number of clusters and then creates a best clustering assignment based on overlap between clustering results for up to fourteen different clustering algorithms.

It is evident that there are numerous cluster analysis R packages, however this paper will focus on two specific packages, **clValid** and **RankAggreg**, while presenting a new package called **optCluster**. The **clValid** package offers nine different cluster validation measures and can create ranked lists of clustering algorithms based on the calculated validation scores for each validation measure chosen. Using these lists, the **RankAggreg** package is able to perform weighted rank aggregation and obtain an optimal ranked list of clustering algorithms such that the first clustering algorithm in this list is the "best" clustering algorithm for that specific dataset across all the cluster validation measures and all the different cluster sizes that the user has selected. Capitalizing on how well these two packages work together, the **optCluster** package determines an optimal clustering algorithm and optimal number of clusters for a given dataset by combining functions from both the **clValid** and **RankAggreg** packages into one single, easy to use function.

The R package **clValid** was developed to provide researchers with a useful resource for cluster validation. This package offers nine different validation measures, classified as "biological", "internal", or "stability", and ten different clustering algorithms for use in cluster analysis. The user may select the type (or types) of validation measures to use, define the number (or range of numbers) of clusters to create, and select any number of clustering algorithms to evaluate in a single function call. For a given dataset, the *clValid( )* function calculates the validation measure scores for each selected clustering algorithm with each desired cluster size. This function provides an output of the optimal cluster algorithm and optimal number of clusters for each validation measure. These optimal choices are determined by maximizing or minimizing the validation scores of multiple clustering algorithms, with multiple cluster sizes, for each of the given validation methods.

Clustering algorithms can be placed in a list of ranks according to their performances given a particular validation measure. When using multiple validation measures, there are multiple ranked lists. However, given one dataset, it is desirable to have a unique answer with one optimized list. A weighted rank aggregation method was proposed by Pihur et al. (2007) as a way to achieve a unique list, given a dataset, using a stochastic optimization technique. The idea is that the ranked lists from multiple validation measures can be combined and analyzed with a Monte Carlo cross-entropy approach (Pihur et al., 2007) and an overall optimal ranked list is produced that is as similar as possible to the ordered lists created by each validation measure. From this optimal list, the "best" clustering algorithm for a specific dataset can then be obtained.

An R package called **RankAggreg** was developed by Pihur et al. (2009) to provide users with the Monte Carlo cross-entropy method for combining ordered lists. A second stochastic optimization method, the Genetic algorithm (Goldberg, 1989), was also included in this package as another option for rank aggregation. A matrix of ordered rows (clustering algorithms) as well as a matrix of weights (validation measure scores) can be input into the function *RankAggreg( )* for evaluation, and an optimal ordered list of clustering algorithms is provided in the output.

Both the **clValid** and **RankAggreg** packages address different challenges in cluster analysis. While these two packages are independent from another, they can be used in conjunction to obtain an optimal clustering algorithm and optimal number of clusters for a given dataset. The *clValid( )* function is able to calculate validation scores for multiple clustering algorithms and the *getRanksWeights( )* function (also available in the package **clValid**) can create a matrix of clustering algorithm ranks and a matrix of validation score weights. The matrices of ranks and weights can then be entered into the *RankAggreg( )* function to obtain an optimal list of algorithms based on the chosen weighted rank aggregation method. Since these two individual packages work so well with each other, it seems logical to combine them in some way, ultimately minimizing the amount of code needed to run the entire process of finding a "best" clustering algorithm and also the optimal number of cluster size for clustering a particular dataset.

This work introduces an R package called **optCluster** that determines the optimal clustering algorithm and optimal cluster size for a given dataset. In this package, the function *optCluster( )* utilizes the capabilities of the functions *clValid( )*, *getRanksWeights( )*, and *RankAggreg( )* to obtain an optimal result. Figure 1 displays two side-by-side flowcharts describing different procedures an R user would have go through in order to obtain the optimal clustering algorithm and optimal number of clusters. In these flowcharts, ellipses indicate internal processes and rectangles denote output that the user can obtain. The procedure on the left uses the functions from the **clValid** and **RankAggreg** packages, while the procedure on the right simply uses the *optCluster( )* function. From this figure, we see that the user only has to enter the dataset and arguments from the *clValid( )* and *RankAggreg( )* functions into the *optCluster( )*

function directly and the rest of the procedure is carried out internally. Therefore, the number of steps one needs to take in order to obtain the final result is reduced with this new function.

The **optCluster** package is able to cluster and obtain the optimal clustering algorithm and optimal cluster size for any microarray, RNA-Seq, or protein expression data. To the best of the our knowledge, an R package that combines cluster validation measures, including biological validation, with weighted rank aggregation to determine these optimal results is currently not available.

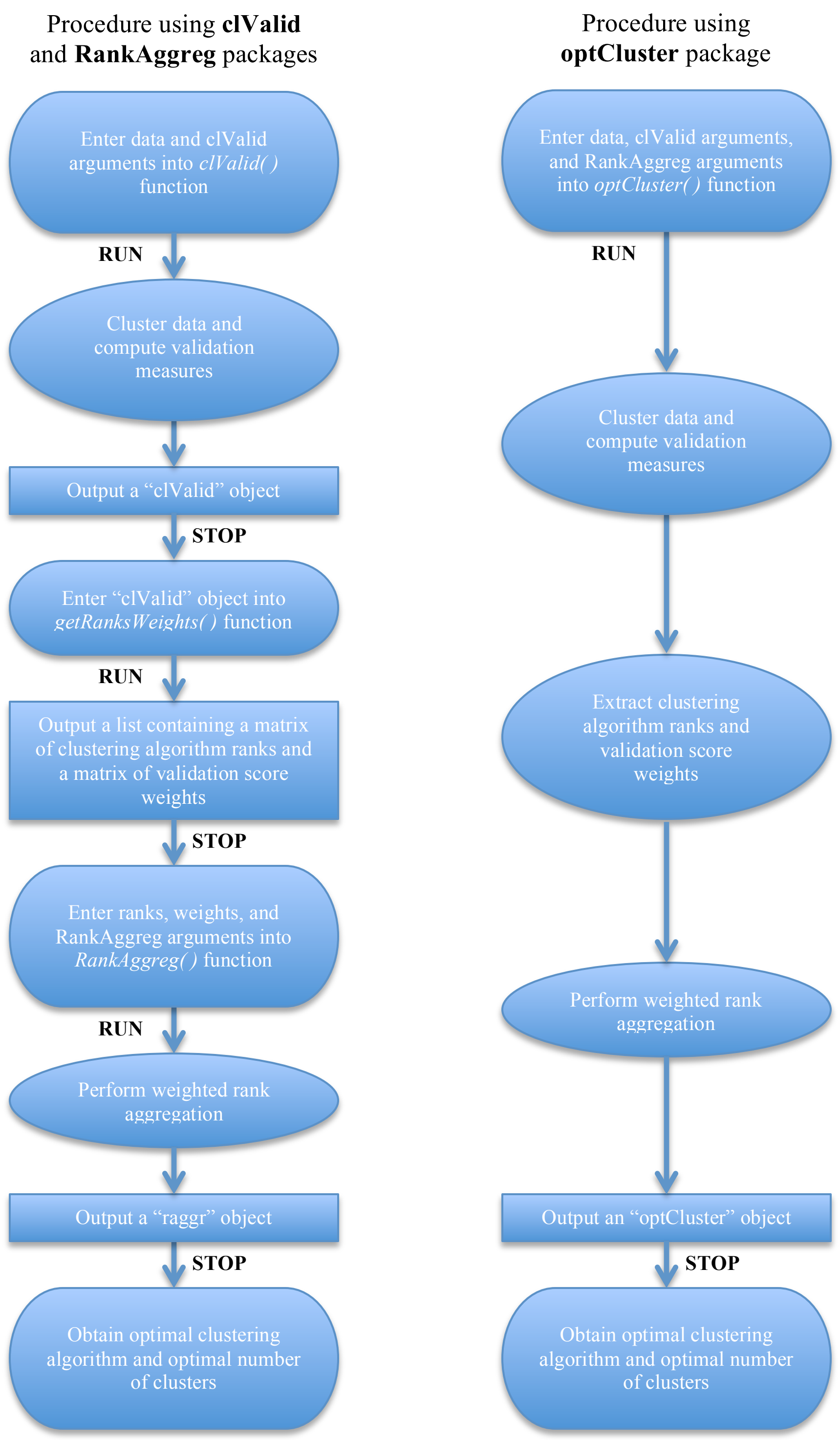
Procedure using clValid and RankAggreg packages
Enter data and clValid arguments into clValid( ) function
RUN
Cluster data and compute validation measures
Output a "clValid" object
STOP
Enter "clValid" object into getRanksWeights( ) function
RUN
Output a list containing a matrix of clustering algorithm ranks and a matrix of validation score weights
STOP
Enter ranks, weights, and RankAggreg arguments into RankAggreg( ) function
RUN
Perform weighted rank aggregation
Output a "raggr" object
STOP
Obtain optimal clustering algorithm and optimal number of clusters
Procedure using optCluster package
Enter data, clValid arguments, and RankAggreg arguments into optCluster( ) function
RUN
Cluster data and compute validation measures
Extract clustering algorithm ranks and validation score weights
Perform weighted rank aggregation
Output an "optCluster" object
STOP
Obtain optimal clustering algorithm and optimal number of clusters

Figure 1: Two flowcharts comparing procedures for determining an optimal clustering algorithm and optimal number of clusters. The chart on the left explains the procedure using both the **clValid** and **RankAggreg** packages. The chart on the right explains the procedure using only the **optCluster** package.

CHAPTER II

CLUSTERING ALGORITHMS

The package **optCluster** provides a total of ten popular clustering algorithm options that are obtained through the package **clValid**. The **clValid** package itself contains a function for the SOTA algorithm (Dopazo & Carazo, 1997; Herrero et al., 2001) and utilizes other R packages for the remaining nine algorithms. The model-based algorithm is included the **mclust** package (Fraley et al., 2014), and the self-organizing maps (SOM) algorithm is available through the **kohonen** package (Wehrens & Buydens, 2007). The remaining algorithms are provided by either the **cluster** package (Maechler et al., 2015) or the **stats** package in the base distribution of R (R Core Team, 2014).

These ten algorithms represent a wide range of clustering methods but are by no means an exhaustive list. In the *optCluster( )* function, the ten available clustering algorithms are: "hierarchical", "agnes", "diana", "kmeans", "pam", "clara", "fanny", "model", "som", and "sota". A description for each clustering algorithm is provided in this chapter.

Agglomerative Hierarchical Clustering

Hierarchical methods create a cluster hierarchy and are usually illustrated by dendrograms. The desired number of clusters does not need to be set in advance since a dendrogram can be sliced at a particular height in order to generate a specified number of clusters. The agglomerative hierarchical algorithms, such as the "hierarchical" and "agnes" options in the *optCluster( )* function, begin with each observation in a separate individual cluster and merge the two closest clusters together to form a larger one. This process of merging the two closest clusters together continues until all of the observations are combined into a single cluster.

There are various options to determine the distance between clusters in agglomerative hierarchical clustering algorithms. The four options considered in the *optCluster( )* function are: average method, complete linkage method, single linkage method, and Ward's method. The average method uses the average pairwise distance between two clusters to determine how close the clusters are to each other. This algorithm is known as the unweighted pair group method with arithmetic mean or UPGMA (Sneath & Sokal, 1973). The complete linkage method evaluates the observations that are the farthest away from each other, for each pair of clusters, and merges the clusters that have the smallest maximum pairwise distance (Anderberg, 1973). The single linkage method evaluates the observations that are nearest to each other, for each pair of clusters, and combines the clusters that have the smallest minimum pairwise distance (Anderberg, 1973). Ward's method (Ward, 1963) evaluates the within-cluster variance of merged clusters and combines the two clusters that produce the smallest increase in the total within-cluster variance.

Diana

Diana is a hierarchical clustering algorithm that is divisive rather than agglomerative thus all the observations start in one large cluster (Kaufman & Rousseeuw, 1990). In each step, the largest cluster is first identified. Here, the largest cluster is the one with the largest diameter as determined by the maximum distance between any two points inside it. Within this largest cluster, the observation that has the greatest average distance from all of the other members inside the cluster (as computed by a distance function) is chosen to form a new group. Observations in the original cluster are then moved into this new cluster if they are deemed "closer" to this group than to the original one, as determined by average distance. The dividing process stops once all of the members "closer" to the new group have been moved, and as a result, two smaller clusters are created. The process of selecting the largest cluster and dividing it repeats until each observation is in its own cluster.

K-means

K-means is an iterative clustering algorithm requiring a fixed number of clusters before it begins (Hartigan & Wong, 1979). An initial set of cluster centers, or centroids, is established and each observation is grouped to the cluster with the closest center. Once all observations have been allocated to a cluster, the cluster centers are recalculated and the process of assigning observations to a cluster repeats. This process continues until there are no new assignments, achieving a minimization of the total within-class sum of squares.

PAM

Partitioning around medoids (PAM) is a clustering algorithm that works in a very similar way to the K-means clustering algorithm (Kaufman & Rousseeuw, 1990). A set cluster size is determined and an initial set of cluster centers is established before this clustering technique begins. Unlike K-means, these centers are medoids (not centroids) and are always observations from the dataset. Once each observation is assigned to the closest medoid cluster, new medoids are chosen in order to minimize a sum of dissimilarities and reassignment of observations to the nearest medoid occurs. This process repeats until there are no new assignments.

Clara

Clara (Kaufman & Rousseeuw, 1990) is an extension of PAM and is a time-efficient algorithm for clustering large datasets. A number of sub-datasets are drawn as representatives of the larger dataset and medoids are established using PAM. The best clustering is chosen from the sub-datasets and the rest of the observations in the entire dataset are assigned to the closest medoid.

Fanny

Fanny is a fuzzy clustering algorithm that uses probability to determine the cluster allocation of the observations (Kaufman & Rousseeuw, 1990). Rather than having each observation defined to a single cluster, this method allows observations to have some degree of association to each cluster. Once the number of desired clusters is determined, probability vectors are created. These vectors denote the partial membership an observation has to each of the clusters. Each observation is sorted into the group with the highest probability, creating hard clusters.

Model-based Algorithm

The model-based algorithm of clustering assumes that the observations come from a finite mixture of normal distributions and each component of that mixture corresponds to a different cluster (Fraley & Raftery, 2002). Under this assumption, a Gaussian mixture model is fit to the data. Estimates for the mixture components and assignment of the observations to those clusters are obtained using maximum likelihood in the expectation maximization (EM) algorithm (Dempster et al., 1977; McLachlan & Krishnan, 1997).

SOM

Self-organizing maps (SOM) is a clustering algorithm that is based on biological neural networks (Kohonen, 2001). Initially, a two-dimensional grid of units is established, each with its own randomly generated weight vector. These weight vectors adjust according to predetermined rules during an iterative unsupervised learning process. During each step of this process, a random observation is selected from the data and the distance between that observation and all of the weight vectors is computed to determine the closest vector. A best matching unit is determined and the weight vectors of all the units are adjusted as defined by the preset rules. This process continues until the weight vectors are no longer changing, at which point the observations are assigned to the closest unit.

SOTA

Self-organizing tree algorithm (SOTA) is a divisive clustering algorithm that has properties similar to both hierarchical and SOM techniques (Dopazo & Carazo, 1997; Herrero et al., 2001). This method maps a complex input space to a more simple output much like SOM, but instead of mapping a two-dimensional grid, SOTA creates a binary tree resembling those created by hierarchical methods. Starting with a root node with two cells (clusters), a training cycle occurs in order to sort the observations into the two clusters. Two new cells are formed from the cluster with the most variable population and the process is repeated. The divisive process can be stopped at any step based on a variety of different criteria or it can proceed until a complete dendrogram is created.

CHAPTER III

VALIDATION MEASURES

A high performing clustering algorithm is one that groups the data in such a way that it retains some internal quality control of the clusters. These groups, or clusters, should be stable and statistically meaningful. Additionally, as in the case of dealing with microarray data and data generated from Next Generation Sequencing technology, the groupings should also be biologically relevant. Validation measures are the quantities that measure the above-mentioned qualities of clustering algorithms and provide scores to help determine the validities of clustering algorithms for specific data. Various measures for cluster validation have been introduced in the literature, but the focus of this chapter will be on describing nine measures that can be utilized in the **optCluster** package. These measures are directly sourced from the **clValid** package, which classifies them into three groups: "biological", "internal", and "stability".

Datta and Datta (2006) proposed two types of biological validation measures to help provide some guidance in choosing a clustering technique for microarray data. These measures can also be used for any other molecular expression data as well. The biological homogeneity index (BHI) and biological stability index (BSI), both evaluate the performance of an algorithm to produce biologically similar clusters.

Internal validation measures provide guidelines on the statistical properties of clusters. Connectivity is a useful internal validation technique, measuring the extent at which neighboring observations are clustered together (Handl et al., 2005). In 1987, Rousseeuw introduced the silhouette width validation measure, which mathematically combines how close the different clusters are to each other (inter-cluster separation) with how large or small the intra-cluster variances are (compactness). The third internal validation measure included in the **clValid** package**,** the Dunn index (Dunn, 1974), also evaluates the compactness and separation of the clusters.

