Clustering of Biological Datasets in the Era of Big Data

Richard Röttger[1,*]

[1]Department of Mathematics and Computer Science, University of Southern Denmark, Campusvej 55, 5230 Odense, Denmark, `http://imada.sdu.dk/~roettger/`

Summary

Clustering is a long-standing problem in computer science and is applied in virtually any scientific field for exploring the inherent structure of datasets. In biomedical research, clustering tools have been utilized in manifold areas, among many others in expression analysis, disease subtyping or protein research. A plethora of different approaches have been developed but there is only little guideline what approach is the optimal in what particular situation. Furthermore, a typical cluster analysis is an entire process with several highly interconnected steps; from preprocessing, proximity calculation, the actual clustering to evaluation and optimization. Only when all steps seamlessly work together, an optimal result can be achieved. This renders a cluster analyses tiresome and error-prone especially for non-experts. A mere trial-and-error approach renders increasingly infeasible when considering the tremendous growth of available datasets; thus, a strategic and thoughtful course of action is crucial for a cluster analysis. This manuscript provides an overview of the crucial steps and the most common techniques involved in conducting a state-of-the-art cluster analysis of biomedical datasets.

1 Introduction

Modern technology enables and facilitates the generation of massive amounts of data and without doubt, this trend will continue. With current wet-lab technology, a vast amount of various kinds of data is produced at an ever increasing pace [1]. Just to illustrate this fact, GenBank stores almost 200 million sequences[1], over 6000 completed bacterial genomes are stored at the NCBI[2] (as of Oktober 2016) and attempts are made to boost the number of available whole human genomes in the hundreds of thousands[3]. This trend can be observed in virtually all different types of biological data and the sheer amount of data alone poses already considerable challenges to the researcher and requires sophisticated machine learning techniques.

Nevertheless, the availability of this data alone is only the beginning; extracting actual knowledge of this plethora of data is the actual challenge. John Naisbitt said in his 1982 book Megatrends [2] that ”We are drowning in information but starved for knowledge”; and this also holds true for biomedical data. In order to extract knowledge from an unknown dataset, clustering is a common first step for unraveling the inherent structure of the dataset. Clustering describes

[*]To whom correspondence should be addressed. Email: roettger@imada.sdu.dk
[1]Source: `https://www.ncbi.nlm.nih.gov/genbank/statistics/`
[2]Source: `https://www.ncbi.nlm.nih.gov/genome/browse/`
[3]The 100,000 Genomes Project `https://www.genomicsengland.co.uk/`

the unsupervised machine learning task of grouping similar objects together into so-called clusters. Clustering is long-standing problem in computer science and receives consistent attention from the research community. Due to its versatility, clustering is applied in almost all scientific fields, like economics and marketing, astronomy, archeology, and many others [3]. In the field of bioinformatics, cluster analyses are used, for example, for the analysis of microarry data [4, 5, 6], cancer subtyping [7], protein homology detection [8, 9, 10, 11, 12, 13], and many more. Even though the application fields for clustering are very different, the researcher of any field faces similar decisions and challenges along the way.

One of the major problems of clustering is the absence of a clear definition of a *good* clustering. Given a set of N objects $X = \{x_1, \dots, x_N\}$, the following types of clustering tasks can be differentiated [14, 15]:

Partitional Clustering (sometimes also called *crisp* or *disjoint* clustering) is the task of seeking a k-partition $C = \{C_1, \dots, C_k\}$ of X, such that

1. $C_i \neq \emptyset, \quad i = 1, \dots, k$
2. $\bigcup_{C_i \in C} C_i = X$
3. $C_i \cap C_j = \emptyset \quad i, j = 1, \dots, k$ and $i \neq j$

Overlapping Clustering follows the same principle as the partitional clustering with the difference that condition 3 does not hold, i.e., each object can be member of several clusters.

Fuzzy Clustering assigns each object x_i a degree of membership $u_{i,j}$ to each cluster C_j of the k partitioning $C = \{C_1, \dots, C_k\}$, such that

1. $\sum_{i=1}^{k} u_{i,j} = 1 \quad \forall j$
2. $0 < \sum_{i=1}^{k} u_{i,j} < N \quad \forall i$

The two conditions ensure that the sum of the membership degrees of every object equal to 1, i.e., the object completely contained in the clustering, and that each cluster has at least one object with a membership greater than 0, i.e., all clusters are non-empty.

Hierarchical Clustering is the task of constructing a tree-like nested structure partition of X, $H = \{H_1, \dots, H_Q\} (Q \leq N)$, such that $C_i \in H_m, C_j \in H_l$, and $m > l$ imply $C_i \in C_j$ or $C_i \cap C_j = \emptyset$ for all $i, j \neq i, m, l = 1, \dots, Q$.

This manuscript mainly focuses on the partitional clustering. Nevertheless, parts of the discussion and presented methods can also be applied to other forms of clustering as well. Referring back to the major problem of the lack of a definition of a *good* clustering: These definitions only state what can technically be considered a clustering but they do not impose any requirements on the quality of the clustering. For instance, putting all objects in one single cluster is a perfectly valid clustering according to the definition but yields no insight. Judging the quality of a clustering is a non-trivial task and is highly situation dependent. As mentioned before, clustering is very widely used and thus there exist various different definitions rendering it impossible to agree upon one definition of a good clustering fulfilling all demands of all scientific areas [15, 16]. As Xu *et al.* [15] put it, there exists no universally agreed-upon and precise definition of the term cluster, partially due to the inherent subjectivity of clustering, which precludes an absolute judgment as to the relative efficacy of all clustering techniques.

The researcher has to find a method which suites the properties and the expected shapes of the clusters of the problem best, which requires a rather deep knowledge of the different clustering tools and the domain of the dataset. The determination of the best cluster criteria can be regarded as one of the most challenging questions when performing a cluster analysis. Despite considerable effort, there is no overall best-performer excelling in all possible application scenarios, not even when limiting to a particular dataset type; the quality of the result is always highly dependent on the actual dataset [17].