Several other validation measures, called stability measures, have been proposed to address validation of clustering techniques for microarray data. Unlike the internal validation techniques, these measures evaluate whether the cluster assignments remain stable even if the data is reduced somewhat. Yeung et al. (2001) proposed the figure of merit technique (FOM), and in 2003, Datta and Datta presented the three other stability measurements available in the **clValid** package: average proportion of non-overlap, average distance, and average distance between means.

Biological Homogeneity Index (BHI)

The biological homogeneity index (Datta & Datta, 2006) has the range [0,1] and is a biological validation measure evaluating how biologically similar defined clusters are. A microarray dataset will have M number of rows (genes) and T number of columns (e.g. time points). Here we define $\mathcal{A} = \{A_1, \ldots, A_F\}$ with F functional classes, which are not necessarily disjoint, such that $A(i)$ is the functional class that contains gene i. In this definition, there is the possibility that i is contained in more than one functional class. $A(\ell)$ is defined in a similar manner as the functional class that contains gene ℓ. An indicator function is assigned such that $I(A(i) = A(\ell))$ takes the value of 1 when $A(i)$ and $A(\ell)$ match and 0 otherwise. In the case of multiple memberships to functional classes, any single match is adequate. Given the biological class set of $\mathcal{A}$ and the statistical clustering partition of K number clusters obtained from a clustering algorithm $P' = \{C_1, \ldots, C_K\}$, BHI can be defined as

$$BHI(P', \mathcal{A}) = \frac{1}{K}\sum_{k=1}^{K} \frac{1}{\alpha_k(\alpha_k - 1)} \sum_{i \neq \ell \in C_k} I\big(A(i) = A(\ell)\big),$$

where $\alpha_k = n(C_k \cap A)$ is the number of annotated genes within the cluster C_k. Genes belonging to the same functional classes should ideally be placed in the same statistical cluster, so more biologically homogeneous clusters correspond to larger BHI values.

Biological Stability Index (BSI)

The second biological validation measurement is the biological stability index (Datta & Datta, 2006), which measures the reliability an algorithm has of clustering similar biologically functioning genes together. After removing the observations from column *r* in the data, the cluster assignments of genes with similar functionality from the reduced dataset are compared to the cluster assignments from the full dataset. Using previously defined terms from BHI, the definition for BSI is

$$BSI(P', \mathcal{A}) = \frac{1}{F}\sum_{f=1}^{F} \frac{1}{n(A_f)(n(A_f)-1)T} \sum_{r=1}^{T} \sum_{i \neq \ell \in A_f} \frac{n(C^{i,0} \cap C^{\ell,r})}{n(C^{i,0})},$$

where $C^{i,0}$ is the original cluster, based on the full set of data, containing observation *i*, and $C^{\ell,r}$ is the cluster containing observation ℓ when column *r* is removed. Just like BHI, the value for BSI has the range of [0,1] and stable clusters of similarly functioning genes have larger values.

Connectivity

For this internal validation measure let $n^*_{i(j)}$ be the jth nearest neighbor of observation *i* such that $y_{i,n^*_{i(j)}}$ is 0 if *i* and $n^*_{i(j)}$ are in the same cluster, otherwise it is $1/j$.

The connectivity (Handl et al., 2005) for *M* observations into *K* clusters for a specific clustering partition $P' = \{C_1, \ldots, C_K\}$ is defined as

$$Conn(P') = \sum_{i=1}^{M} \sum_{j=1}^{G} y_{i,n^*_{i(j)}} ,$$

where the parameter *G* determines the number of neighbors contributing to the connectivity measure. The connectivity, with a value ranging between zero and ∞, should be minimized.

Silhouette Width

The average of all of the observation's silhouette values defines a second type of internal validation measure called silhouette width (Rousseeuw, 1987). For an observation *i*, the silhouette value has a range of [-1,1] and is a measurement of the degree of confidence in that specific observation's cluster assignment. This value is defined as

$$Sil(i) = \frac{c_i^* - c_i}{\max(c_i^*, c_i)} ,$$

where c_i is the average distance between observation *i* and the remaining observations in the same cluster and c_i^* is the average distance between the observations in the "nearest neighboring cluster" and observation *i*. Defining C^i as the cluster that contains the observation *i,* and C^j as the "nearest neighboring cluster" to *i,* the equations for c_i and c_i^* are as follows

$$c_i = \frac{1}{(n(C^i))} \sum_{\ell \in C^i} dist(i, \ell), \quad c_i^* = \min_{C^j \in P' \setminus C^i} \sum_{\ell \in C^j} \frac{dist(i, \ell)}{n(C^j)} ,$$

where the cardinality of a cluster *C* is *n(C)* and the distance (Euclidean, Manhattan, etc.) between observations *i* and ℓ is represented by $dist(i, \ell)$. Silhouette values near 1 mean that the observation is clustered well, while values near -1 mean the observation is poorly clustered. Therefore, silhouette width should be maximized.

Dunn Index

The Dunn index (Dunn, 1974) is an internal validation measure indicating a ratio of the minimum distance between observations in different clusters and the maximum cluster diameter. With C^{ℓ} defined as cluster that contains the observation ℓ, this ratio is calculated by the equation

$$Dunn(P') = \frac{\min_{C^i, C^{\ell} \in P', C^i \neq C^{\ell}} \left(\min_{i \in C^i, \ell \in C^{\ell}} dist(i, \ell) \right)}{\max_{C_d \in P'} diam(C_d)},$$

where $diam(C_d)$ is the maximum distance between any two points inside cluster C_d. With a value between zero and ∞, the Dunn index should be maximized.

Figure of Merit (FOM)

The figure of merit (Yeung et al., 2001) is a stability validation measurement based on the remaining samples when a single column is removed. FOM can be defined for column *r*, which is removed from *T* total columns, as

$$FOM(r, P') = \sqrt{\frac{1}{T} \sum_{k=1}^{K} \sum_{i \in C_k(r)} dist\left(y_{i,r}, \bar{y}_{C_k(r)}\right)}\ ,$$

where $y_{i,r}$ is the value of the *i*th observation in the *r*th column and $\bar{y}_{C_k(r)}$ is the center (average) of the cluster $C_k(r)$. There is a tendency for FOM to decrease as the number of clusters increases, so an adjustment of $\sqrt{\frac{T}{T-K}}$ is multiplied to the measurement. The final value is determined by calculating the mean of all of the FOM values from the removed columns. The FOM values can range between zero and ∞ and a better performing clustering algorithm is indicated by smaller values.

Average Proportion of Non-Overlap (APN)

The average proportion of non-overlap measure (Datta & Datta, 2003) determines the average proportion of observations placed in different clusters, as compared to the full data clustering, when a single column of data is removed. Defining $C^{i,0}$ as the original cluster, based on the full set of data, containing observation *i*, and $C^{i,r}$ as the new cluster containing observation *i* when column *r* is removed, the APN measure can be computed as

$$APN(P') = \frac{1}{TM}\sum_{i=1}^{M}\sum_{r=1}^{T}\left(1 - \frac{n(C^{i,r} \cap C^{i,0})}{n(C^{i,0})}\right).$$

The APN measure can range from 0 to 1, with highly stable results having values close to zero.

Average Distance (AD)

The average distance (Datta & Datta, 2003) stability validation measure calculates the average distance between the observations in clusters formed with the full dataset and the observations in clusters formed with data with a single column removed. AD can be defined as

$$AD(P') = \frac{1}{TM}\sum_{i=1}^{M}\sum_{r=1}^{T}\frac{1}{n(C^{i,0})n(C^{i,r})}\left[\sum_{i \in C^{i,0}, \ell \in C^{i,r}} dist(i,\ell)\right].$$

The AD measure can range between zero and ∞ and should be minimized.

Average Distance Between Means (ADM)

The average distance between means (Datta & Datta, 2003) stability validation measure computes the average distance between the centers of clusters for observations put into the same cluster based on the full data and the data with a single column removed. Let $\bar{y}_{C^{i,0}}$ be the mean of the observations in the cluster based on the full dataset containing observation *i* and let $\bar{y}_{C^{i,r}}$ be the mean of the observations in the cluster containing observation *i* with column *r* removed. ADM is defined as

$$ADM(P') = \frac{1}{TM}\sum_{i=1}^{M}\sum_{r=1}^{T} dist(\bar{y}_{C^{i,0}}, \bar{y}_{C^{i,r}}).$$

Much like the AD measure, ADM values can range between zero and ∞, with smaller values representing better performing algorithms.

CHAPTER IV

RANK AGGREGATION

A ranked list of clustering algorithms can be generated based on the performance of those algorithms for a specific validation measure. If multiple validation measures are used in cluster analysis, multiple lists are created. The problem of rank aggregation is faced when trying to combine these multiple ranked lists together in some way to form a single list that best represents the original rankings. Many rank aggregation techniques have been developed and used in various applications ranging from building meta-search engines on the Web (Dwork et al., 2001) to combining microarray experiment results (DeConde et al., 2006). For rank aggregation of cluster validation measures, the stochastic optimization method of the cross-entropy Monte Carlo algorithm (Pihur et al., 2007) was proposed. This algorithm and the Genetic algorithm (Goldberg, 1989) are included in the **RankAggreg** package and are available as rank aggregation methods in the **optCluster** package. These two aggregation techniques are explained in more detail in this chapter along with the two distance functions that can be utilized with these techniques, the weighted Spearman's footrule distance and the weighted Kendall's tau distance (Pihur et al., 2007, 2009).

Objective Functions

In a mathematical context, ranked aggregation can be explained in a fairly straightforward manner. Suppose there are m number of ranked lists $(V_1, \dots , V_m)$ of length k, such that each V_i is associated with a weight μ_i. We obtain a definition of an objective function for a proposed list β as

$$\Phi(\beta) = \sum_{i=1}^{m} \mu_i dist(\beta, V_i) \, ,$$

where $dist(\beta, V_i)$ represents a distance function. The goal is to find a list β^* that will minimize the total distance between all of the V_i's and β^*. There are many options available to measure the distances, but careful selection of an option is warranted since final results depend on the choice of distance measure. Two different methods used for distance measures in rank aggregation are included in **RankAggreg** package: weighted Spearman's footrule distance and weighted Kendall's tau distance. Pihur et al. (2007, 2009) introduced these weighted methods as modifications to two popular methods, Spearman's footrule distance and Kendall's tau distance (Fagin et al., 2003), in order to stabilize the aggregation algorithms while using a discrete ranking system.

Suppose for an ordered list V_i there are $S_i(1), \ldots, S_i(k)$ scores associated with it. These scores are normalized in order to spread the corresponding values for each list over the interval [0,1], avoiding a strong influence from disproportionally large or small values. In this notation the scores are ordered with $S_i(1)$ being the best and $S_i(k)$ being the worst. Depending on the context, $S_i(1)$ may either be the maximum or minimum score. Let $R^{V_i}(j)$ denote the rank of j in list V_i, again with 1 being the best rank all the way to a rank k. If the rank of j does not fall within the top k ranks, the value of $R^{V_i}(j)$ will equal $k + 1$. With these definitions, weighted Spearman's footrule distance is defined as

$$WS(\beta, V_i) = \sum_{j \in V_i \cup \beta} \left| S\left(R^{\beta}(j)\right) - S\left(R^{V_i}(j)\right) \right| \times \left| R^{\beta}(j) - R^{V_i}(j) \right|.$$

The weight of $\left|S\left(R^{\beta}(j)\right)-S\left(R^{V_i}(j)\right)\right|$ adjusts this distance by the score differences between the ranks of j in lists β and V_i .

The weighted Kendall's tau distance compares pairs of elements (j and ℓ) from the union of the two lists. This algorithm is defined as

$$WK(\beta, V_i) = \sum_{j,\ell \in V_i \cup \beta} \left|S\left(R^{V_i}(j)\right)-S\left(R^{V_i}(\ell)\right)\right| \times K_{j,\ell}^{p}.$$

Here $K_{j,\ell}^{p}$ can take three possible values 0, 1, or p. If the rank of j is greater than the rank of ℓ (or vice versa) for both lists V_i and β, $K_{j,\ell}^{p}$ is 0. If j is ranked higher than ℓ for one list but ℓ is ranked higher than j on the other list, $K_{j,\ell}^{p}$ is 1. When neither j nor ℓ appear in either list, $K_{j,\ell}^{p}$ takes the value of p, which can be specified within the range [0,1]. The adjustment for the weighted Kendall's tau distance $\left|S\left(R^{V_i}(j)\right)-S\left(R^{V_i}(\ell)\right)\right|$ is the difference in scores for j and ℓ within list V_i.

Cross-Entropy Monte Carlo Algorithm

Originally developed by Rubinstein (1997) to compute probabilities of rare events, the cross-entropy Monte Carlo algorithm was later proposed by Pihur et al. (2007) as a method for weighted rank aggregation of lists composed of cluster validation measures. Here, a ranked list can be represented in terms of an *n x k* matrix where all entries are either 0 or 1. The dimensions of this matrix are defined as *n*, the total number of unique elements (clustering algorithms) in all the ordered lists being combined, and *k*, typically the length of the combined ordered lists. It is important to note that the value of *k* can also be made smaller if one desires. The constraints of this matrix are that the columns sum up to 1 and the rows sum up to at most 1. Thus, each column variable follows a multinomial distribution. Reading from left to right, the position of the 1's in each column of this matrix determines the ordering in the ranked list (Lin & Ding, 2009). This approach is available in the package **RankAggreg** and can be briefly summarized in the following four steps:

1. Initialization: An initial matrix of parameters is established such that each of the *n* elements (clustering algorithms) has an equal chance of being selected in each of the *k* positions of the ranked list.
2. Sampling: A random sample is selected from the most recently generated matrix of parameters. The corresponding optimal lists are determined and objective functions values are calculated.
3. Updating: The parameter matrix is updated based on the current sample and the objective function values so that the next sample group will have smaller objective function values.
4. Convergence: The sequence of sampling and updating will repeat until the optimal list remains the same for a selected number of iterations.

Genetic Algorithm

Genetic algorithms (Goldberg, 1989) were developed as a natural selection type solution to problems involving many possible solutions. The idea is that the "fittest" solution survives after many generations of the algorithm, just like the concept of natural selection in evolution. The Genetic algorithm in the **RankAggreg** package follows the five steps below:

1. Initialization: An initial population of solutions is created as randomly generated ordered lists, each of the same length, *k*. The number of lists generated is based

on a predetermined population size, with larger sizes having a better chance of containing the “best” solution at some point.

2. <u>Selection:</u> Each list is evaluated for fitness by the selected objective function. Using the objective function scores as weights, a weighted random sampling technique is employed to select the lists that will create a new population.
3. <u>Crossover:</u> From a specified crossover probability, the selected lists in the new population perform a one-point crossover. This means a list will swap the elements that are ranked lower than a given point (as determined by the crossover probability) with the elements from another list that are also ranked lowered than that same point.
4. <u>Mutation:</u> To create lists in the new population that are drastically different from the current ones, one or more elements from any of the lists are randomly changed or rather, mutated. A mutation probability is pre-determined before the algorithm begins, which affects the frequency at which the mutations occur.
5. <u>Convergence:</u> The newly created population replaces the previous population and the sequence of the selection, crossover, and mutation steps will repeat until a list remains the optimal one for a selected number of successive iterations.

CHAPTER V

OPTCLUSTER PACKAGE

When a minimal amount of information is know about a particular dataset, which is much the case in high dimensional data, cluster analysis becomes a vital technique for examining the data. A variety of R packages have been developed as tools to help researchers with cluster analysis and the issues associated with it. There are some packages that offer clustering algorithm functions (e.g. **cluster***,* **kohonen**, and **mclust**), while some packages provide functions to calculate validation measures (e.g. **cclust**, **clusterSim***,* **clv**, and **fpc**). Performing a thorough process of determining the optimal number of clusters and finding the "best" clustering algorithm for a given dataset often requires the use of more than one R package.

Writing and running code for multiple functions in multiple packages may be a minor inconvenience, but it can become a tedious task prone to error especially if using the same functions and code over and over for a variety of different sets of data. An efficient way to streamline the cluster analysis process is to unite multiple functions from different packages together into one overlapping function in a single package. This chapter introduces the package **optCluster**, which joins together functions from the packages **clValid** and **RankAggreg** in order to determine the optimal clustering algorithm and optimal number of clusters for a given dataset.

The core function available in the **optCluster** package is called *optCluster( ).* As we saw from Figure 1 in Chapter I, this function completely eliminates the need for the user to extract the necessary information provided by the *clValid( )* function and input it into the *RankAggreg( )* function for additional analysis. Essentially, the user only needs to enter a dataset, along with a few additional arguments (e.g. cluster size range, clustering algorithms, validation measure types, and rank aggregation method), and the optimal clustering algorithm and optimal number of clusters will be determined. Since the use of functions from the **clValid** and **RankAggreg** packages are required, both packages must be installed in the user's R library in order to run the *optCluster( )* function. All of the R code used in this function, as well as all of the code for the entire **optCluster** package, can be found in Appendix A.

A total of ten clustering algorithms, coming from various R packages, are available for cluster analysis through the *optCluster( )* function argument *clMethods*. The **stats** package contains the algorithms of "hierarchical" and "kmeans". The algorithms of "agnes", "clara", "diana", "fanny", and "pam" are available in the **cluster** package. The algorithm "model" is available through the package **mclust**, "som" is provided through the package **kohonen**, and "sota" is in the **clValid** package. Details for all of these clustering algorithms were described in Chapter II. It should be noted that *clMethods* is the same argument that the *clValid( )* function uses for selecting clustering algorithms. In the **clValid** package, arguments and output use the term "methods" to denote clustering algorithms. To remain consistent with this terminology, the *optCluster( )* function also refers to clustering algorithms as "methods" in arguments and output.