The reason for the lack of an overall best-performing clustering tool becomes apparent when looking at the problem from a different point of view: When trying to sort n objects into m groups, this totals to

$$\frac{1}{m!}\sum_{k=0}^{k-m}(-1)^{m-k}\binom{m}{k}k^n$$

different possibilities. Concretely, the number of possibilities grows to $2.4 \cdot 10^{15}$ for only 25 objects separated into 5 groups [18]. This renders it absolutely infeasible to exhaustively test all possible combinations in order to determine the ideal clustering. Consequently, each clustering tool can only be an approximation trying to optimize the tool's inherent fitness function (i.e., inherent definition of a good clustering) which reflects the developers' ideas of an ideal clustering. This idea might be appropriate for some applications but can be very misleading for others.

Nevertheless, most definitions of a cluster criteria seek to optimize a combination of *separation* and *compactness* [3] reflected in the internal fitness function of the clustering approach. Furthermore, even when considering the same dataset, there is not necessarily one unique best solution: for example, clustering proteins either into homologs, families or super families are all relevant tasks; nevertheless, the clusterings will be hugely different from each other despite being based on the same dataset. This is directly reflected in another challenge of clustering: Most tools have at least one parameter defining the number and size (in more or less direct ways) of the clusters. Finding the optimal parameter setting is not trivial and is highly interconnected with the cluster quality analysis. Thus, clustering might be a series of trial-and-error steps before an acceptable result can be achieved. Considering the recent explosion of the amount and wealth of available biomedical data, the problems amplify: the datasets get more and more diverse, larger and more complex. Considering the runtime of a clustering tool on a large-scale dataset it becomes apparent that a meaningful and thoughtful decision on all involved steps of a cluster analysis is crucial in order to reduce unnecessary trial-and-error runs.

2 Performing a Cluster Analysis

According to Jain *et al.* [19], a cluster analysis consists of the steps depicted in Figure 1: (1) preprocessing (pattern representation), (2) definition of a pattern proximity measure appropriate

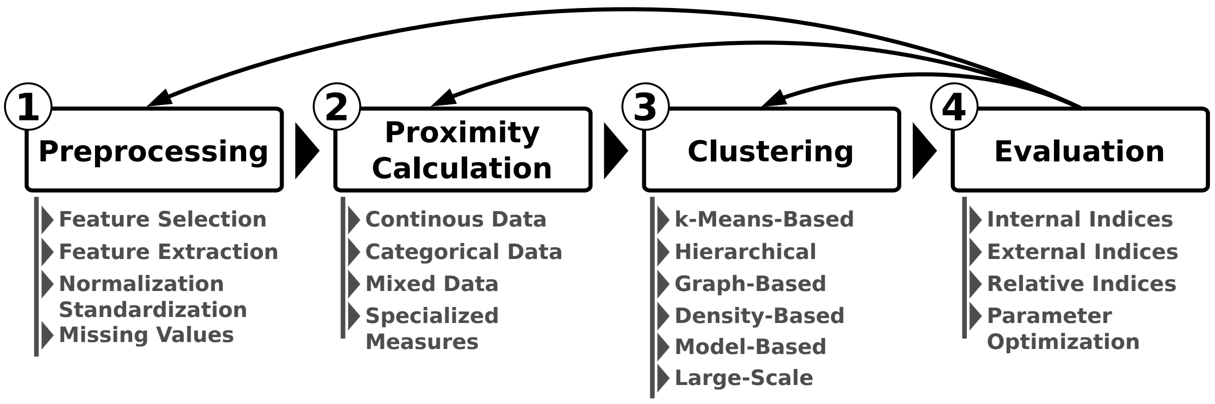

Figure 1: Overview of a typical cluster analysis. Each of the individual steps consist of several sub-steps which will be discussed in the according sections. Please note, that all steps are highly interconnected and rely on each other.

to the data domain, (3) clustering, and (4) assessment of the output. Note, in contrast to Jain *et al.*, the "data abstraction" step was excluded as it will not be discussed here. This is only a very broad overview of a cluster analysis and each of the steps consist of several sub-steps (refer to Figure 1). All steps are highly interconnected, meaning a suboptimal decision in the beginning may have severe effects on the overall quality in the end. Given the overview, the researcher has to answer the following questions:

1. What are the relevant features?
2. Should the features be normalized?
3. What is the most appropriate proximity measure?
4. What is the most appropriate clustering tool?
5. How should the parameters of the clustering tool be set?
6. How can the result be evaluated?
7. How can this be done efficiently on massive datasets?

3 Preprocessing

The beginning of every cluster analysis is the preparation of the dataset. Data can generally be present in two different forms [3]:

One-mode data comprises of a $n \times n$ data matrix describing the inter-object relationship, e.g., by a distance matrix or a similarity matrix. The name "one-mode" refers to the fact that columns and rows in the proximity matrix indexing the same thing (i.e., objects) [3].

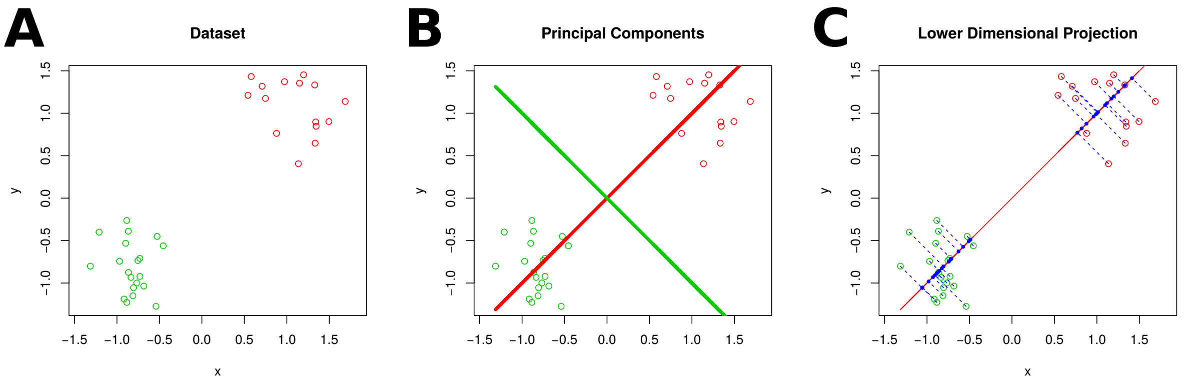

Figure 2: (A) clustering of a toy dataset of 35 objects with two distinct. The colors of the objects represent the clusters. (B) depicts the principal components of this dataset. The first principal component (red line) captures the most variance in the dataset. (C) displays the 1-dimensional projection of the dataset onto the first principal component (indicated in blue).