All ten algorithms are included as default in the *optCluster( )* function so that every available option is being considered in the evaluation. The user, however, may select any number of algorithms as desired. The package **cluster** is required to run the *optCluster( )* function because it is automatically loaded when the **clValid** package is loaded. The packages **mclust** and **kohonen** are suggested packages and will only be needed if the user chooses to evaluate the "som" and "model" algorithms respectively.

For each cluster size entered, validation measures are calculated for all of the selected clustering algorithms. The function *clValid( )* is called by *optCluster( )* to perform these validation measure calculations. Using the argument *validation*, the user has the option of choosing nine validation measures, described in Chapter III, based on three categories: "biological" (BHI and BSI), "internal" (connectivity, Dunn index, and silhouette width), and "stability" (FOM, APN, AD, and ADM). While "stability" is the default option as defined by *clValid( )*, the function *optCluster( )* uses the default of both "internal" and "stability" to include more validation measures in determining the optimal clustering algorithm.

While it would be most efficient to also include the biological validation measures (Datta & Datta, 2006) in the default analysis, there are additional packages that would need to be required. If "biological" is chosen as a type of validation measure, the packages **Biobase** (Gentleman et al., 2004)*,* **GO.db** (Carlson, 2015a), and **annotate** (Gentleman, 2014) will need to be installed and loaded from Bioconductor (Gentleman et al., 2004). The user would also need to provide an appropriate *annotation* argument of either the name of a Bioconductor package that maps genes to Gene Ontology (GO) categories or a list of functional classes and the observations that belong to each one. Since the *annotation* argument depends on the data and may not even be available for a particular dataset, the "biological" type of validation measures is not included in the default for the *validation* argument in the *optCluster( )* function.

Once validation measures are calculated for all combinations of number of clusters and clustering algorithms, lists are compiled for each validation measure ranking the algorithms according to their performance. These lists are created within the *optCluster( )* function by calling the *getRanksWeights( )* function from the package **clValid**. In many cases, the top-performing algorithm for one measure is different from the top-performing algorithm in another. This makes determining the optimal clustering algorithm with a visual inspection of the lists nearly impossible. To overcome this challenge, the *optCluster( )* function calls the *RankAggreg( )* function (from the package **RankAggreg**) to combine all of the lists into one overall "top" list.

The methods of a cross-entropy Monte Carlo algorithm and a Genetic algorithm are available as options for rank aggregation with the *RankAggreg( )* function, using either the weighted Spearman's footrule distance or the weighted Kendall's tau distance. Chapter IV described the methods for both distance measures and both rank aggregation algorithms in detail. The user may select either aggregation method along with either distance measure in the *optCluster( )* function using the *rankMethod* and *distance* arguments respectively. Since the Genetic algorithm may require many iterations to converge and the weighted Kendall's tau distance calculations are time intensive, the default weighted rank aggregation arguments are the cross-entropy Monte Carlo algorithm and the weighted Spearman's footrule distance.

The function *optCluster( )* will output the "best" clustering algorithm along with the corresponding optimal number of clusters after an overall "top" list has been decided. An object of S4 class "optCluster" is returned by this function, which has several options for viewing results. The *top.method( )* statement will return only the name of the optimal clustering algorithm and the optimal number of clusters. The *print( )* statement returns the same output as *top.method( )* along with the optimal ranked list and the minimum objective score calculated by the selected rank aggregation algorithm. The *summary( )* statement displays a table of all of the calculated validation measures, the optimal clustering algorithm and number of clusters for each individual validation measure, and all of the previous information provided in the *print( )* statement.

The “optCluster” class object also allows access to the internal objects that were created by *clValid( )* and *RankAggreg( )* functions while the *optCluster( )* function was running. With the *cl.valid( )* method, the user can obtain an S4 object of class “clValid” to extract and even plot the clustering results and validation measure scores for each measure. The *rank.aggreg( )* method allows the user to obtain an S3 object of class “raggr” to get the overall top list of clustering algorithms. A visual representation of the rank aggregation results can also be viewed using the *plot( )* function. These two methods for the “optCluster” class acquire all of the information provided by the functions *clValid( )* and *RankAggreg( )*, so there is no need to run either of the functions independently of the *optCluster( )* function.

Almost all of the arguments from the *clValid( )* and *RankAggreg( )* functions can be entered by the user directly into the *optCluster( )* function and passed to the appropriate function. The arguments of *method* and *verbose* are used by both of these functions, and therefore have been divided into separate input arguments for the *optCluster( )* function. The *method* argument is divided into *hierMethod* to specify the agglomerative method used in the hierarchical clustering algorithms (either “ward”, “single”, “complete”, or “average”) and *rankMethod* to specify the rank aggregation method as either cross-entropy Monte Carlo ("CE") or Genetic algorithm ("GA"). The default for *hierMethod* is "average", the UPGMA algorithm, and the default for *rankMethod* is the "CE" method. To account for the commonality of the *verbose* argument, *clVerbose* is used for displaying output on the cluster validation progress in the *clValid( )* function, while *rankVerbose* is used for displaying output at each iteration of the *RankAggreg( )* function. To avoid superfluous output and increase performance speed, both *clVerbose* and *rankVerbose* are set to FALSE as default.

The **optCluster** package also offers a second function called *repRankAggreg( )* that can repeat weighted rank aggregation once the *optCluster( )* has produced an initial result. Since cluster validation scores are often the same as long as the same clustering algorithms and validation measures are desired, it is redundant to repeat these calculations again with the *optCluster( )* function. For small datasets this may be a minor inconvenience, but calculating validation scores for high dimensional data can be very time consuming. Therefore, it would be beneficial to only have to run cluster validation once.

A user can simply input an "optCluster" class object into the *repRankAggreg( )* function and weighted rank aggregation using the same rank aggregation method and distance measure chosen in the original *optCluster( )* function will be performed. A new rank aggregation method can be selected using the *rankMethod* argument and a new distance measure can be chosen using the *distance* argument. Just like the *optCluster( )* function, the *rankVerbose* argument will determine whether or not output is displayed for each iteration of rank aggregation. All other arguments for the *RankAggreg( )* can also be passed through the *repRankAggreg( )* function.

CHAPTER VI

EXAMPLES

In this chapter, we provide the detailed steps that are needed to run cluster analysis using the proposed **optCluster** package on two different types of biological data. These two examples were selected specifically for this paper because they represent common methods of gene expression profiling in biomedical research. The first dataset is microarray gene expression data from Bhattacherjee et al. (2007) and the second dataset is RNA-Seq data from Di et al. (2011), generated from Next Generation Sequencing technology. Both are arranged in the typical clustering format for gene expression where the genes are rows and the samples are columns. Depending on an individual dataset it may be advantageous to cluster either the genes or the samples, but for these two examples we will focus on clustering only the genes.

To begin cluster analysis with the **optCluster** package, it must first be loaded into R with the *library( )* function, *library(optCluster).* Upon loading the **optCluster** package, three other packages will be automatically loaded: **clValid***,* **cluster**, and **RankAggreg**. Once the function *optCluster( )* begins running, it will load the packages **kohonen** and **mclust** if the clustering algorithms of "som" and "model" are selected respectively. If "biological" is entered as a validation type, three additional packages needed for biological validation will also be loaded once the function *optCluster( )* begins running: **Biobase***,* **GO.db***,* and **annotate.** Figure 2 displays a flowchart of this package loading process.

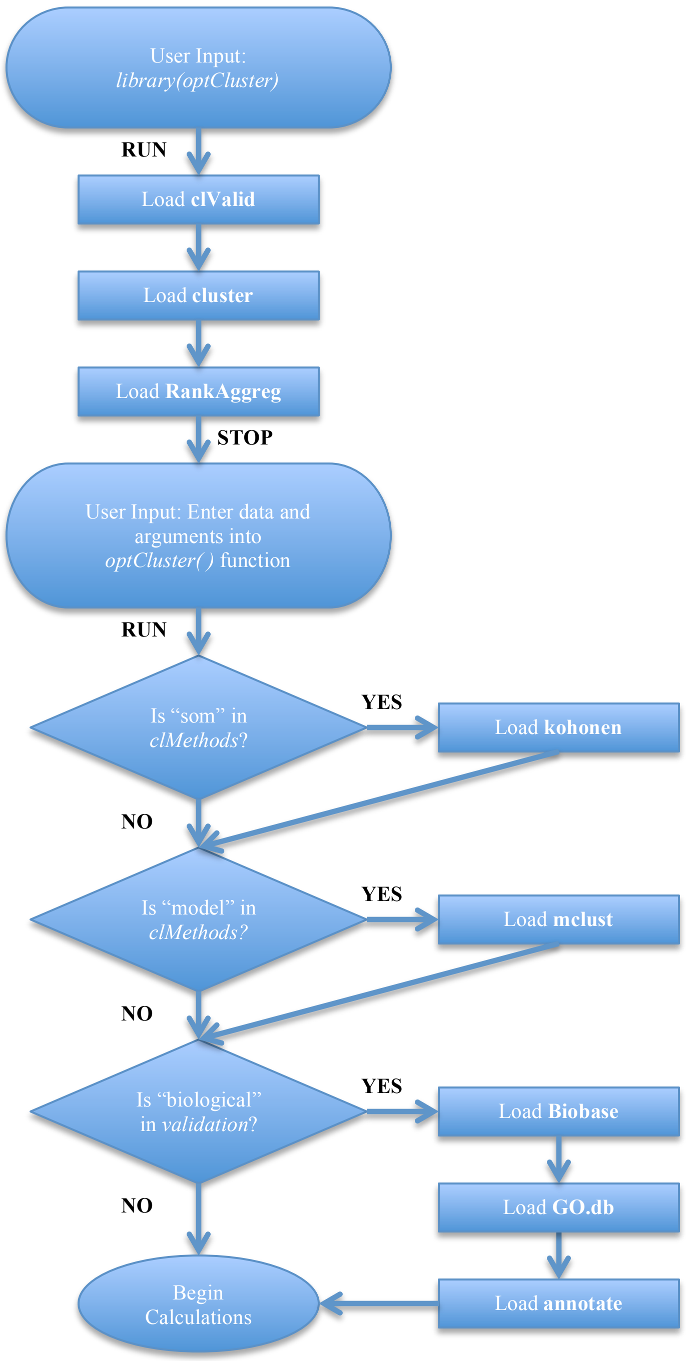

Figure 2: A flowchart describing when additional R packages are loaded by the **optCluster** package. The user input of *library(optCluster)* will load the **clValid**, **cluster**, and **RankAggreg** packages. Other packages will be loaded depending on the user inputs for the *clMethods* and *validation* arguments in the *optCluster( )* function.

Example 1: Microarray Data Analysis

This dataset comes from the Bhattacherjee et al. (2007) microarray experiment evaluating gene expressions of mesenchymal cells in mice. It is available in the package **clValid**, which is automatically loaded with the **optCluster** package. Two lineages of mesenchymal cells important in the development of the orofacial region were compared: neural crest and mesoderm-derived. In the analysis, the expressions of genes and Expressed sequence tags (ESTs) that were different between the two cell lineages by at least 1.5 fold (either in an increase or decrease) were considered to be significantly different. A total of 147 genes and ESTs were determined to be significantly different and are represented as the rows in this data. Three samples were taken from both the neural crest and mesoderm-derived cells, for a total of six columns in the dataset. The *data()* function allows us to load this data from the **clValid** package, and some initial manipulation is needed to gather only the necessary information from the original dataset.

```
> data(mouse)
> ex1 <- mouse[, c("M1", "M2", "M3", "NC1", "NC2", "NC3")]
> rownames(ex1) <- mouse$ID
```

Both Brock et al. (2011) and Pihur et al. (2009) have evaluated this data in a limited context as examples of cluster analysis using weighted rank aggregation. In the Brock et al. analysis, three algorithms ("hierarchical", "kmeans", and "pam") were evaluated over the range of four to six clusters using the "internal" and "stability" types of validation measures available in the package **clValid**. After applying rank aggregation using the cross-entropy Monte Carlo algorithm with weighted Spearman's footrule distance from the **RankAggreg** package, PAM with six clusters was determined the best clustering algorithm over this range. The Pihur et al. rank aggregation analysis only used the first 100 genes of the dataset and only evaluated clustering algorithms for five clusters to obtain the optimal clustering algorithm, rather than using a range of cluster sizes. All ten of the clustering algorithms available in the **clValid** package were evaluated ("agnes", "clara", "diana", "fanny", "hierarchical", "kmeans", "model", "pam", "som", and "sota") using the "internal" and "stability" validation measures. Both rank aggregation methods concluded SOM was the best clustering algorithm for five clusters using weighted Spearman's footrule distance and K-means was the optimal algorithm for five clusters using weighted Kendall's tau distance.

To determine the optimal clustering algorithm and optimal number of clusters using the **optCluster** package, an extensive cluster analysis is performed for this microarray dataset. All ten of the clustering algorithms available in the *optCluster( )* function are evaluated. As default for the agglomerative hierarchical clustering algorithms, "hierarchical" and "agnes", the agglomerative method is set to "average" (Unweighted Pair Group Method with Arithmetic Mean or UPGMA). All three types of validation measures are chosen for a total of nine different validation measures and the package **moe430a.db** (Carlson, 2015b) is set as the biological annotation. The cross-entropy Monte Carlo algorithm ("CE") is selected for rank aggregation methods using the method of weighted Spearman's footrule ("Spearman") for distance measures. The maximum number of aggregation iterations is set to 1500.

While testing the *optCluster( )* function with this dataset, we found that the *RankAggreg( )* function often chooses the optimal number of clusters to be the upper bound of the given range of cluster sizes. Because of this, the appropriate range of cluster sizes to be considered for this example was determined by looking for a situation where the upper bound was not chosen as the optimal number of clusters. We first started with an initial cluster size range of two to four and ran the *optCluster( )* function using the above-mentioned arguments. If the upper bound of the range was chosen as the optimal cluster size, we increased the upper bound by one and ran the *optCluster( )* function again. This process continued until the optimal number of clusters was not equal to the upper bound of the range. For this dataset, the range of cluster sizes being considered is from two to nine. This range is represented in the second argument in the *optCluster( )* function.

```
> optMouse <- optCluster(ex1, 2:9, clMethods = “all”, validation = “all”, seed
+ = 123, annotation ="moe430a.db", maxIter = 1500)
> top.method(optMouse)
[1] "som-8"
```

The time needed to run the *optCluster( )* function for this analysis was 6.10 hours using 16 GB of RAM on the University of Louisville’s Cardinal Research Cluster (CRC). To view the results of the analysis, the *top.method( )* statement is used. From this simple output, we see that SOM with eight clusters is the optimal clustering algorithm for the data. Appendix B contains the results produced by the output from the *print( )* and *summary( )* methods, which are much more detailed compared to the *top.method( )* statement but also very similar to the rest of the analysis provided in this section.

An object of class “clValid” is created using the method *cl.valid( ),* allowing the utilization of all of the methods available for that S4 class. The optimal values for each validation measure, over the cluster size range from two to nine, are obtained with the “clValid” class method *optimalScores( )*. These values represent the optimal rank aggregated score for each validation measure and are listed under the score column in the output table.

```
> valMouse <- cl.valid(optMouse)
> optimalScores(valMouse)
                 Score Method Clusters
APN          0.0424388    som        7
AD           1.3075210  clara        9
ADM          0.1296167    som        7
FOM          0.4328808    som        9
Connectivity 5.3269841  agnes        2
Dunn         0.1467466  agnes        9
Silhouette   0.5132739  agnes        2
BHI          0.3457009  model        8
BSI          0.7949653  agnes        2
```

The results shown above highlight the complexity of selecting a "best" clustering algorithm. The optimal values from the nine validation measures come from six different combinations of clustering algorithms and cluster sizes. Agnes with two clusters is the best algorithm for three measures (connectivity, silhouette width, and BSI), SOM with seven clusters is the optimal algorithm for APN and ADM, and the four remaining algorithms (Clara with nine clusters, SOM with nine clusters, Agnes with nine clusters, and model-based with eight clusters) are only the top choice for one measure.

A graphical representation of the validation measures can be produced using the "clValid" class method *plot( )*. The plots of all the validation measures can be seen in Figure 3. The clustering algorithms, with the exception of the model-based clustering algorithm, follow the same decreasing trend across the number of clusters for the biological stability index, average distance, and figure of merit measures. These algorithms (excluding the model-based algorithm) also have a similar increasing trend across the number of clusters in connectivity. There is much more variation between the validation scores in the biological homogeneity index, Dunn index, and average proportion of non-overlap measures. From these plots, it is difficult to visually determine one overall best clustering algorithm.

An object of class “raggr” can be created, using the *rank.aggreg( )* method for the “optCluster” class object, in order to gather additional results dealing with the weighted rank aggregation of this data. With this object, an overall optimal list can be acquired along with the minimum objective score from the selected rank aggregation method and distance measure.

```
> aggMouse <- rank.aggreg(optMouse)
> aggMouse
The optimal list is:
        som-8 som-7 clara-8 kmeans-9 hierarchical-9 agnes-9 agnes-8 clara-9
        kmeans-8 pam-6 hierarchical-8 pam-9 agnes-7 pam-8 hierarchical-7
        fanny-7 pam-7 kmeans-7 diana-8 sota-9 som-6 som-3 diana-9 agnes-6
        som-9 sota-8 agnes-5 clara-7 hierarchical-6 hierarchical-5 fanny-9
        som-4 clara-6 fanny-6 fanny-5 fanny-8 diana-4 kmeans-3 diana-6 agnes-4
        sota-7 diana-7 diana-5 kmeans-6 agnes-2 som-5 fanny-3 kmeans-2
        kmeans-5 hierarchical-2 clara-5 fanny-4 hierarchical-4 sota-2 diana-3
        diana-2 hierarchical-3 kmeans-4 sota-3 agnes-3 som-2 pam-5 sota-6
        clara-4 fanny-2 sota-5 pam-3 clara-3 clara-2 pam-2 model-2 sota-4
        pam-4 model-3 model-7 model-8 model-9 model-6 model-5 model-4

  Algorithm:   CE
  Distance:    Spearman
  Score:       239.4503
```

Looking at the optimal list, the top three clustering algorithms are SOM with eight clusters, SOM with seven clusters, and Clara with eight clusters. Agnes with two clusters may have been the top performer for three validation measures but it ranks 45th in the final list because of its poor performance in the five other measures. Final rank aggregation correctly identified that fact. The model-based clustering algorithm is an overall poor choice for this dataset as it occupies eight of the ten lowest ranked algorithm spots in the list.

A visual representation of the weighted rank aggregation results can be displayed using the *plot( )* function on the “raggr” class object. There are three plots produced (see Figure 4) providing information on the convergence properties of the rank aggregation method and the final ranking analysis. After 1048 iterations, convergence was achieved with a minimum objective score of 239.45 from the cross-entropy Monte Carlo algorithm with weighted Spearman's footrule distance. The final sampling distribution has a very high concentration centered very close to the minimum objective score.

All R code used for this cluster analysis can be found in Appendix A. The *repRankAggreg( )* function, in the **optCluster** package, was run twenty times using the same set of arguments but different seeds to look at the consistency of the optimal clustering algorithm and cluster size result from rank aggregation. SOM with eight clusters was determined the optimal clustering algorithm all twenty out of twenty times. Using the *repRankAggreg( )* function, the rank aggregation analysis took an average of 6.02 hours using 16 GB of RAM on the CRC. This function is slightly more time efficient than running the *optCluster( )* function over again for this dataset, with a difference of 0.08 hours (about 5 minutes).

The combinations of the cross-entropy Monte Carlo algorithm with weighted Kendall's tau distance and the Genetic algorithm with both types of distance measures were also considered as weighted rank aggregation options for this dataset analysis. However when running the *repRankAggreg( )* function, these methods did not converge within the long queue wall-time of 168 hours on the Cardinal Research Cluster (CRC), using 16 GB of RAM. For this reason, the results from these methods could not be obtained and, therefore, are not included in this paper.

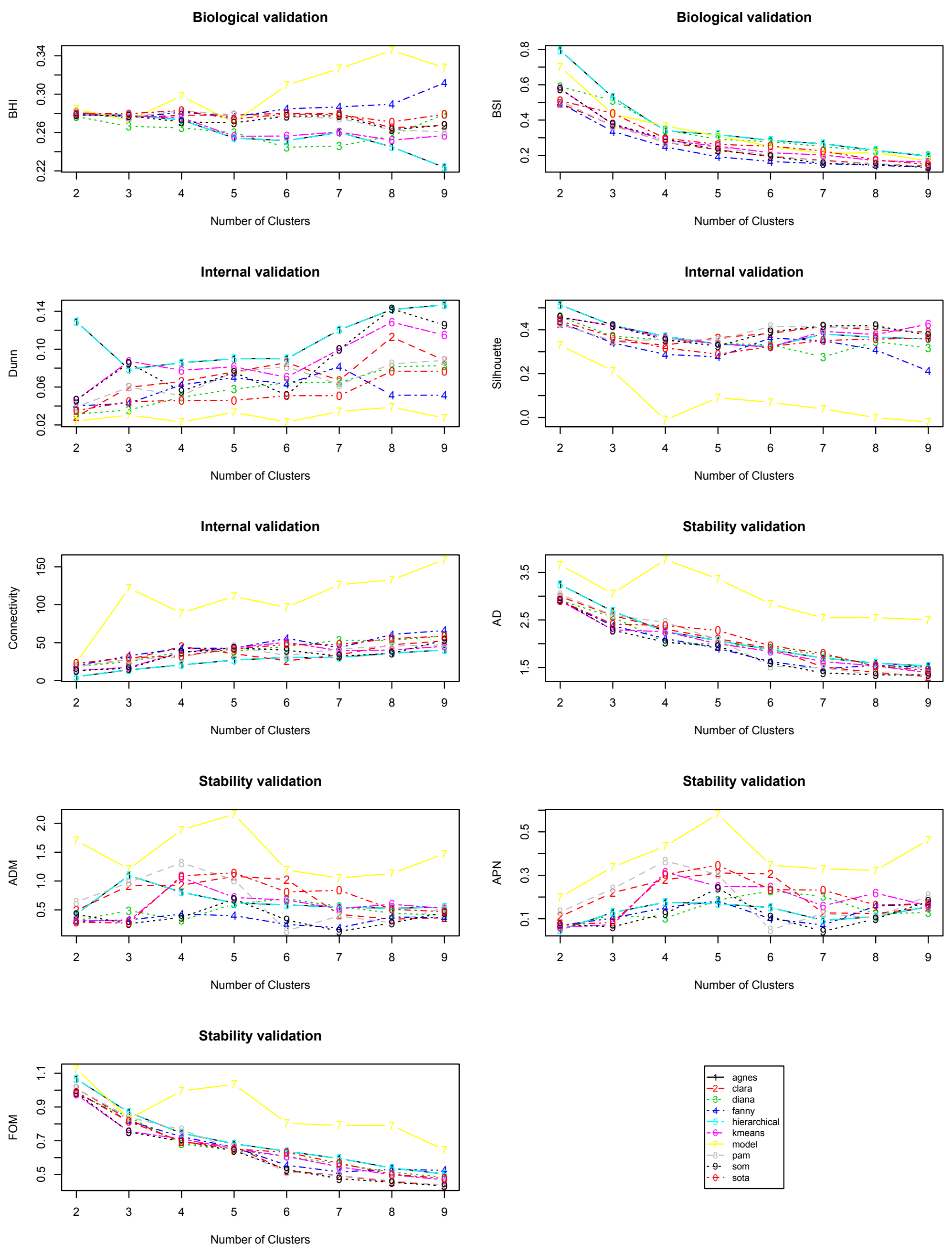

Figure 3: Plots of all nine validation measures for Example 1. The BHI, BSI, Dunn index, and silhouette width measures should be maximized. The AD, ADM, APN, Connectivity, and FOM measures should be minimized.

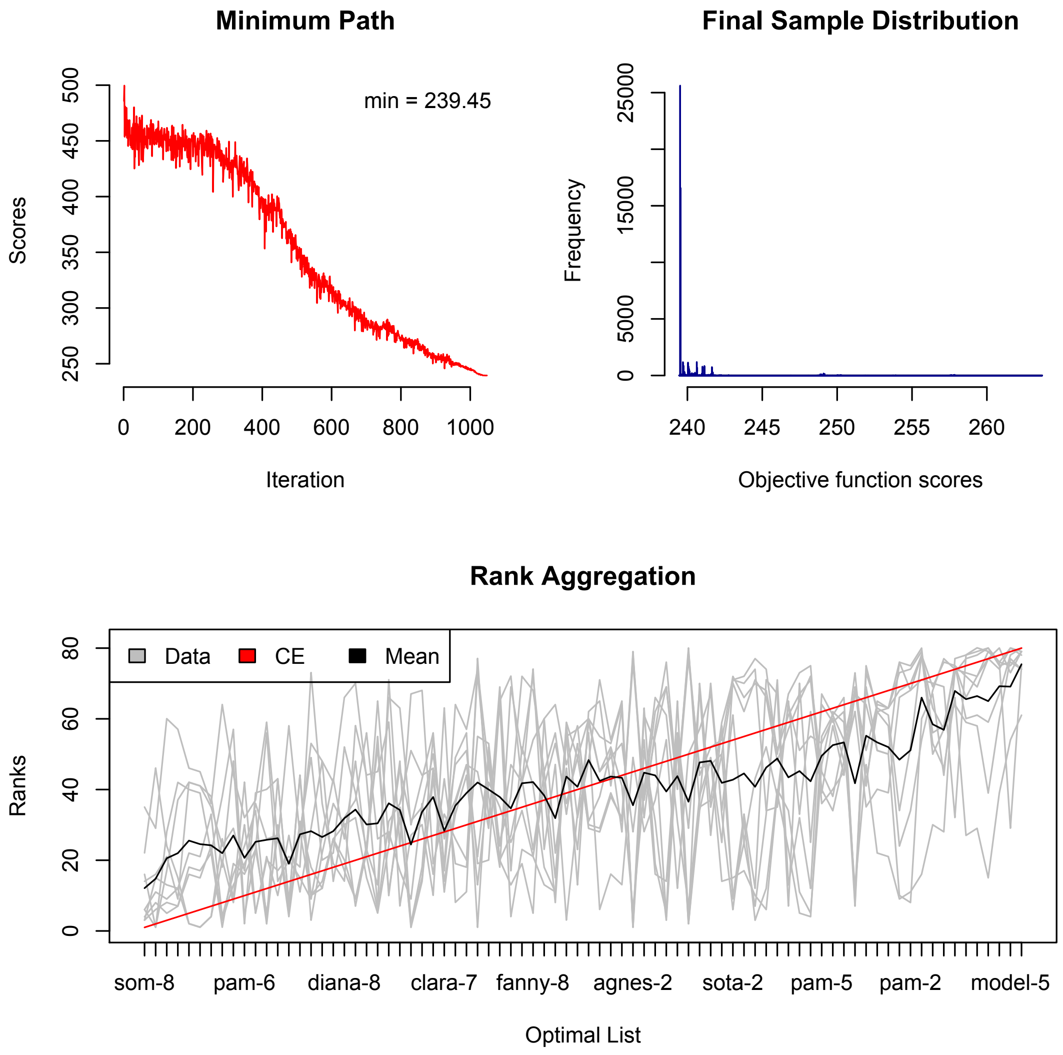

Figure 4: Visual representation of the "CE" rank aggregation results for Example 1 using the "Spearman" distance. The top left plot shows the minimum values of the objective function as the number of iterations increases. The top right plot is a histogram of the objective function scores at the last iteration. The bottom plot shows the individual ranks from the data (in grey), the final solution (in red), and average ranking (in black).

Example 2: RNA-Seq Data Analysis

This second set of data from Di et al. (2011) is an RNA-Sequencing analysis of *Arabidopsis thaliana*, a plant widely used in genetic and molecular biology research. In this study, plants were infected with either a bacteria (*ΔhrcC* mutant of *Pseudomonas syringae* pathovar *tomato* DC3000) or a mock inoculation (10 mM $MgCl_2$) in order to study the defense response of the plants. Three independent samples were used for each infection type in this experiment, for a total of six columns in the dataset. The 26,222 rows of data correspond to different genes, and each individual cell of the matrix contains the counts of the RNA-Seq reads that are mapped to a reference database of known genes. This dataset is available in R through the package **NBPSeq** (Di et al., 2015). Because the obtained dataset consists of plain read counts, Di et al. (2011) normalized the data with respect to library size (column totals) and so we will do the same for our analysis.

```
> library(NBPSeq)
> data(arab)
> ex2 <- t(arab)/colSums(arab)
> ex2 <- t(ex2)
```

To use the **optCluster** package on this dataset, a cluster analysis using eight out of the ten available clustering algorithms is performed. Due to the wall-time of 168 hours on the Cardinal Research Cluster (CRC), two clustering algorithms, Agnes and Diana, were removed from the analysis. Using 32 GB of RAM on the CRC, these two algorithms had the longest times of calculating validation measures. Table 1 contains the average times for all ten of the clustering algorithms to calculate the nine validation measures available in the **clValid** package using only a cluster size of both two, rather than a range of cluster sizes.

Algorithm	Time (Hours)
Clara	0.63
SOTA	1.06
SOM	1.13
PAM	1.13
Kmeans	1.23
Hierarchical	1.24
Fanny	2.00
Model	11.31
Agnes	83.04
Diana	92.97

Table 1: Average calculation times for all nine validation measures with a cluster size of two for Example 2, using 32 GB of RAM.

For only a cluster size of two, Agnes took on average 83.04 hours to compute validation measures, and Diana took an average of 92.97 hours for the same calculations. In comparison, out of the eight algorithms used in this analysis, the model-based algorithm had the longest calculation time for two clusters, with an average time of 11.31 hours. The complete analysis (both cluster validation and rank aggregation) using the eight other clustering algorithms took 85.33 hours using 32 GB of RAM on the CRC.

For the remaining arguments in the *optCluster( )* function, UPGMA is used as the "hierarchical" clustering algorithm, with the agglomerative method set to the default of "average". All nine cluster validation measures are selected and the package **org.At.tair.db** (Carlson, 2015c) is designated as the biological annotation. The rank aggregation method is set as the cross-entropy Monte Carlo algorithm ("CE") with distance measures calculated by weighted Spearman's footrule distance ("Spearman").

Just like Example 1, the cluster size range was determined by looking for a situation where the optimal number of clusters was not equal to the upper bound of the given range of cluster sizes for the arguments mentioned above. The *optCluster( )*

function was initially run with a cluster size range of two to four and the upper bound was not chosen. Therefore, the range of two to four clusters is used in this analysis.

```
> optArabid <- optCluster(ex2[,], 2:4, clMethods = c("clara", “fanny”,
+ "hierarchical", "kmeans", "model", "pam", "som", "sota"), validation = “all”,
+ seed = 123, annotation = "org.At.tair.db", maxitems = nrow(ex2[,]))
> print(optArabid)

The overall optimal method with number of clusters is:
       hierarchical-3

The optimal list is:
        hierarchical-3 kmeans-4 som-4 kmeans-2 sota-4 hierarchical-4
        hierarchical-2 som-3 clara-3 clara-2 sota-3 pam-4 sota-2 kmeans-3 som-2
        pam-3 pam-2 fanny-2 fanny-3 model-2 fanny-4 clara-4 model-3 model-4

  Algorithm:   CE
  Distance:    Spearman
  Score:       55.13205
```

From the *print( )* method, the optimal clustering algorithm is obtained, which is UPGMA (hierarchical) with three clusters. The optimal list of relative performances of all the clustering algorithms considered from the weighted rank aggregation is also displayed using this method. The cross-entropy Monte Carlo algorithm using the weighted Spearman’s footrule distance reported the minimum objective score of 55.13205. Figure 5 shows a visual display of the aggregation results. Convergence to the optimal list was achieved after 186 iterations and most of the mass in the final Monte Carlo sampling distribution is slightly right of the minimum objective score.

The top performing clustering algorithm and optimal validation score for each validation measure can be acquired using the *optimalScores( )* function on a created “clValid” class object. Looking at the optimal scores below, the hierarchical algorithm (UPGMA) seems like an appropriate clustering algorithm for this range of cluster sizes. UPGMA with three clusters is the optimal choice for four validation measures (APN, ADM, Dunn index, and BSI), UPGMA with two clusters is the top choice for two different validation measures (connectivity, silhouette width), and UPGMA with four clusters in the "best" algorithm with BHI. Appendix B has the full list of validation scores obtained from the *summary( )* method for an object of "optCluster" class.

```
> valArabid <- cl.valid(optArabid)
> optimalScores(valArabid)
                    Score       Method Clusters
APN          0.000000e+00 hierarchical        3
AD           9.715409e-05          pam        4
ADM          0.000000e+00 hierarchical        3
FOM          1.533618e-04       kmeans        4
Connectivity 2.928968e+00 hierarchical        2
Dunn         1.108241e+00 hierarchical        3
Silhouette   9.971414e-01 hierarchical        2
BHI          3.340580e-01 hierarchical        4
BSI          9.999677e-01 hierarchical        3
```

The graphs of the validation scores in Figure 6 also seem to support the idea that the hierarchical algorithm is a suitable choice as a clustering algorithm for this cluster size range. The hierarchical algorithm (UPGMA) clearly outperforms the other algorithms across all three numbers of clusters for the Dunn index and the average distance between means measures. This algorithm also appears to be the among the top performing algorithms, if not the top performing algorithm, for all three values of cluster sizes for the biological stability index, silhouette width, connectivity, and average proportion of non-overlap measures. However, looking at the average distance (AD) validation measure, UPGMA is not the best but rather the worst performing algorithm across all three numbers of clusters. Hence, concluding just by visual inspection of the figures may not be suitable while optimizing in terms of a large number of validation measures.