Two-mode data describes a $n \times p$ data matrix relating n objects to p *features* (also called *variables*). Depending on the domain, this format is also called raw-data. Many clustering tools require an one-mode input, thus the main challenge is to derive the inter-object relationships given the two-mode matrix by means of a suitable proximity function (refer to chapter 4). All clustering tools require a method of relating the object to each other; tools directly operating on two-mode data have an internal mechanism for achieving that.

When presented with two-mode data, the features may be of different data types which are distinguished as follows [3, 19, 20]:

Quantitative features which can be subdivided into:

1. continuous values (e.g., fold changes of gene expressions);
2. discrete values (e.g., number of genes);
3. interval values (e.g., timespan of a treatment, 1-2 days, 3-4 days, . . .).

Qualitative features which can be subdivided into:

1. nominal or unordered (e.g., eye-color);
2. ordinal (e.g., qualitative evaluations of pain with ”no pain”, ”some pain”, and ”strong pain”).

Structural data are comprised of repeated measurements of the same variable under different conditions, e.g., time-series data of gene expression.

Having a dataset of any type (or a mixture of several feature types), one crucial step might be extracting the most informative features for clustering. Here, researchers are often tempted to follow the approach along the lines ”more information is better” raising the question why selecting only a subset of features should be beneficial. This does not necessarily hold true for a clustering analysis: imagine a feature which is uniformly at random distributed over the entire dataset. Such a feature is entirely useless for identifying any groups in the dataset, to the contrary, it might even blur the existing different clusters into each other. Furthermore, features can also be highly correlated and thus not yielding additional information but might bias the proximity calculation. Generally, we distinguish between the following methods of deriving a suitable set of features:

Feature Selection describes the task of selecting the most informative features for the clustering task. The selected features are not modified. Generally, the goal of feature selection is the removal of irrelevant, redundant, or noisy features [21], which serves several objectives: (1) to improve the cluster performance, (2) to reduce the dataset size, (3) to learn about the importance of the features, or (4) a combination of them [22]. Classically, feature selection is utilized for supervised learning (e.g., classification) [23] but it plays an increasingly important role in unsupervised learning as well. Furthermore, when considering the growing dataset sizes, the reduction of the features might be crucial to reduce the computational time. It is differentiated between three general approaches of feature selection: (1) *filter* models, (2) *wrapper* models and (3) combinations of both [21, 24, 25]. Filter models decide on the importance of the selected features solely based on the features themselves. Wrapper models decide on the importance by evaluating the result of a coupled mining method, e.g. the classification performance. Hybrid models aim to combine both approaches [21]. The filter methods are commonly preferred for clustering because judging the quality of a clustering is a difficult problem in itself and thus unsuited to judge the feature selection quality (refer to section 6 for cluster evaluation methods). A general problem of all methods is that the researcher has to find a balance between feature reduction while maintaining the structural information in the dataset. Especially when using filter methods (which do not provide a feedback by means of a cluster validity index, for instance) determining the optimal feature number might be challenging. Monitoring the variance of the dataset provides an indication of how much information is lost during the feature selection process. A preserved variance after the feature selection process indicates that only correlated features got removed and thus the inherent structure of the dataset was maintained. A good overview of the existing models can be found in the work of Liu *et al.* [21], Guyon *et al.* [22], Dash *et al.* [22] and Alelyani *et al.* [25].

Feature Extraction describes methods which, in contrast to feature selection, create entirely new features based on the original features. The goals of feature extraction are similar to the goals of feature selection, i.e., the reduction of the feature space to only the most relevant features. The so-called principal component analysis (PCA) [26] is probably the most frequently used feature extraction method. The central idea of a PCA is to transform the dataset to a reduced set of *new uncorrelated* features which retain most of the variation of the original dataset [26]. Figure [**?**] depicts an example PCA. Another common technique is the multidimensional scaling (MDS) [27] which attempts an embedding of the objects into a lower-dimensional feature space such that the pairwise distances between the objects remain (as closely as possible) identical to the original feature space [28]. The MDS can also be applied to one-mode datasets (e.g., a distance matrix) when Euclidean coordinates are required for the clustering tool. Both methods are also commonly utilized for data visualization by reducing the feature space to two or three dimensions. For both the classical PCA as well as the classical MDS multiple extensions have been defined, e.g., covering non-linear relationships of the features, which are beyond the scope of this paper [29, 30, 31].

Generally, even though feature selection as well as feature extraction follow similar goals they have different characteristics. The "advantage of the former is greater interpretability, whereas the advantage of the latter is that a lesser number of transformed directions is required for the representation process" [32]. The loss of interpretability of the PCA originates from the fact that the principal components are linear combinations of the original features and thus the importance of the different original features becomes less apparent compared to the feature selection. A plethora of additional feature manipulation techniques exists, even methods expanding the number of features, the so-called feature expansion [33]. These methods create combined features, e.g., all pair-wise products of two features, which can improve the clustering performance by unraveling higher-order dependencies.

Another important preprocessing step is the normalization or standardization. Even if the dataset consists only of features of the same type (e.g., continuous variables), they might be on very different scales. The different scales might have a strong influence on the similarity function applied afterwards. For example, when using the Euclidean distance, a variable ranging between $[0, 1]$ will be almost entirely disregarded when a second feature ranges in the millions [34]. Therefore, it might be crucial to *normalize/standardize* the variables before usage. The most common methods are [34, 35]:

Min-max normalization is a linear transformation of a feature to a predefined value range, normally between $[0, 1]$:

$$f' = \frac{f - \min_F}{\max_F - \min_F}$$

where $\min_F$ and $\max_F$ denotes the global minimum/maximum of the feature F in the dataset and f represents some value of F.

Autoscaling or Z-score scaling, uses the standard deviation σ_F of the values of F and the mean $\overline{F}$ to scale the variable:

$$f' = \frac{f - \overline{F}}{\sigma_F}.$$

This method is of particular interest in cases where the extreme values (i.e., $\min_F, \max_F$) cannot be determined or in cases where few extreme outliers would bias the min-max normalization [34]. The transformed variable will have a mean of 0 and a standard deviation of 1 [36]. There exists a similar measure which abstains from centering the variable (i.e. $v' = \frac{v}{\sigma_F}$.) which preserves the location information of the variables [36].

Decimal scaling describes the shifting of the radix point such that the largest value in the dataset is smaller than 1:

$$f' = \frac{f}{10^j}$$

with j being the smallest integer such that $\frac{\max_F}{10^j} < 1$.

Further standardization methods can be found in Jajuga *et al.* [37]. The influence of the data normalization is widely researched for some special cases, e.g., the normalization of microarray data [38]. It is important to note that through normalization the knowledge of the location and scale of the original data might be lost and thus should only be used in case the applied distance function is sensitive to scales [36].

Further, a dataset might have missing values, either due to technical limitations, noise or error. Nevertheless, many similarity functions cannot handle missing values and simply removing all incomplete objects or features from the dataset might not be an acceptable solution. More sophisticated approaches for treating missing values can be categorized as follows [39, 36]:

Prereplacing is part of the preprocessing; with these methods the missing values get replaced before the actual clustering. The most common methods try to reconstruct the missing values by for example, the mean-and-mode method, nearest neighbor estimators or by means of a linear regression. Furthermore, for some applications specialized methods exist, for instance for reconstructing missing values in microarray data [40, 41]. Regardless of the employed method, the researcher must be aware that these methods cannot reconstruct the actual values and thus necessarily introduce errors which can lead to misleading and wrong clustering results when used without care.

Embedded methods describe attempts to handle missing values during the clustering process itself, i.e., the clustering tool allows for missing values in the dataset. Most attempts for handling missing values have been undertaken with k-means like variants [42, 43, 44] but there are also other methods like the graph-based Transitivity Clustering [45] which can handle missing values.

4 Proximity Measures

When handling two-mode datasets, one central aspect of the cluster analysis is to establish a relationship between the objects. This is achieved by so-called proximity functions, i.e., either a distance function or a similarity function. Similarity functions are maximal the more similar two objects are, the distance functions are minimal the more similar two objects are. The choice of the suitable proximity function is highly depended on the data types of the two-mode matrix:

Continuous Data is probably the most common case and does thus require the most attention. The most common approaches for dealing with continuous data, are

- *Euclidean distance:* This is the most common distance measure when presented with continuous variables which corresponds to the distance of two points when measured with a ruler. Let $u = \{u_1, \ldots, u_p\}, v = \{v_1, \ldots, v_p\} \in X$ be two objects with p features, then the Euclidean distance is calculated as

$$d(u, v) = \sqrt{\sum_{i=1}^{p} (u_i - v_i)^2}$$

- *Minkowski Distance:* The Minkowski distance is the generalization of the Euclidean distance:

$$d(u,v) = \left(\sum_{i=1}^{p} |u_i - v_i|^l \right)^{1/l}$$

  with the parameter l. For $l = 2$, the Minkowski distance corresponds to the Euclidean distance. Another noteworthy special case is $l = 1$, the so-called Manhattan distance [46]. This distance corresponds to the accumulative absolute difference in each feature between two objects. Generally, if the dataset is supposed to have compact and/or isolated clusters, the Minkowski distances have proven to work well [47, 36]. Nevertheless, in higher dimensional datasets, the Minkowski distances have some unwanted properties for clustering. Aggarwal *et al.* [48] write that ”under certain reasonable assumptions on the data distribution, the ratio of the distances of the nearest and farthest neighbors to a given target in high dimensional space is almost 1 for a wide variety of data distributions and distance functions”. When the furthest and closest neighbor have essentially the same distance, a meaningful application of most clustering tools is no longer possible.

- *Correlation:* The Minkowski distances measure the distances in terms of the absolute difference in the values of the features but neglect the linear relationship between the feature. In cases, where this linear relationship between the feature vectors is crucial in contrast to the magnitude of the values, correlation based methods should be used. In other words, when considering the features of an object as a vector, the similarity is established solely by the direction of the vectors and not their length. A primary example for such data are gene expression datasets [49]. The Pearson correlation coefficient is defined as

$$\Phi(u,v) = \frac{\sum_{i=1}^{p}(u_i - \overline{u})(v_i - \overline{v})}{\sqrt{\sum_{i=1}^{p}(u_i - \overline{u})^2 \cdot \sum_{i=1}^{p}(v_i - \overline{v})^2}}$$

  The values of the Pearson correlation range between $[-1, 1]$ and are often transferred to a distance ranging between $[0, 1]$: $d(u,v) = {}^{(1-\Phi(u,v))}/_{2}$. The Pearson correlation is highest when the two variables are perfectly linearly correlated. The Spearman rank correlation replaces the actual values of the features with their rank within the feature vector and thus does not require a linear relationship but a monotonic relationship for a perfect score.

Categorical Data is less common in technical measurements but appear, for instance, very often in clinical patient data.

- *Boolean Variables:* This special case of a categorical variable describes the presence or absence of a certain feature. When comparing two objects u and v, there are in total four possible outcomes:

		Object v	
	Feature	1	0
Object u	1	**a**	**b**
	0	**c**	**d**

When comparing two objects with several features, a counts the number of positive matches, d the number of negative matches and b and c the number of mismatches, as illustrated in the table above. Most measures define the similarity between two objects by relating the number of matching features (i.e., a and d) to the number of mismatching features (i.e., b and c). While b and c are equivalent, a and d are not: Depending on how prevalent the presence of a given feature is, the common absence of that feature might only yield little information compared to the common presence. Whereas when the absence and the presence of the feature are equally common, a and d have the same explanatory power. Most common measures for the boolean variables differ in the way they weight a and d [3]:

$$s(u,v) = \frac{a+d}{a+\lambda(b+c)+d}$$

Are common measures for cases d yielding the same explanatory power than a. For $\lambda = 1$, this corresponds to the matching coefficient, for $\lambda = 2$ to the coefficient by Rogers and Tanimoto [50] and for $\lambda = 1/2$ to the coefficient by Gower and Legendre [51]. In cases the common absence of the feature should not positively influence the similarity, most measures have the following form:

$$s(u,v) = \frac{a}{a+\lambda(b+c)}$$

For $\lambda = 1$ this corresponds to the Jaccard coefficient [52], for $\lambda = 2$ to the coefficient by Sneath and Sokal [53], and for $\lambda = 1/2$ to the coefficient by Gower and Legendre [51].