To look at the consistency of the optimal clustering algorithm and cluster size choice, using the cross-entropy Monte Carlo algorithm with the weighted Spearman's footrule distance, the function *repRankAggreg( )*, in the **optCluster** package, was used to repeat rank aggregation for a large number of seeds (twenty). Since this dataset is so big, calculating all of the validation measures for all of the clustering algorithms is very time-consuming. The *repRankAggreg( )* function was used to repeat only the rank aggregation analysis and took, on average, only 13.83 seconds to complete using 16 GB of RAM on the CRC. For all twenty times this dataset was run using the same arguments but different seeds, UPGMA with three clusters was the optimal choice. However, using weighted Kendall's tau as the rank aggregation distance in the *repRankAggreg( )* function does change the results.

```
> KenArabid <- repRankAggreg(optArabid, distance = “Kendall”)
> print(KenArabid)

The overall optimal method with number of clusters is:
        kmeans-4

The optimal list is:
        kmeans-4 hierarchical-3 kmeans-2 hierarchical-4 hierarchical-2 som-4
        som-3 clara-2 kmeans-3 som-2 sota-4 sota-3 fanny-2 pam-4 clara-3
        fanny-4 fanny-3 pam-2 sota-2 model-2 clara-4 pam-3 model-4 model-3

  Algorithm:   CE
  Distance:    Kendall
  Score:       19.56713
```

The *repRankAggreg( )* function took an average of 8.36 hours to complete this analysis using 16 GB of RAM on the CRC. For this changed distance measure, K-means with four clusters is chosen as the optimal algorithm, while hierarchical (UPGMA) with three clusters is ranked 2nd in the optimal list. By connecting the results with the validation measure plots in Figure 6, K-means with four clusters seems to fall within the upper quartile of rankings (top six clustering algorithms) for many of the validation measures. This algorithm's performance also seems to rank either slightly higher or lower than UPGMA with three clusters for several of these measures. Therefore, it is not surprising that these two algorithms were switched in the optimal lists produced by rank aggregation when different distance measures were used for the cross-entropy Monte Carlo algorithm. Figure 7 displays the visual results from the rank aggregation using Kendall's tau distance. Convergence was achieved after 192 iterations with a minimum score of 19.56713, and similar to the weighted Spearman's footrule distance plot, the mass of the final sampling distribution is slightly right of the minimum objective score.

Changing the distance measure for the cross-entropy Monte Carlo algorithm did change both the optimal clustering algorithm and the optimal number of clusters. For this range of cluster sizes, UPGMA with three clusters was chosen with the weighted Spearman's footrule distance while K-means with four clusters was selected using the weighted Kendall's tau distance. From these results, we can see that the rank aggregation using weighted Kendall's tau distance chose the upper bound of the cluster size range as the optimal number of clusters, just as we had mentioned previously. This choice of cluster size makes it difficult to determine whether or not four clusters is an optimal number of clusters for the dataset. Since this chapter's intent is to demonstrate the capabilities of the **optCluster** package, our analysis stops here. However, further investigation into the optimal number of clusters, by extending the range of cluster sizes for this dataset, would be required if one were to choose rank aggregation using the weighted Kendall's tau distance for cluster analysis.

The Genetic algorithm was also used for rank aggregation is this analysis, and produced the same results as the cross-entropy Monte Carlo algorithm for the weighted Spearman's footrule distance (UPGMA with 3 clusters) but the optimal clustering algorithm and cluster size was SOM with four clusters when the weighted Kendall's tau distance was used. For the sake of brevity, visual representations of the aggregation using “Spearman” and “Kendall” distance measures are displayed in Figure 8 and Figure 9 respectively, and the *print( )* method outputs are provided in Appendix B. Table 2 lists the average amount of time required to run the *repRankAggreg( )* function for all methods of weighted rank aggregation for Example 2. It is interesting that although the cross-entropy Monte Carlo algorithm takes, on average, less time when using the weighted Spearman’s footrule distance compared to the Genetic algorithm, when the weighed Kendall’s tau distance is used, the Genetic algorithm, on average, is actually more time-efficient.

Method	**Distance**	**Time (Minutes)**
“CE”	“Spearman”	0.23
“GA”	“Spearman”	0.75
“CE”	“Kendall”	501.60
“GA”	“Kendall”	97.53

Table 2: Average weighted rank aggregation calculation times for Example 2, using 16 GB of RAM.

Appendix A contains the entire code used for the *Arabidopsis thaliana* RNA-Seq data analysis. The package **bigmemory** (Kane et al., 2013) was utilized in this analysis to create a file-backed matrix for this dataset. The function *as.big.matrix( )* stored this high dimensional data on the hard drive rather than keeping it in memory, which avoided maxing out the memory and freezing R while running the *optCluster( )* function.

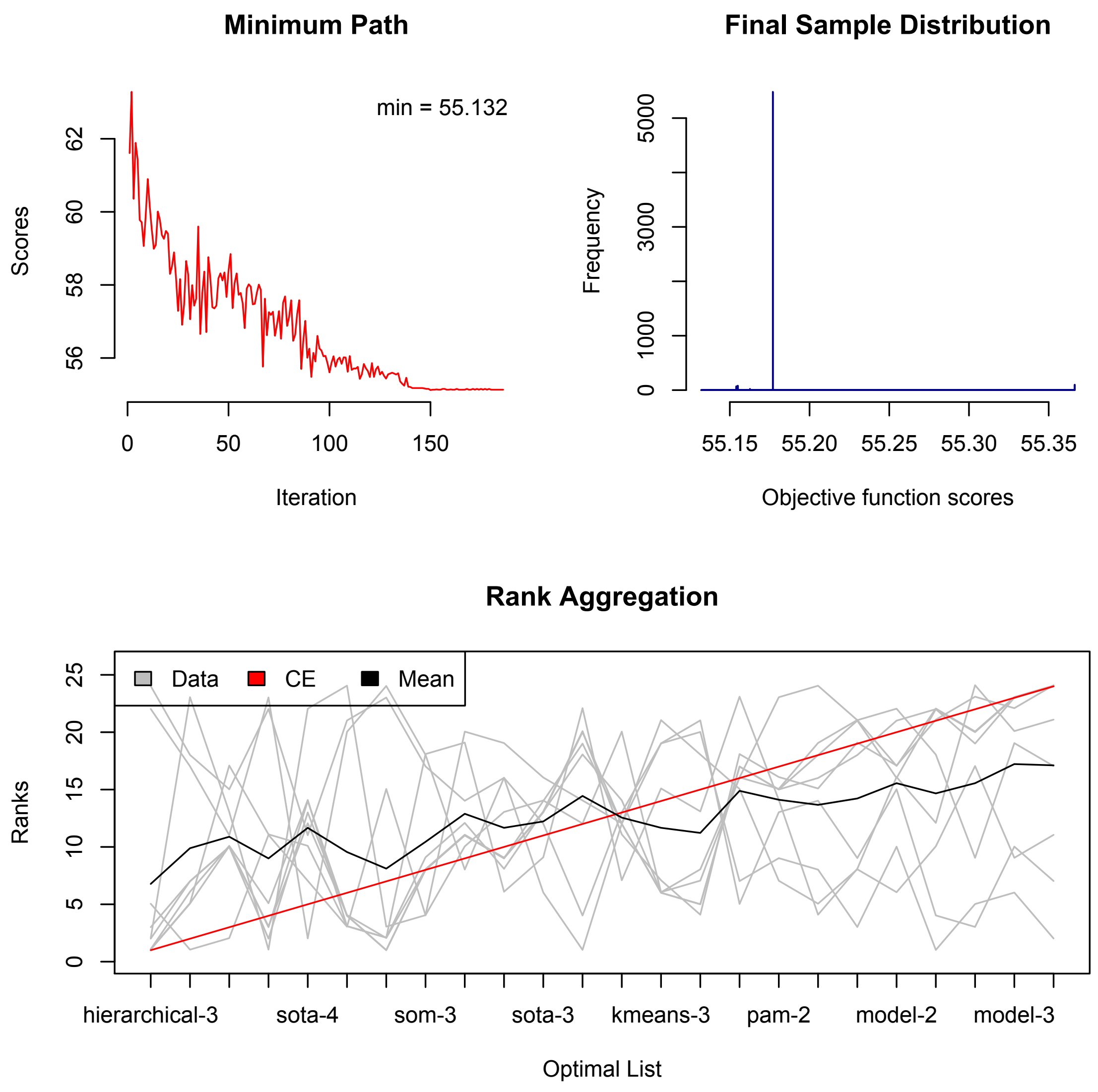

Figure 5: Visual representation of "CE" rank aggregation results for Example 2 using the "Spearman" distance. The top left plot shows the minimum values of the objective function as the number of iterations increases. The top right plot is a histogram of the objective function scores at the last iteration. The bottom plot shows the individual ranks from the data (in grey), the final solution (in red), and average ranking (in black).

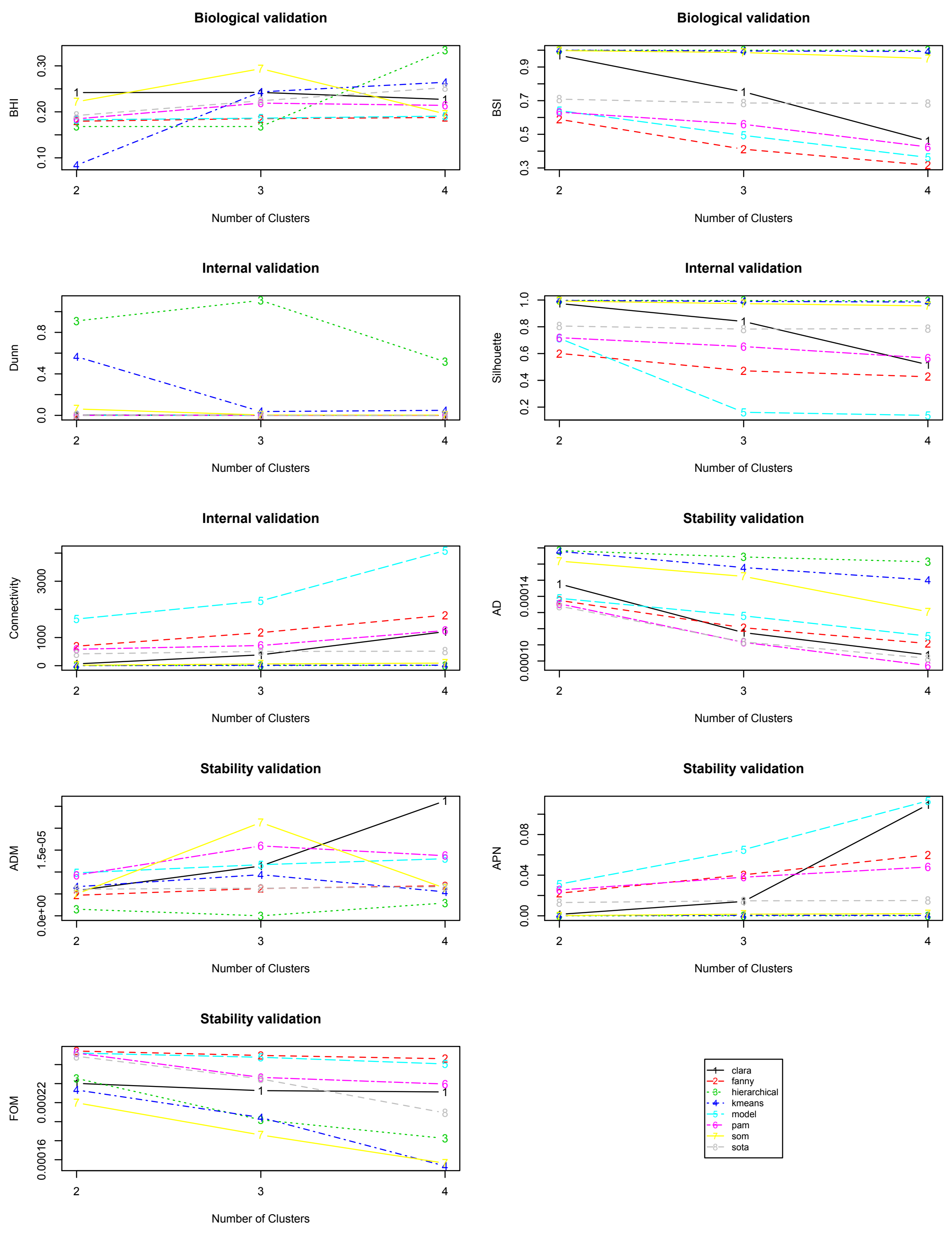

Figure 6: Plots of all nine validation measures for Example 2. The BHI, BSI, Dunn index, and silhouette width measures should be maximized. The AD, ADM, APN, Connectivity, and FOM measures should be minimized.

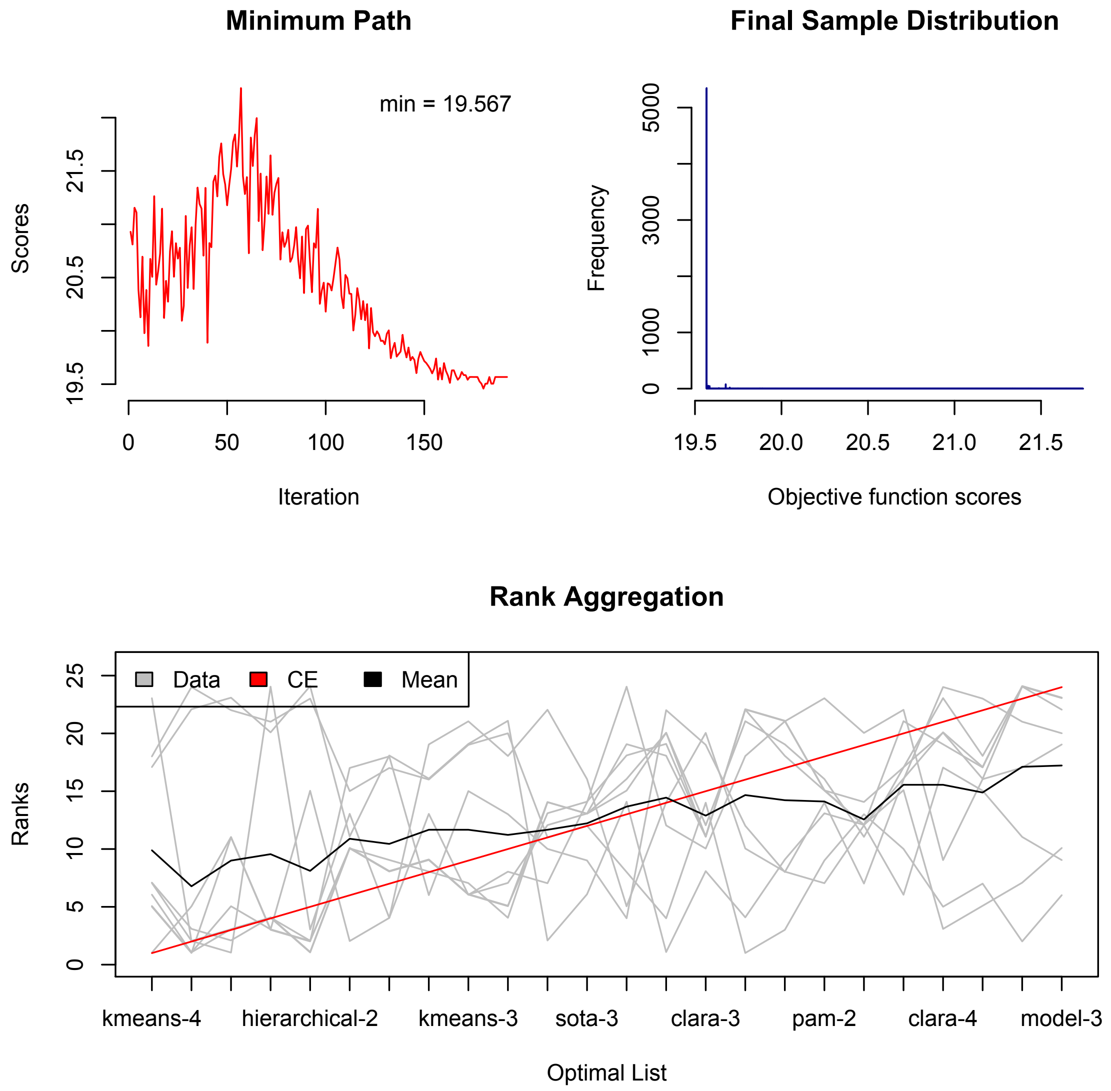

Figure 7: Visual representation of "CE" rank aggregation results for Example 2 using the "Kendall" distance. The top left plot shows the minimum values of the objective function as the number of iterations increases. The top right plot is a histogram of the objective function scores at the last iteration. The bottom plot shows the individual ranks from the data (in grey), the final solution (in red), and average ranking (in black).