- *General categorical variables:* When dealing with categorical variables with more than two levels (the possible values a variable can assume), different measures need to be employed. One straight forward approach is the dichotomization of the categorical value, i.e., creating for every single categorical level a own variable indicating whether this particular level is present or not. This allows for the usage of the aforementioned coefficients, but will consequently lead to a multitude of negative matches [3]. A different approach is deriving a score by counting the categorical features the two objects agree upon:

$$s(u,v) = \sum_{i=1}^{p} \delta_k(u,v)$$

with $\delta_k(u,v)$ being an indicator function whether u and v are in the same category of feature k. Additionally, a weighting factor w_k for feature k may be introduced in order to reflect the importance of the different features or to allow for missing values when set to 0. Goodall *et al.* have proposed a measure which seeks to normalize the similarity between objects by the probability of observing a certain similarity by chance [54]. A comprehensive overview can be found in the review of Boriah *et al.* [55] and the book of Gan *et al.* [36].

Mixed Data Types describe a mixture of different data types, e.g., a mixture of continuous and categorical features. The most common approach is the usage of a generalized distance coefficient as introduced by Gower [56]:

$$d(u,v) = \frac{\sum_{i=1}^{p} w_k(u,v) s_k(u,v)}{\sum_{i=1}^{p} w_k(u,v)}$$

Here, $s_k(u,v)$ defines the similarity between u and v for feature k using a suitable similarity function with respect to the type of feature k. The indicator function $w_k(u,v)$ is normally either set to 0 or 1 depending on whether the comparison of u and v is valid in feature k and thus allows for handling missing features.

Specialized Measures describe similarity functions which are designed for a particular data type or use case. In the field of Bioinformatics, specialized functions calculating sequence similarities have probably seen the most attention. In a naive fashion, each position of a sequence could be regarded as a categorical variable with the four nucleotides as possible values and a general measure for categorical variables could be applied. Nevertheless, this approach neglects the biological background of how sequences have evolved; more suitable comparison measures for sequences have been developed as early as in 1969 [57]. Nowadays, the research community has a multitude of different measures for comparing nucleotide or protein sequences, probably most prominently NCBI BLAST [58]. For protein comparison, not only the sequence could be used but for example also structural comparisons when available. Tools like MAMMOTH (Matching molecular models obtained from theory) seek to align two structures and the goodness of the alignment can be used as similarity measure [59]. All these measures have in common that they seek to incorporate additional domain knowledge (e.g., evolutionary history of proteins) for a particular type of datasets.

The proximity measures discussed above comprise of both, similarity measures $s(u,v)$ and dissimilarity measures $d(u,v)$. Depending on the employed clustering tool, a similarity might be required to be transferred into a distance or vise-versa. This can be achieved by using a monotone-decreasing function [60]. Some commonly applied transformation functions, assuming the range of the proximity function is within $[0,1]$, are: $d(u,v) = \frac{1}{1+s(u,v)}$, $d(u,v) = 1 - s(u,v)$, or $d(u,v) = -log(s(u,v))$.

The choice of the proximity function is crucial for the resulting clustering quality. There is no general best proximity function for all cases; the researcher has to be completely aware of the properties of the utilized proximity function and the consequences to the data in question. The lack of a best proximity function becomes clear when considering a perfect proximity function: such a function would already give an answer whether the two objects belong in the same cluster or not and thus would optimally solve the clustering problem. This should emphasize the importance of the proximity function during the clustering process.

The successful calculation of the proximities forms the input of the clustering tools which will be discussed in the next section.

5 Tool Selection

Selecting the best-suited tool or algorithm for a clustering task is an unsolved problem and only little guidance exist. Even in a large-scale performance comparison study, it is not entirely possible to determine the best tool, not even only for a specific type of datasets [17]. The focus of this manuscript is not to compile a comprehensive list of available clustering tools. Many studies already discuss a multitude of tools; for example Jain *et al.* [19, 61], Xu *et al.* [14], or Milligan *et al.* [62] are discussing clustering tools in general. Reviews with a particular biomedical focus are for example Jiang *et al.* [63], Andreopoulos *et al.* [64], or Xu *et al.* [15]. The focus of this manuscript is rather providing a general overview of the different clustering strategies and further emphasizing different important traits of the clustering tools. For each clustering strategy, examples for their application in the biomedical context are provided in order to help the researcher to make an informed decision for a tool.

5.1 Clustering Strategies

Generally, the strategy of a clustering tool can be categorized as follows [64, 15]:

k**-Means based** algorithms (sometimes also called Squared Error-Based Clustering) describe algorithms which seek to identify an optimal k clustering. This is achieved by iteratively updating cluster centers and object memberships optimizing a given cluster criteria (for instance the sum-of-squared-error criteria). k-means is probably the most widely used clustering algorithm and performs reasonably well on certain problems [65]. Nevertheless, k-means also has serious drawbacks as it, for example, prefers clusters which are compact and spherical [15]. Further, the number of clusters needs to be known and the algorithm can be caught in local optima during the optimization process and thus produces suboptimal clusterings. The later problem can generally be tackled by repeated runs with different random initializations or by more sophisticated strategies for initialization; a review of those methods can be found in Celebi *et al.* [65] or Pena *et al.* [66].

Multiple extensions and improvements have been developed for k-means, rendering k-means the basis for an entire family of clustering tools. Just to mention a few, partitioning around medoids (PAM) enables the application of k-means in presence of a one-mode proximity matrix in contrast to actual raw-data which could be interpreted as points in an Euclidean space [67]. PAM does not compute a cluster-center but selects the object $u \in X$ of the dataset which is closest to all members of the cluster as center. Another extension is the development of fuzzy variants, e.g., fuzzy c-means [68] which already has proven useful in biomedical contexts [69]. Further, variants of k-means exist which are particularly suited for handling large-scale datasets, like CLARA (Clustering Large Applications) [70]. CLARA performs a PAM like clustering on a subset of the original dataset and thus can run also on large datasets.

Hierarchical clustering methods create an entire nested structure of the clusterings. In order to receive a partitional clustering, the retrieved tree-structure is cut at a certain level in order to produce k clusters, or more sophisticated tree-cutting approaches cutting at different levels can be applied [71]. Generally, hierarchical clustering tools are differentiated into *agglomerative* and *divisive* methods; the former start with all objects being singletons (i.e., clusters containing only one object) and consecutively join clusters together, the later start with one cluster and subsequently divide the large clusters into smaller ones. The agglomerative methods are computationally more efficient but the decision of joining clusters can not be reversed; thus, the cluster solution can be far from an optimal clustering due to accumulated small local errors. In contrast, the divisive clusterings first generate the large clusters and thus are less prone to accumulate small local mistakes; on the other hand, dividing large clusters into optimal smaller clusters is computationally hard and can only be tackled by heuristics. This is the reason why most commonly agglomerative methods are used. The main criteria of agglomerative clustering methods is the so-called linking function which decides what clusters should be joined. The most popular ones are

- *Single linkage:* The distance between two clusters is defined as the distance between the two closest objects between both clusters. This method tends to create unevenly shaped, elongated clusters due to an effect called *chaining* [3].
- *Complete linkage:* This distance function can be regarded as the opposite of single linkage clustering. The distance between two clusters is defined as the distance of the two points furthest apart. This method does not suffer of the chaining but tends to create clusters with equal diameter also in inappropriate situations.
- *Average linkage:* Average linkage seeks to combine the advantages of the both previous linkage functions and defines the distance between two clusters as the average distance between all pairs of objects of both clusters.

Hierarchical clustering is a very common clustering technique and has been applied in a multitude of different biomedical studies, from multiple sequence alignment [72], protein-protein interaction networks [73], protein evolution [74, 75, 76] to gene expression analysis [4, 49, 77] to name just a few.

Graph-based clustering algorithms represent the input data internally as a graph, with the objects corresponding to the nodes and the (normally weighted) edges to the similarities between the objects. Most algorithms then seek to identify densely connected subgraphs by incorporating the neighborhood of the objects in the graph or by performing random walks on the given graphs. Most of the algorithms do not require the desired number of clusters k as parameter but are tuned by more indirect means. The probably most prominent examples are Affinity Propagation [78], clusterONE [79], Markov Clustering [80], Spectral Clustering [81], and Transitivity Clustering [82]. Graph-based algorithms have been widely applied to biomedical datasets, particularly in the context of biological network and complex analysis [83, 79, 84, 85, 86, 87] but also for protein homology detection [13, 88].

Density-based clustering approaches seek to identify arbitrarily shaped clusters by separating high-density areas from low-density areas. Normally, density-based cluster algorithms require only one scan of the dataset, are insensitive to noise [36] and can also detect outliers. Density-based clustering methods can be very efficient in terms of memory consumption and computational time and are thus often used for large-scale datasets [64]. The probably most prominent examples are density-based spatial clustering of applications with noise (DBSCAN) [89] and density clustering (DENCLUE) [90]. The primary biomedical application area of these clustering tools is the identification of dense subspaces in interactome data [64].

Model-based clustering methods assume that the given dataset was generated by an underlying probabilistic model. The aim is to maximize the model in such a way that it best describes the observed data. One advantage of model-based clustering tools is that they allow for the integration of background distributions which is especially in a biomedical context of great importance. The most well-known representatives are Hidden-Markov-Model-based clustering [91], Self-Organizing Maps [92] or Finite Mixture Models [93]. Typical application areas are gene expression analysis [94, 95, 96] and sequence analysis [97].

Large-scale clustering methods might belong to any of the above groups but were particularly developed for coping with large datasets, often running on distributed computer systems. Especially in the era of big data, coping with large-scale datasets becomes a crucial element. Normally, these approaches achieve their performance by either random sampling, data condensation, divide and conquer, incremental learning, density-based approaches, grid-based approaches, or by a combination thereof [15]. This might result in a less accurate clustering but in cases were other methods fail, those methods might be a beneficial solution. Noteworthy examples of this category are the hierarchical approach BIRCH (Balanced Iterative Reducing and Clustering using Hierarchies) [98], CURE (Clustering Using Representatives) [99] or the aforementioned tools CLARA [70], DBSCAN [89], and DenCLUE [90].

Other strategies comprises of further methods which a less relevant for this manuscript as the focus is on the most popular clustering approaches applied in the biomedical context. Refer to the aforementioned reviews for an overview of additional approaches, e.g., grid-based clustering, neural network-based clustering, evolutionary algorithms-based Clustering and several more.

5.2 Clustering Properties

Solely based on the general strategy a clustering tool follows, an assumption whether a tool is suitable or not can not be made. Given the general compatibility of the clustering tool with the dataset, the researcher should consider at least the following properties of the clustering tools and their relevance for the problem [64, 15]:

Scalability describes the behavior of the algorithm with increasing dataset size in terms of runtime and memory consumptions. For large-scale examples, algorithms possessing a quadratic runtime might already be regarded inappropriate [15].

Dimensionality is the ability of the algorithm to handle the number of present features in the dataset. Especially very high dimensional datasets like gene expression data, where the number of dimensions supersedes the number of samples, might challenge some tools. For instance, density based clustering tools might be unable to discover any significant change of densities in such datasets.

Robustness describes the ability of an algorithm to cope with noise in the data. An algorithm which reacts to small variations in the input with significant changes in the output is not well-suited for the normally noisy biological datasets. K-means and some variants of hierarchical clustering are tools which tend to react erratically in the presence of noise [64].

Number of parameters and sensitivity describes the method's reliance on the user-input in tuning the algorithm. Here, several aspects are important. (1) The number of parameters should be as small as possible in order to avoid over-training of the algorithm and to limit the effort of the parameter training. (2) It should be possible to anticipate the effect of changing the parameters and not show near random effects on the clustering, i.e., small changes in the parameter should not lead to tremendous changes in the result. (3) The parameters should be interpretable to the user. Some clustering algorithms have five or more parameters greatly influencing the clustering and only tune hard-to-understand details of the algorithm. For a non-expert user, it is difficult to understand and use such tools properly.