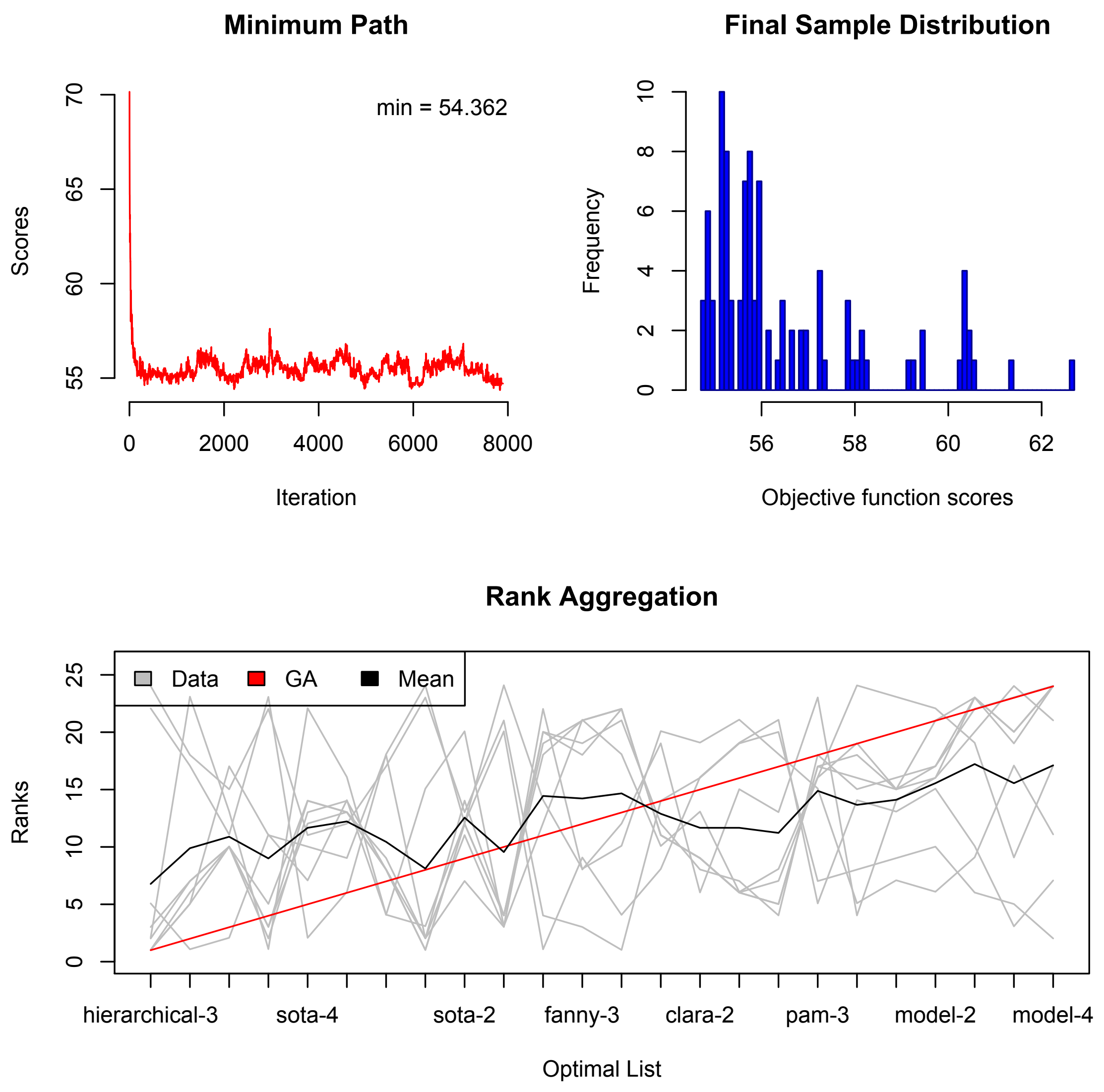

Figure 8: Visual representation of "GA" rank aggregation results for Example 2 using the "Spearman" distance. The top left plot shows the minimum values of the objective function as the number of iterations increases. The top right plot is a histogram of the objective function scores at the last iteration. The bottom plot shows the individual ranks from the data (in grey), the final solution (in red), and average ranking (in black).

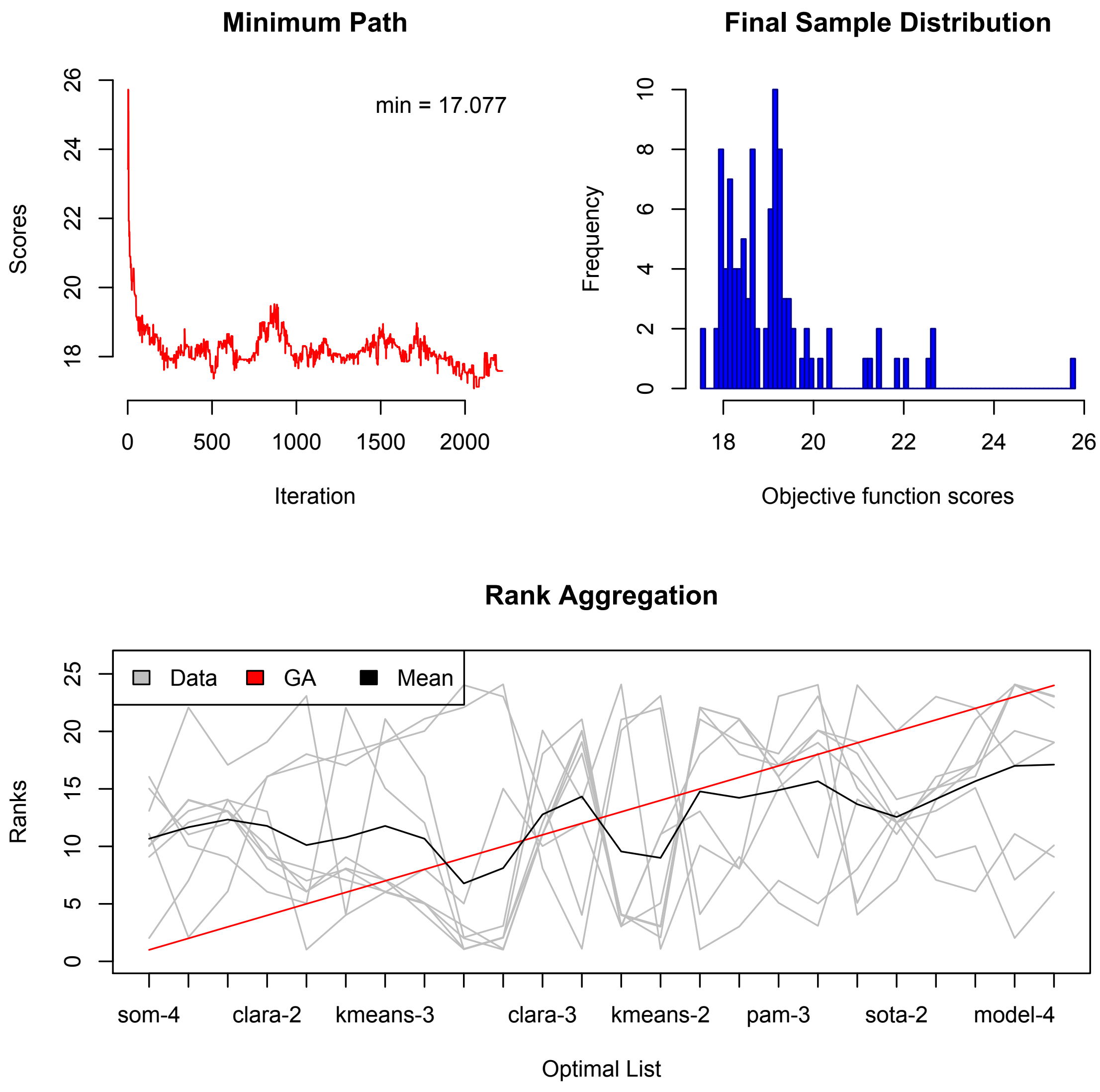

Figure 9: Visual representation of "GA" rank aggregation results for Example 2 using the "Kendall" distance. The top left plot shows the minimum values of the objective function as the number of iterations increases. The top right plot is a histogram of the objective function scores at the last iteration. The bottom plot shows the individual ranks from the data (in grey), the final solution (in red), and average ranking (in black).

CHAPTER VII

CONCLUSIONS AND FUTURE RESEARCH

The package **optCluster** is introduced in this thesis as a simple and convenient option to add to the many cluster analysis packages already available for R. The *optCluster( )* function, in this package, offers the user nine validation measures categorized into three types, “biological”, “internal”, and “stability”, as well as ten unique clustering algorithms to be used in cluster analysis. A weighted rank aggregation using either a cross-entropy Monte Carlo algorithm or a Genetic algorithm determines the optimal clustering algorithm along with the optimal number of clusters for a given dataset. The **optCluster** package also provides easy to use methods that give the user access to the graphical capabilities of the **clValid** and **RankAggreg** packages.

It is recommended that the user run the *optCluster( )* function several times with different arguments to compare results. This is especially important for the cluster size range because the rank aggregation methods may choose the upper bound of this range as the optimal number of clusters. Changing the range of cluster sizes or the selections of clustering algorithms will often produce different results, so choices in arguments may need to be fine-tuned in order to discover the overall best result. The mouse microarray dataset (Bhattacherjee et al., 2007) analyzed in Chapter VI provided an example of producing different results using different arguments. This set of data was used as an example of the **clValid** package by Brock et al. (2011), where three clustering algorithms were analyzed using “internal” and “stability” validation measures over the range of four to six clusters. PAM with six clusters was chosen as the optimal clustering method using the cross-entropy Monte Carlo weighted rank aggregation from the **RankAggreg** package. The same dataset was analyzed in this thesis with the *optCluster( )* function, but ten clustering algorithms were evaluated using all three types of validation measures over a larger range of two to nine clusters. Under these new conditions, SOM with eight clusters was chosen as the best clustering algorithm with rank aggregation using the cross-entropy Monte Carlo method, a different result with a different selection of arguments.

It is also recommended that the user tests the consistency of the rank aggregation results, and the **optCluster** package provides the *repRankAggreg( )* function as a useful tool to do so. This function repeats the weighted rank aggregation using same weighted rank aggregation method, ranked clustering algorithm lists, and validation score lists as the original *optCluster( )* function. A different aggregation algorithm or type of distance measure for weighted rank aggregation can also be evaluated using the *repRankAggreg( )* function, but doing so may affect the final results. This was the case for analysis of the *Arabidopsis thaliana* RNA-Seq dataset (Di et al., 2011) in Chapter VI. Using weighted Spearman's footrule distance, the optimal clustering algorithm and number of clusters was determined to be UPGMA with three clusters. By only changing the rank aggregation distance to weighted Kendall's tau, the results changed and K-means with four clusters was chosen as the optimal algorithm and cluster size.

Future Research

As more and more research is being done on cluster analysis, especially in microarray gene expression data and data generated from Next Generation Sequencing technology, new tools and resources for analysis are becoming readily available. The Comprehensive R Archive Network (CRAN) provides a whole slew of packages offering different clustering algorithms and validation measures for researchers to use. Submitting the **optCluster** package to CRAN is the next step following this paper, and there are several opportunities to develop this package into a robust cluster analysis tool before submission.

Currently, there are only ten clustering algorithms and nine validation measures available through this package. Extending the number of clustering algorithms available, including those specifically designed for RNA-Seq data like those found in the package **MBCluster.Seq** (Si, 2015), would enhance the analysis capabilities of the **optCluster** package. There are packages available for R that offer more validation measures for researchers to use, such as the package **NbClust** (Charrad et al., 2014), which has a collection of 30 different measures for users choose from. Adding more validation measures to the **optCluster** package would also be a way to increase its usefulness as a tool for cluster analysis.

The **optCluster** package is able to analyze small subsets (less than 1000 rows) of high dimensional data across a small range of three or four cluster numbers in a fairly time-efficient manner. While this may be sufficient in some cases, it may be more worthwhile to analyze the entire dataset over a large range of cluster numbers. When the datasets get larger and/or the range for the number of clusters increases, the amount of time it takes for the *optCluster( )* function to obtain a result becomes lengthy.

Both examples in Chapter VI experienced issues with long running times while using the *optCluster( )* function. The computations for the *Arabidopsis thaliana* RNA-

Seq dataset took over 85 hours on the Cardinal Research Cluster (CRC) to complete using only eight of the ten available clustering algorithms and a range of three different numbers of clusters. The two excluded algorithms (Agnes and Diana) were omitted from the analysis because they took just too long to perform validation measures given the time constraints for this paper. The weighted rank aggregation for the mouse microarray data using either the Genetic algorithm or the weighted Kendall's tau distance did not converge after 168 hours on the CRC. The large range of cluster sizes resulted in long lists to be used in the aggregation methods, and therefore, required more time to complete all of the computations and iterations. With the abundance of resources available for high-performance and parallel computing in R, increasing the speed of the **optCluster** package would make it even more valuable for cluster analysis, especially for high dimensional data.

REFERENCES

Anderberg, M. R. (1973). *Cluster Analysis for Applications*. Academic Press: New York.

Arbelaitz, O., Gurrutxaga, I., Muguerza, J., Pérez, J., & Perona, I. (2013). An extensive comparative study of cluster validity indices. *Pattern Recognition, 46*, 243-256.

Bhattacherjee, V., Mukhopadhyay, P., Singh, S., Johnson, C., Philipose, J. T., Warner, C. P., ... & Pisano, M. M. (2007). Neural crest and mesoderm lineage-dependent gene expression in orofacial development. *Differentiation*, *75*, 463-477.

Brock, G., Pihur, V., Datta, S., & Datta, S. (2011). clValid, an R package for cluster validation. *Journal of Statistical Software (Brock et al., March 2008)*.

Carlson, M. (2015a). *GO.db: A set of annotation maps describing the entire Gene Ontology*. R package version 3.0.0.

Carlson, M. (2015b) *moe430a.db: Affymetrix Mouse Expression Set 430 annotation data (chip moe430a)*. R package version 3.0.0.

Carlson, M. (2015c) *org.At.tair.db: Genome wide annotation for Arabidopsis*. R package version 3.0.0.

Charrad, M., Ghazzali, N., Boiteau, V., & Niknafs, A. (2014). NbClust: An R Package for Determining the Relevant Number of Clusters in a Dataset. *Journal of Statistical Software, 61*:*6*.

Chen, A., Sweeney, T.E., Gevaert, O. (2015). COMMUNAL: Robust Selection of Cluster Number K. R package version 1.0, URL http://CRAN.R-project.org/package=COMMUNAL.

Datta, S., & Datta, S. (2003). Comparisons and validation of statistical clustering techniques for microarray gene expression data. *Bioinformatics, 19*, 459-466.

Datta, S., & Datta, S. (2006). Evaluation of clustering algorithms for gene expression data using a reference set of functional classes. *BMC Bioinformatics, 7:397.*

DeConde, R., Hawley, S., Falcon, S., Clegg, N., Knudsen, B., & Etzioni, R. (2006). Combining Results of Microarray Experiments: A Rank Aggregation Approach. *Statistical Applications in Genetics and Molecular Biology, 5*.

Dempster, A. P., Laird, N. M., & Rubin, D. B. (1977). Maximum likelihood from incomplete data via the EM algorithm. *Journal of the royal statistical society. Series B (methodological)*, 1-38.

Di, Y., Schafer, D. W., Cumbie, J. S., & Chang, J. H. (2011). The NBP negative binomial model for assessing differential gene expression from RNA-Seq. *Statistical Applications in Genetics and Molecular Biology*, *10*, 1-28.

Di, Y., Schafer, D. W., Cumbie, J. S., & Chang, J. H. (2015). NBPSeq: Negative Binomial Models for RNA-Sequencing Data. R package version 0.3.0, URL http://CRAN.R-project.org/package=NBPSeq.

Dimitriadou, E. (2014). cclust: Convex Clustering Methods and Clustering Indexes. R package version 0.6-19, URL http://CRAN.R-project.org/package=cclust.

Dopazo, J., & Carazo, J. M. (1997). Phylogenetic reconstruction using an unsupervised growing neural network that adopts the topology of a phylogenetic tree. *Journal of molecular evolution*, *44*, 226-233.

Dunn, J. (1974). Well-Separated Clusters and Optimal Fuzzy Partitions. *Journal of Cybernetics, 4*, 95-104.

Dwork, C., Kumar, R., Naor, M., & Sivakumar, D. (2001). Rank aggregation methods for the web. *Proceedings of the 10th international conference on World Wide Web*, 613-622.

Fagin, R., Kumar, R., & Sivakumar, D. (2003). Comparing Top k Lists. *SIAM Journal on Discrete Mathematics, 17*, 134-160.

Fraley, C., & Raftery, A. (2002). Model-Based Clustering, Discriminant Analysis, and Density Estimation. *Journal of the American Statistical Association, 97*, 611-631.

Fraley, C., Raftery, A., & Scrucca, L. (2014). mclust: Normal Mixture Modeling for Model-Based Clustering, Classification, and Density Estimation. R package version 4.4, URL http://CRAN.R-project.org/package=mclust.

Gentleman, R. (2014). *annotate: Annotation for microarrays*. R package version 1.44.0

Gentleman, R. C., Carey, V. J., Bates, D. M., Bolstad, B., Dettling, M., Dudoit, S., ... & Zhang, J. (2004). Bioconductor: open software development for computational biology and bioinformatics. *Genome biology*, *5*(10), R80.

Goldberg, D. (1989). *Genetic algorithms in search, optimization, and machine learning*. Reading, Mass.: Addison-Wesley Pub.

Handl, J., Knowles, J., & Kell, D. (2005). Computational cluster validation in post-genomic data analysis. *Bioinformatics, 21*, 3201-3212.

Hartigan, J.A., & Wong, M.A. (1979). A K-Means Clustering Algorithm. *Applied Statistics*, *28*, 100-108.

Hennig, C. (2015). fpc: Flexible procedures for clustering. R package version 2.1-9, URL http://CRAN.R-project.org/package=fpc.

Herrero, J., Valencia, A., & Dopazo, J. (2001). A hierarchical unsupervised growing neural network for clustering gene expression patterns. *Bioinformatics, 17*, 126-136.

Kane, M., Emerson, J., Weston, S. (2013). Scalable Strategies for Computing with Massive Data. *Journal of Statistical Software, 55:14*.

Kaufman, L., & Rousseeuw, P. (1990). *Finding Groups in Data: An Introduction to Cluster Analysis*. New York: Wiley.