Arbitrary cluster shapes refers to the ability of the algorithm identifying arbitrarily shaped clusters. For instance, k-means produces spherical shaped clusters. This behavior is not necessarily bad but the researcher should be well-aware of this tendency and expect this particular shape of clusters in the dataset. If there is no such prior knowledge about the cluster shape, tools allowing arbitrarily shaped clusters should be employed.

Usability and availability of the clustering tool is an important, yet widely neglected criteria for a clustering tool. Considering that most analyses are neither carried out by clustering experts nor computer experts, a pure command-line tool might lead to maloperation and ultimately to bad clustering results. This is also strongly connected to the interpretability of the tools' parameters as discussed above. Furthermore, the tools should be freely available as open-source in order to ensure the results of a study are generally reproducible and there is full disclosure of the applied method.

6 Cluster Evaluation

One of the most critical steps in a cluster analysis is the actual evaluation of the clustering result. The inherent problem of this endeavor is that normally a gold standard, i.e., the ground truth, is missing. Clustering results are normally evaluated by means of so-called cluster validity indices (or measures). Generally, three different types of validity indices are distinguished [100]:

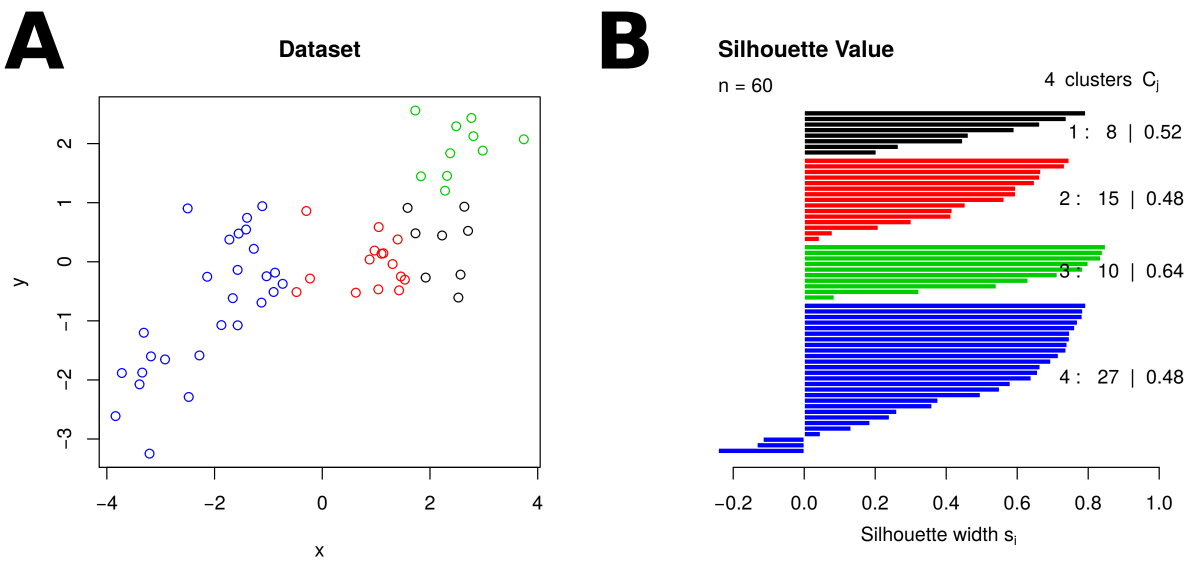

Figure 3: (A) clustering of a toy dataset with 60 objects. The colors of the objects represent the clusters. (B) depicts a typical silhouette plot of the clustering. The objects of each cluster get sorted by their silhouette value and plotted vertically. Negative silhouette values indicate that the object is on average closer to the points of the nearest other cluster than to the objects in its own cluster.

External validity indices compare a clustering result against a gold standard. When a gold standard K is present, one can establish the similarity between the clustering C and K. One approach is considering all pairs of objects u, v and determine whether they are clustered together in C and K. Comparable to the treatment of binary features, let a define the number of pairs being clustered together in C and K, d the number of pairs which clustered apart in C and in K, b the number of pairs which are clustered together in C but not in K and c the number of pairs which are clustered apart in C but are clustered together in K. The most common measure based on this definition are:

- *Rand index* [101]:
$$R(C, K) = \frac{a + d}{a + b + c + d}$$
- *Jaccard index* [52]:
$$J(C, K) = \frac{a}{a + b + c}$$

The Jaccard index can be seen as a stricter definition of the Rand index because it only counts positive agreements [102]. The main criticism on these indices, particularly on the Rand index, is their rather small range of actually achieved values and their tendency to generally report higher values with an increasing number of clusters [103, 3]. The most notable extensions to the Rand index is the so-called adjusted Rand index which seeks to counter the aforementioned short-comings by statistically correcting for chance [104].

Another way of comparing a clustering to a gold standard is by mapping every cluster $C_i \in C$ to the cluster $K_j \in K$ with the greatest overlap. Assuming that the cluster C_i has the largest overlap with K_j, each element $u \in C_i \cup K_j$ can be defined as true-positive (TP) if $u \in C_i$ and also $u \in K_i$, as false-positive (FP) if $u \in C_i$ but not in K_i and as false-negative (FN) if $u \in K_j$ but not in C_i. There is no meaningful definition of true-negatives in this scenario. Based on these findings, we can define different validity indices:

- *Precision*
$$\mathrm{Prec}(C, K) = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FP}}$$
- *Recall*
$$\mathrm{Rec}(C, K) = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FN}}$$
Precision as well as recall have the problem that they could easily be optimized by producing a trivial solution; a clustering with only singletons would give a perfect precision due to the lack of FPs; a clustering with one single cluster containing all objects would result in an optimal recall due to the lack of FNs.
- *F-measure* is probably the most commonly used external validity index and corrects the obvious shortcomings of precision and recall as it is defined as the harmonic mean of both:
$$F_\beta(C, K) = \frac{(1 + \beta^2) \cdot \mathrm{Prec}(C, K) \cdot \mathrm{Rec}(C, K)}{(\beta^2 \cdot \mathrm{Prec}(C, K)) + \mathrm{Rec}(C, K)}$$
The parameter β is used to influence the weight of the precision or recall. In most cases, $\beta = 1$ (in that case, it is also often called the F1-measure) meaning both precision and recall have the same importance.