Kohonen, T. (2001). *Self-organizing maps* (3rd ed.). Berlin: Springer.

Lam, Y., & Tsang, P. (2012). EXploratory K-Means: A new simple and efficient algorithm for gene clustering. *Applied Soft Computing, 12*, 1149-1157.

Lin, S., & Ding, J. (2009). Integration of ranked lists via Cross Entropy Monte Carlo with applications to mRNA and microRNA studies. *Biometrics*, *65*, 9-18.

Maechler, M., Rousseeuw, P., Struyf, A., Hubert, M., & Hornik, K. (2015). cluster: Cluster Analysis Extended Rousseeuw et al. R package version 2.0.1. URL http://CRAN.R-project.org/package=cluster.

Mavridis, L., Nath, N., & Mitchell, J. (2013). PFClust: A novel parameter free clustering algorithm. *BMC Bioinformatics, 14:213*.

McLachlan, G., & Krishnan, T. (1997). *The EM algorithm and extensions*. New York: Wiley.

Nazeer, K., Sebastian, M., & Kumar, S. (2013). A novel harmony search-K means hybrid algorithm for clustering gene expression data. *Bioinformation, 9*, 84-88.

Nieweglowski L (2013). clv: Cluster Validation Techniques. R package version 0.3-2.1, URL http://CRAN.R-project.org/package=clv.

Pihur, V., Datta, S., & Datta, S. (2007). Weighted rank aggregation of cluster validation measures: A Monte Carlo cross-entropy approach. *Bioinformatics, 23*, 1607-1615.

Pihur, V., Datta, S., & Datta, S. (2009). RankAggreg, an R package for weighted rank aggregation. *BMC Bioinformatics, 10:62*.

R Core Team. (2014) R: A Language and Environment for Statistical Computing . R Foundation for Statistical Computing, Vienna, Austria. URL http://www.R-project.org.

Rendón, E., Abundez, I., Arizmendi, A., & Quiroz, E. M. (2011). Internal versus External cluster validation indexes. *International Journal of computers and communications*, *5*, 27-34.

Rousseeuw, P. (1987). Silhouettes: A graphical aid to the interpretation and validation of cluster analysis. *Journal of Computational and Applied Mathematics, 20*, 53-65.

Rubinstein, R. Y. (1997). Optimization of computer simulation models with rare events. *European Journal of Operational Research*, *99*, 89-112.

Si, Y. (2015). MBCluster.Seq: Model-based Clustering for RNA-seq Data. R package version 1.0, URL http://CRAN.R-project.org/package= MBCluster.Seq.

Si, Y., Liu, P., Li, P., & Brutnell, T. (2014). Model-based clustering for RNA-seq data. *Bioinformatics, 30*, 197-205.

Sneath, P., & Sokal, R. (1973). *Numerical taxonomy: The principles and practice of numerical classification*. San Francisco: W.H. Freeman.

Tseng, G., & Wong, W. (2005). Tight Clustering: A Resampling-Based Approach for Identifying Stable and Tight Patterns in Data. *Biometrics*, *61*, 10-16.

Thalamuthu, A., Mukhopadhyay, I., Zheng, X., & Tseng, G. (2006). Evaluation and comparison of gene clustering methods in microarray analysis. *Bioinformatics, 22*, 2405-2412.

Theodoridis, S., & Koutroumbas, K. (2006). *Pattern recognition* (3rd ed.). Amsterdam: Elsevier/Academic Press.

Walesiak, M., & Dudek, A. (2014). clusterSim: Searching for Optimal Clustering Procedure for a Dataset. R package version 0.44-1, URL http://CRAN.R-project.org/package= clusterSim.

Ward Jr, J. H. (1963). Hierarchical grouping to optimize an objective function. *Journal of the American statistical association*, *58*, 236-244.

Wehrens, R., & Buydens, L.M.C. (2007). Self- and Super-organising Maps in R: the kohonen package. *Journal of Statistical Software, 21:5*.

Witten, D. (2011). Classification and clustering of sequencing data using a Poisson model. *The Annals of Applied Statistics, 5*, 2493-2518.

Wu, F. (2008). Genetic weighted k-means algorithm for clustering large-scale gene expression data. *BMC Bioinformatics, 9*, S12-S12.

Yeung, K., Haynor, D., & Ruzzo, W. (2001). Validating clustering for gene expression data. *Bioinformatics, 17*, 309-318.

APPENDIX A

R CODE

optCluster Package Code

```
########### optCluster Class Definitions ############

## Change raggr to S4 class
setOldClass("raggr")
setClass("optCluster",representation(clVal ="clValid",
                                  optMethod ="character",
                                        rankAgg="raggr"))

########### optCluster Function ###########

## Determine optimal clustering method
optCluster <- function(obj, nClust, clMethods = "all", validation =
 c("internal", "stability"), hierMethod = "average", clVerbose = FALSE,
 rankMethod = "CE", rankVerbose= FALSE,...) {

 if("all" %in% clMethods) {
 clMethods <- c("agnes", "clara", "diana", "fanny", "hierarchical", "kmeans",
 "model", "pam", "som", "sota")
    }

     if("all" %in% validation) {
 validation = c("internal", "stability", "biological")
    }

 addArgs <- list(...)
 ## 'verbose' and 'method' are both used in clValid and RankAggreg
 if(exists(c("verbose"), where = addArgs)){
     stop("must specify 'verbose' as 'clVerbose' or 'rankVerbose'")
 }
 if(exists(c("method"), where = addArgs)){
     stop("must specify 'method' as 'hierMethod' or 'rankMethod'")
 }

 ## Sort arguments to clValid or RankAggreg
 RankAggreg.names <- c(names(formals(RankAggreg)),"p")
 RankAggreg.args <- addArgs[names(addArgs) %in% RankAggreg.names]
 if(length(RankAggreg.args) > 0){
     clValid.args <- addArgs[-which(names(addArgs) %in% RankAggreg.names)]
 } else{
     clValid.args <- addArgs
 }
 ## Obtain validation measures for clustering results, cluster ranks and weights
 clusters <- do.call('clValid',c(list(obj, nClust, clMethods, validation, method
 = hierMethod, verbose = clVerbose),
              clValid.args))
 cluster.order <- getRanksWeights(clusters)

 if(exists(c("k"), where = RankAggreg.args)){
     nList <- RankAggreg.args$k
     RankAggreg.args <- RankAggreg.args[-which(names(RankAggreg.args) == "k")]
 } else {
     nList <- ncol(cluster.order$ranks)
 }

 if(exists(c("weights"), where = RankAggreg.args)){
     rank.weight <- RankAggreg.args$weights
     RankAggreg.args <- RankAggreg.args[-which(names(RankAggreg.args) ==
 "weights")]
 } else {
     rank.weight <- cluster.order$weights
 }

 ## Perform weighted rank aggregation
 optimal.list <- do.call('RankAggreg', c(list(x = cluster.order$ranks, k =
 nList, weights = rank.weight,
 method = rankMethod, verbose = rankVerbose), RankAggreg.args))

 ## Create 'optCluster' class object
 new("optCluster", rankAgg = optimal.list, clVal = clusters, optMethod =
 optimal.list$top.list[1])

}

########## repRankAggreg Function ##########

## Repeat weighted rank aggregation on "optCluster" object
repRankAggreg <- function(optObj, rankMethod = "same", distance = "same",
 rankVerbose = FALSE, ... ){

 RankAggreg.args <- list(...)

 clusters <- cl.valid(optObj)
 rankAgg <- rank.aggreg(optObj)
 cluster.order <- getRanksWeights(clusters)
 method <- match.arg(rankMethod, c("same", "CE", "GA"))
 distance <- match.arg(distance, c("same", "Spearman", "Kendall"))

 if(method == "same") {
     method <- rankAgg$method
 }

 if(distance == "same") {
     distance <- rankAgg$distance
 }

 if(exists(c("method"), where = RankAggreg.args)){
     message(" The argument 'method' has been changed to 'rankMethod' ")
     rankMethod <- RankAggreg.args$method
     RankAggreg.args <- RankAggreg.args[-which(names(RankAggreg.args) ==
 "method")]
 }


 if(exists(c("verbose"), where = RankAggreg.args)){
     message(" The argument 'verbose' has been changed to 'rankVerbose' ")
     rankVerbose <- RankAggreg.args$verbose
     RankAggreg.args <- RankAggreg.args[-which(names(RankAggreg.args) ==
 "verbose")]
 }        }

 if(exists(c("k"), where = RankAggreg.args)){
     nList <- RankAggreg.args$k
     RankAggreg.args <- RankAggreg.args[-which(names(RankAggreg.args) == "k")]
 } else {
     nList <- ncol(cluster.order$ranks)
 }

 if(exists(c("weights"), where = RankAggreg.args)){
     rank.weight <- RankAggreg.args$weights
     RankAggreg.args <- RankAggreg.args[-which(names(RankAggreg.args) ==
 "weights")]
 } else {
     rank.weight <- cluster.order$weights
 }

 ## Perform weighted rank aggregation
 optimal.list <- do.call('RankAggreg', c(list(x = cluster.order$ranks, k =
 nList, weights = rank.weight,
 method = method, distance = distance, verbose = rankVerbose), RankAggreg.args))

 ## Create 'optCluster' class object
 new("optCluster", rankAgg = optimal.list, clVal = clusters, optMethod =
 optimal.list$top.list[1])
}

########### optCluster Methods ###########

########### Create Accessor Functions ##########

## cl.valid accessor
setGeneric("cl.valid", function(object, ...) standardGeneric("cl.valid"))
setMethod("cl.valid",signature(object="optCluster"),
          function(object) return(object@clVal))

## rank.aggreg accessor
setGeneric("rank.aggreg", function(object, ...) standardGeneric("rank.aggreg"))
```

```
setMethod("rank.aggreg",signature(object="optCluster"),
          function(object) return(object@rankAgg))

## top.method accessor
setGeneric("top.method", function(object, ...) standardGeneric("top.method"))
setMethod("top.method",signature(object="optCluster"),
          function(object) return(object@optMethod))

########### Print, Show, and Summary Methods ##########

setMethod("print","optCluster",
          function(x) {
            cat("\nThe overall optimal method with number of clusters is: \n\t
 ", top.method(x), "\n\n")
            print(rank.aggreg(x))
          })

setMethod("show","optCluster",
          function(object) {
            cat("\nThe overall optimal method with number of clusters is: \n\t
 ", top.method(object), "\n\n")
            print(rank.aggreg(object))
          })

setMethod("summary","optCluster",
          function(object) {
            cat(summary(cl.valid(object)), "\nThe overall optimal method with
 number of clusters is: \n\t   ",
            top.method(object), "\n\n")
            print(rank.aggreg(object))
          })
```

Example 1 Analysis Code

```
library(optCluster)

## load mouse data from clValid
data(mouse)
ex1 <- mouse[, c("M1", "M2", "M3", "NC1", "NC2", "NC3")]
rownames(ex1) <- mouse$ID

## Run optCluster Function and Record Time
start.time <- Sys.time()
optMouse <-optCluster(ex1, 2:9, clMethods = “all”, validation = “all”, seed =
 123, annotation = "moe430a.db", maxIter = 1500)
end.time <- Sys.time()
time.taken <- end.time - start.time
time.taken

## Results
top.method(optMouse)
print(optMouse)
summary(optMouse)
```

```
## Validation Scores
valMouse <- cl.valid(optMouse)
optimalScores(valMouse)

## Validation Plots
par(mfrow = c(5,2))
plot(valMouse, measure = c("BHI", "BSI"), legend = FALSE)
plot(valMouse, measure = c("Dunn", "Silhouette", "Connectivity"), legend =
 FALSE)
plot(valMouse, measure = c("AD", "ADM", "APN", "FOM"), legend = FALSE)
plot(nClusters(valMouse), measures(valMouse, "AD")[, , 1], type = "n", axes =
 F, xlab = " ", ylab = " ")
legend("center", clusterMethods(valMouse), col = 1:10, lty = 1:10, pch =
 paste(c(1:9,0)), cex = 0.8)

## Aggregation Results
aggMouse <- rank.aggreg(optMouse)
aggMouse
aggMouse$num.it
plot(aggMouse)
which(aggMouse$top.list == "agnes-2")
```

Example 2 Analysis Code

```
library(NBPSeq)
data(arab)
ex2 <- t(arab)/colSums(arab)
ex2 <- t(ex2)
## Create File-backed Matrix
ex2 <- as.big.matrix(ex2, backingfile = "arab.big.txt", backingpath =
 "/scratch/home/mnseku01")

## Run optCluster Function and Record Time
start.time <- Sys.time()
optArabid <- optCluster(ex2[,], 2:4, clMethods = c("clara", "fanny",
     "hierarchical", "kmeans", "model", "pam", "som", "sota"), validation =
     "all", seed = 123, annotation = "org.At.tair.db", maxitems = nrow(ex2[,]))
print(optArabid)
end.time <- Sys.time()
time.taken <- end.time - start.time
time.taken

save.image("arabTest24.Rdata")

## Plot of Aggregation
aggArabid <- rank.aggreg(optArabid)
plot(aggArabid)
aggArabid$num.it

## Validation Scores
valArabid <- cl.valid(optArabid)
optimalScores(valArabid)
```

```
## Results
summary(optArabid)

## Validation Plots
par(mfrow = c(5,2))
plot(valArabid, measure = c("BHI", "BSI"), legend = FALSE)
plot(valArabid, measure = c("Dunn", "Silhouette", "Connectivity"), legend =
 FALSE)
plot(valArabid, measure = c("AD", "ADM", "APN", "FOM"), legend = FALSE)
plot(nClusters(valArabid), measures(valArabid, "AD")[, , 1], type = "n", axes =
 F, xlab = " ", ylab = " ")
legend("center", clusterMethods(valArabid), col = 1:8, lty = 1:8, pch =
 paste(c(1:8)), cex = 0.8)

## Additional Rank Aggregation Using "Kendall" Distance
KenArabid <- repRankAggreg(optArabid, distance = “Kendall”)
print(KenArabid)
plot(KenArabid)
```

APPENDIX B

ADDITIONAL R OUTPUT FROM CHAPTER VI

Example 1 *print( )* Output

```
The overall optimal method with number of clusters is:
        som-8

The optimal list is:
        som-8 som-7 clara-8 kmeans-9 hierarchical-9 agnes-9 agnes-8 clara-9 kmeans-8 pam-6
        hierarchical-8 pam-9 agnes-7 pam-8 hierarchical-7 fanny-7 pam-7 kmeans-7 diana-8 sota-9 som-6
        som-3 diana-9 agnes-6 som-9 sota-8 agnes-5 clara-7 hierarchical-6 hierarchical-5 fanny-9 som-4
        clara-6 fanny-6 fanny-5 fanny-8 diana-4 kmeans-3 diana-6 agnes-4 sota-7 diana-7 diana-5
        kmeans-6 agnes-2 som-5 fanny-3 kmeans-2 kmeans-5 hierarchical-2 clara-5 fanny-4 hierarchical-4
        sota-2 diana-3 diana-2 hierarchical-3 kmeans-4 sota-3 agnes-3 som-2 pam-5 sota-6 clara-4
        fanny-2 sota-5 pam-3 clara-3 clara-2 pam-2 model-2 sota-4 pam-4 model-3 model-7 model-8
        model-9 model-6 model-5 model-4

  Algorithm:   CE
  Distance:    Spearman
  Score:       239.4503
```

Example 1 *summary( )* Output

```
Clustering Methods:
 agnes clara diana fanny hierarchical kmeans model pam som sota

Cluster sizes:
 2 3 4 5 6 7 8 9

Validation Measures:
                                 2       3       4       5       6       7       8       9

agnes        APN            0.0478  0.1288  0.1755  0.1689  0.1516  0.0927  0.1095  0.1562
             AD             3.2430  2.6814  2.2571  2.0642  1.8732  1.6962  1.5925  1.5290
             ADM            0.4283  1.0953  0.8070  0.6196  0.5867  0.5475  0.5286  0.5463
             FOM            1.0658  0.8678  0.7451  0.6823  0.6371  0.5949  0.5388  0.5029
             Connectivity   5.3270 14.2528 20.7520 27.0726 30.6194 30.6194 36.1615 40.6222
             Dunn           0.1291  0.0788  0.0857  0.0899  0.0899  0.1203  0.1419  0.1467
             Silhouette     0.5133  0.4195  0.3700  0.3343  0.3233  0.3808  0.3655  0.3582
             BHI            0.2781  0.2783  0.2732  0.2543  0.2515  0.2602  0.2450  0.2236
             BSI            0.7950  0.5293  0.3389  0.3178  0.2848  0.2657  0.2287  0.1947
clara        APN            0.1099  0.2199  0.2798  0.3108  0.3061  0.1269  0.1240  0.1518
             AD             2.9902  2.5945  2.3069  2.1053  1.9024  1.5114  1.3948  1.3075
             ADM            0.4907  0.9201  0.9264  1.0816  1.0271  0.4210  0.3399  0.3635
             FOM            1.0103  0.8251  0.6923  0.6671  0.5239  0.4930  0.4557  0.4384
             Connectivity  18.7028 27.9651 44.8234 35.5159 26.1238 33.8361 47.3369 52.3262
             Dunn           0.0287  0.0597  0.0660  0.0761  0.0857  0.0671  0.1127  0.0882
             Silhouette     0.4257  0.3489  0.3304  0.3636  0.3836  0.4146  0.3997  0.3892
             BHI            0.2808  0.2760  0.2778  0.2786  0.2798  0.2795  0.2649  0.2675
             BSI            0.4948  0.3701  0.2736  0.2322  0.1931  0.1720  0.1517  0.1384
diana        APN            0.0634  0.1158  0.1024  0.1861  0.2274  0.2034  0.1225  0.1286
             AD             2.9133  2.5593  2.0303  1.9870  1.9122  1.7617  1.5439  1.4777
             ADM            0.3136  0.4888  0.3180  0.6023  0.6932  0.5596  0.4428  0.4055
             FOM            0.9774  0.8403  0.6816  0.6435  0.6081  0.5620  0.5136  0.4844
             Connectivity  18.7552 25.1187 38.1242 38.8143 45.1349 53.2302 53.2302 58.8917
             Dunn           0.0315   0.0358   0.0492   0.0577   0.0646   0.0648   0.0813   0.0828
             Silhouette     0.4601   0.3705   0.3538   0.3378   0.3316   0.2766   0.3453   0.3174
             BHI            0.2767   0.2665   0.2648   0.2598   0.2446   0.2459   0.2564   0.2791
             BSI            0.5891   0.5111   0.3444   0.2909   0.2774   0.2502   0.2229   0.2005
fanny        APN            0.0691   0.1031   0.1515   0.1818   0.1001   0.0708   0.1582   0.1678
             AD             2.8999   2.3775   2.1079   1.9018   1.6229   1.4741   1.5446   1.5071
             ADM            0.2984   0.3278   0.4259   0.3996   0.2481   0.1894   0.3774   0.3636
             FOM            0.9891   0.8161   0.7244   0.6610   0.5554   0.5151   0.5305   0.5249
             Connectivity  19.8925  32.7579  42.7421  42.7992  55.6552  43.3813  60.8806  66.1210
             Dunn           0.0401   0.0430   0.0623   0.0700   0.0632   0.0816   0.0514   0.0514
             Silhouette     0.4332   0.3401   0.2877   0.2765   0.3624   0.3501   0.3082   0.2123
             BHI            0.2795   0.2754   0.2812   0.2764   0.2847   0.2867   0.2896   0.3117
             BSI            0.4955   0.3359   0.2481   0.1924   0.1652   0.1530   0.1440   0.1319
hierarchical APN            0.0478   0.1288   0.1755   0.1689   0.1516   0.0927   0.1095   0.1562
             AD             3.2430   2.6814   2.2571   2.0642   1.8732   1.6962   1.5925   1.5290
             ADM            0.4283   1.0953   0.8070   0.6196   0.5867   0.5475   0.5286   0.5463
             FOM            1.0658   0.8678   0.7451   0.6823   0.6371   0.5949   0.5388   0.5029
             Connectivity   5.3270  14.2528  20.7520  27.0726  30.6194  30.6194  36.1615  40.6222
             Dunn           0.1291   0.0788   0.0857   0.0899   0.0899   0.1203   0.1419   0.1467
             Silhouette     0.5133   0.4195   0.3700   0.3343   0.3233   0.3808   0.3655   0.3582
             BHI            0.2781   0.2783   0.2732   0.2543   0.2515   0.2602   0.2450   0.2236
             BSI            0.7950   0.5293   0.3389   0.3178   0.2848   0.2657   0.2287   0.1947
kmeans       APN            0.0603   0.0726   0.3146   0.2485   0.2470   0.1595   0.2183   0.1589
             AD             2.9001   2.2923   2.2529   1.9978   1.8389   1.6236   1.5344   1.3898
             ADM            0.3196   0.3101   1.0621   0.7151   0.6700   0.5213   0.5967   0.5253
             FOM            0.9745   0.7548   0.7114   0.6528   0.6074   0.5458   0.4983   0.4686
             Connectivity  13.2548  17.6651  37.3980  43.2655  50.6095  39.5567  40.3567  45.2631
             Dunn           0.0464   0.0873   0.0777   0.0815   0.0703   0.0998   0.1286   0.1148
             Silhouette     0.4571   0.4182   0.3615   0.3367   0.3207   0.3931   0.3780   0.4261
             BHI            0.2784   0.2776   0.2755   0.2561   0.2564   0.2606   0.2520   0.2569
             BSI            0.5775   0.3760   0.2856   0.2526   0.2148   0.2024   0.1704   0.1620
model        APN            0.1991   0.3388   0.4337   0.5815   0.3472   0.3291   0.3225   0.4630
             AD             3.6550   3.0524   3.7695   3.3632   2.8363   2.5435   2.5471   2.5032
             ADM            1.6991   1.2080   1.8851   2.1598   1.1909   1.0524   1.1315   1.4743
             FOM            1.1267   0.8286   0.9952   1.0347   0.8042   0.7910   0.7901   0.6456
             Connectivity  23.7373 121.6671  89.2726 111.0246  96.4258 126.3575 132.8135 159.5603
             Dunn           0.0240   0.0304   0.0232   0.0332   0.0231   0.0342   0.0387   0.0273
             Silhouette     0.3291   0.2131  -0.0106   0.0902   0.0694   0.0388  -0.0004  -0.0205
             BHI            0.2844   0.2729   0.2978   0.2708   0.3094   0.3265   0.3457   0.3277
             BSI            0.7013   0.4334   0.3673   0.3093   0.2510   0.2109   0.2154   0.1681
pam          APN            0.1318   0.2376   0.3658   0.3029   0.0486   0.1240   0.1037   0.2063
             AD             3.0382   2.5993   2.4492   2.0840   1.5272   1.4906   1.3517   1.3614
             ADM            0.6372   0.9733   1.3172   1.0164   0.1401   0.3927   0.2760   0.5385
             FOM            1.0092   0.8391   0.7663   0.6490   0.5158   0.4934   0.4632   0.4370
             Connectivity  18.7917  27.9651  30.9302  44.9671  32.9667  41.8925  45.4353  47.1845
             Dunn           0.0391   0.0597   0.0510   0.0761   0.0816   0.0627   0.0845   0.0882
             Silhouette     0.4271   0.3489   0.3563   0.3530   0.4152   0.4117   0.4120   0.3817
             BHI            0.2820   0.2760   0.2828   0.2784   0.2765   0.2749   0.2627   0.2608
             BSI            0.5083   0.3677   0.2737   0.2467   0.1995   0.1659   0.1513   0.1358
som          APN            0.0738   0.0627   0.1243   0.2399   0.1086   0.0424   0.0991   0.1802
             AD             2.9271   2.2836   2.0304   1.9441   1.5940   1.3853   1.3445   1.3432
             ADM            0.4203   0.2599   0.3698   0.6864   0.3242   0.1296   0.2713   0.4551
             FOM            0.9847   0.7532   0.6956   0.6419   0.5285   0.4760   0.4535   0.4329
             Connectivity  13.2548  16.3000  37.2611  43.0948  40.1087  32.3194  35.8984  53.4611
             Dunn           0.0464   0.0854   0.0554   0.0756   0.0514   0.0996   0.1425   0.1255
             Silhouette     0.4571   0.4185   0.3536   0.3261   0.3907   0.4175   0.4183   0.3761
             BHI            0.2784   0.2773   0.2711   0.2701   0.2771   0.2774   0.2631   0.2676
             BSI            0.5763   0.3780   0.2939   0.2287   0.1952   0.1536   0.1483   0.1339
sota         APN            0.0716   0.0830   0.3035   0.3466   0.2340   0.2319   0.1631   0.1690
             AD             2.9037   2.4119   2.3832   2.2777   1.9553   1.7958   1.5234   1.4429
             ADM            0.2866   0.2714   1.0898   1.1415   0.8108   0.8395   0.5103   0.4693
             FOM            0.9872   0.8133   0.6958   0.6511   0.6315   0.5673   0.5018   0.4767
             Connectivity  22.7690  30.1794  32.6333  41.8321  47.7548  47.7548  55.4425  58.6238
             Dunn           0.0351   0.0446   0.0459   0.0459   0.0509   0.0509   0.0768   0.0768
             Silhouette     0.4395   0.3682   0.3169   0.2887   0.3236   0.3514   0.3577   0.3617
             BHI            0.2795   0.2796   0.2831   0.2741   0.2793   0.2784   0.2710   0.2791
             BSI            0.5096   0.4386   0.2966   0.2620   0.2525   0.2249   0.1741   0.1485
```

```
Optimal Scores:

             Score  Method Clusters
APN          0.0424 som    7
AD           1.3075 clara  9
ADM          0.1296 som    7
FOM          0.4329 som    9
Connectivity 5.3270 agnes  2
Dunn         0.1467 agnes  9
Silhouette   0.5133 agnes  2
BHI          0.3457 model  8
BSI          0.7950 agnes  2

The overall optimal method with number of clusters is:
        som-8

The optimal list is:
        som-8 som-7 clara-8 kmeans-9 hierarchical-9 agnes-9 agnes-8 clara-9 kmeans-8 pam-6
        hierarchical-8 pam-9 agnes-7 pam-8 hierarchical-7 fanny-7 pam-7 kmeans-7 diana-8 sota-9 som-6
        som-3 diana-9 agnes-6 som-9 sota-8 agnes-5 clara-7 hierarchical-6 hierarchical-5 fanny-9 som-4
        clara-6 fanny-6 fanny-5 fanny-8 diana-4 kmeans-3 diana-6 agnes-4 sota-7 diana-7 diana-5
        kmeans-6 agnes-2 som-5 fanny-3 kmeans-2 kmeans-5 hierarchical-2 clara-5 fanny-4 hierarchical-4
        sota-2 diana-3 diana-2 hierarchical-3 kmeans-4 sota-3 agnes-3 som-2 pam-5 sota-6 clara-4
        fanny-2 sota-5 pam-3 clara-3 clara-2 pam-2 model-2 sota-4 pam-4 model-3 model-7 model-8
        model-9 model-6 model-5 model-4

  Algorithm:   CE
  Distance:    Spearman
  Score:       239.4503
```

Example 2 *summary( )* Output

```
Clustering Methods:
 clara fanny hierarchical kmeans model pam som sota

Cluster sizes:
 2 3 4

Validation Measures:
                                  2         3         4

clara        APN             0.0014    0.0142    0.1094
             AD              0.0001    0.0001    0.0001
             ADM             0.0000    0.0000    0.0000
             FOM             0.0002    0.0002    0.0002
             Connectivity   64.1647  384.4389 1208.9567
             Dunn            0.0032    0.0002    0.0000
             Silhouette      0.9737    0.8397    0.5169
             BHI             0.2419    0.2422    0.2270
             BSI             0.9692    0.7526    0.4600
fanny        APN             0.0223    0.0404    0.0599
             AD              0.0001    0.0001    0.0001
             ADM             0.0000    0.0000    0.0000
             FOM             0.0003    0.0003    0.0003
             Connectivity  693.1877 1173.5206 1794.4837
             Dunn            0.0000    0.0000    0.0000
             Silhouette      0.6012    0.4711    0.4267
             BHI             0.1796    0.1853    0.1878
             BSI             0.5889    0.4115    0.3166
hierarchical APN             0.0000    0.0000    0.0001
             AD              0.0002    0.0002    0.0002
             ADM             0.0000    0.0000    0.0000
             FOM             0.0002    0.0002    0.0002
             Connectivity    2.9290    5.8579   10.0159
             Dunn            0.9107    1.1082    0.5169
             Silhouette      0.9971    0.9963    0.9930
             BHI             0.1681    0.1681    0.3341
             BSI             1.0000    1.0000    0.9992
kmeans       APN             0.0003    0.0005    0.0004
             AD              0.0002    0.0002    0.0002
             ADM             0.0000    0.0000    0.0000
             FOM             0.0002    0.0002    0.0002
             Connectivity    5.8579   15.9651   16.1857
             Dunn            0.5659    0.0371    0.0488
             Silhouette      0.9968    0.9897    0.9846
             BHI             0.0841    0.2434    0.2643
             BSI             0.9994    0.9952    0.9925
model        APN             0.0312    0.0651    0.1131
             AD              0.0001    0.0001    0.0001
             ADM             0.0000    0.0000    0.0000
             FOM             0.0003    0.0003    0.0003
             Connectivity  1657.6734 2302.0004 4087.6429
             Dunn            0.0001    0.0000    0.0000
             Silhouette      0.7136    0.1614    0.1382
             BHI             0.1826    0.1863    0.1907
             BSI             0.6407    0.4935    0.3627
pam          APN             0.0254    0.0380    0.0480
             AD              0.0001    0.0001    0.0001
             ADM             0.0000    0.0000    0.0000
             FOM             0.0003    0.0002    0.0002
             Connectivity   583.8889  716.2687 1247.1345
             Dunn            0.0001    0.0001    0.0000
             Silhouette      0.7186    0.6522    0.5671
             BHI             0.1847    0.2190    0.2140
             BSI             0.6320    0.5594    0.4248
som          APN             0.0002    0.0016    0.0020
             AD              0.0002    0.0002    0.0001
             ADM             0.0000    0.0000    0.0000
             FOM             0.0002    0.0002    0.0002
             Connectivity    15.9623   61.1980   90.6306
             Dunn            0.0618    0.0039    0.0031
             Silhouette      0.9927    0.9729    0.9564
             BHI             0.2219    0.2939    0.1956
             BSI             0.9977    0.9860    0.9513
sota         APN             0.0131    0.0148    0.0150
             AD              0.0001    0.0001    0.0001
             ADM             0.0000    0.0000    0.0000
             FOM             0.0003    0.0002    0.0002
             Connectivity   418.9956  505.1250  515.1933
             Dunn            0.0001    0.0002    0.0002
             Silhouette      0.8057    0.7831    0.7869
             BHI             0.1920    0.2241    0.2527
             BSI             0.7089    0.6860    0.6846
Optimal Scores:

             Score  Method       Clusters
APN          0.0000 hierarchical 3
AD           0.0001 pam          4
ADM          0.0000 hierarchical 3
FOM          0.0002 kmeans       4
Connectivity 2.9290 hierarchical 2
Dunn         1.1082 hierarchical 3
Silhouette   0.9971 hierarchical 2
BHI          0.3341 hierarchical 4
BSI          1.0000 hierarchical 3


The overall optimal method with number of clusters is:
        hierarchical-3

The optimal list is:
        hierarchical-3 kmeans-4 som-4 kmeans-2 sota-4 hierarchical-4 hierarchical-2 som-3 clara-3
        clara-2 sota-3 pam-4 sota-2 kmeans-3 som-2 pam-3 pam-2 fanny-2 fanny-3 model-2 fanny-4 clara-4
        model-3 model-4
```

```
  Algorithm:   CE
  Distance:    Spearman
  Score:       55.13205
```

Example 2 *print( )* Output for *"GA"* and *"Spearman"* Rank Aggregation

```
> GAarabid <- repRankAggreg(optArabid, rankMethod = "GA", maxIter = 200000)
> print(GAarabid)

The overall optimal method with number of clusters is:
        hierarchical-3

The optimal list is:
        hierarchical-3 kmeans-4 som-4 kmeans-2 sota-4 sota-3 som-3 hierarchical-2 sota-2
        hierarchical-4 pam-4 fanny-3 fanny-4 clara-3 clara-2 kmeans-3 som-2 pam-3 fanny-2 pam-2
        model-2 model-3 clara-4 model-4

  Algorithm:   GA
  Distance:    Spearman
  Score:       54.36216
```

Example 2 *print( )* Output for *"GA"* and *"Kendall"* Rank Aggregation

```
> GAKarabid <- repRankAggreg(optArabid, rankMethod = "GA", distance = "Kendall", maxIter = 200000)
> print(GAKarabid)

The overall optimal method with number of clusters is:
        som-4

The optimal list is:
        som-4 sota-4 sota-3 clara-2 kmeans-4 som-3 kmeans-3 som-2 hierarchical-3 hierarchical-2
        clara-3 pam-4 hierarchical-4 kmeans-2 fanny-4 fanny-3 pam-3 clara-4 fanny-2 sota-2 pam-2
        model-2 model-4 model-3

  Algorithm:   GA
  Distance:    Kendall
  Score:       17.07676
```

CURRICULUM VITAE

Michael N. Sekula
1132 Goss Ave., Louisville, KY 40217
586-662-0256
mnseku01@louisville.edu

EDUCATION

Bachelor of Arts in Secondary Education May 2010
Saginaw Valley State University, University Center, MI
Major: **Mathematics** Minor: **Physics**
Summa Cum Laude

Master of Science in Biostatistics May 2015
University of Louisville, Louisville, KY

TEACHING EXPERIENCE

Master Educator Nov. 2014 – Present
Visitor Experience Educator Sept. 2013 – Nov. 2014
Kentucky Science Center, Louisville, KY

Science Teacher Aug. 2010 – June 2013
Pleasure Ridge Park High School, Louisville, KY

Math Tutor Aug. 2008 – Dec. 2009
Math Resource Center, University Center, MI

HONORS

ETS Recognition of Excellence for Mathematics: Content Knowledge 2010
Outstanding Senior in Mathematics Education 2010

ADDITIONAL ACTIVITIES

Assistant Coach, Pleasure Ridge Park High School Bowling Team 2010 - 2012
Member, Saginaw Valley State University's Chapter of Alpha Phi Omega 2009 - 2010
Volunteer, Camp Quality of Michigan 2008 - 2009

COMPUTING EXPERIENCE

Statistical Software: SAS, R
Document Markup: LaTeX
Scripting: Bash