Internal validity indices are employed when a gold standard is absent. In such case, the clustering can only be evaluated with the data available, i.e., the clustering result and the unlabeled input. The problem of defining a good validity index is analogous to defining a good cluster criteria and thus suffers the same shortcomings of being rather subjective and problem dependent. Most internal measures seek to reward *compactness* and *separation* of the clusters.

- *Dunn index* relates the maximal cluster diameter $\max_\emptyset$ to the minimal distance between a pair of clusters $\min_d$ [105, 106]:
$$D(C) = \frac{\min_d}{\max_\emptyset}$$
One obvious disadvantage of this index is the sensitivity to outliers as they might tremendously influence the maximal diameter or minimal distance.
- *Davis Bouldin index* can be seen as a relaxed version of the Dunn index as it relates the average distance of objects in a cluster to the cluster center to the distance between cluster centers [107]. Let $\overline{C_i}$ be the center of cluster C_i and $\sigma_i = \sqrt{1/|C_i| \sum_{u \in C_i} |u - \overline{C_i}|}$ and $d(C_i, C_j)$ the Euclidean distance between two cluster centers, then the Davis Bouldin index is defined as:
$$DB(C) = \frac{1}{n} \max_{C_i \neq C_j \in C} \left(\frac{\sigma_i \cdot \sigma_j}{d(|\overline{C_i}|, |\overline{C_j}|)} \right)$$

- *Silhouette value* is calculated for each object $u \in C_i$ individually. It relates the average distance of the object to all objects within the cluster to the average distance to all objects in the closest foreign cluster. Let $a(u)$ denote the average distance of u within its cluster and $b(u)$ the minimal average distance to all objects of a different cluster, then the Silhouette value is defined as

  $$S(u) = \frac{b(u) - a(u)}{\max\{a(u), b(u)\}}$$

  and for the entire dataset as

  $$S(C) = \frac{1}{n} \sum_{u \in C} S(u)$$

  The Silhouette value can assume values between $[-1, 1]$ with negative values indicating that a point is on average closer to a foreign cluster than to its own cluster. In a large-scale study evaluating several popular clustering tools it has been shown that the Silhouette value corresponds well to external indices for real-world datasets but has problems with intertwined clusterings [17]. Figure 3 depicts a typical silhouette plot.

A good review of many more internal cluster validity indices can be found in the work of Liu *et al.* [108].

Relative validity indices compare several clusterings from the same algorithm, e.g., when run with different parameters, or different algorithms with each other. This is a common method for parameter training or determining the number of clusters k in a dataset and will be discussed in the next section.

7 Parameter Training

In real-world problems a gold standard is not available and internal indices have to be employed. The problem is that these indices do not give a measure of the ”absolute quality” of the result, i.e., it is not known how far these indices are away from the ground truth. As all clustering tools have at least one parameter influencing the clustering result. A parameter optimization has to be performed in order to achieve the best possible result for the tool and the given dataset. One crucial component is the number of clusters k which serves as parameter for many clustering tools. Several strategies for optimizing parameters of a clustering tool exist:

Relative cluster validity compares the outcome of the same tool with different parameters to each other. Here, any internal validity index can be employed and optimized. The relative cluster validity can also be used to determine the number of clusters k. Here, Halkidi *et al.* differentiate between two cases [109, 110]:

- *k is not a tool parameter:* For clustering tools where k is not a direct parameter but rather influenced indirectly by means of other parameters, a common procedure is:

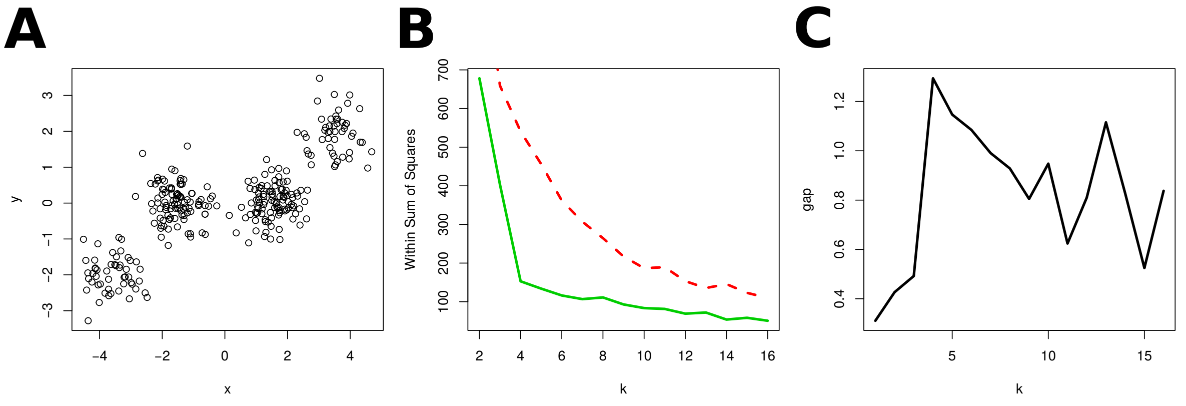

Figure 4: (A) a toy dataset consisting of 4 spherical clusters with 300 objects in total. (B) depicts the within sum of squares, i.e., the distance of the objects to their cluster centers for varying k. The green solid line depicts the value for the dataset, the red dashed line depicts the value for a background model without cluster structure for the gap statistic. The green line shows a clear elbow at the actual cluster size four. (C) depicts the gap statistic with the peak at 4 clusters.

All parameters are varied in an as wide as possible range. Then the widest range of the tool's parameters resulting in a constant number of clusters is selected and the optimal parameter is defined as the mean of this parameter range. A wide range of parameter values yielding a constant k is a good indication that this steadiness is actually due to the inherent cluster structure of the dataset.

- *k as parameter:* If the tool has the desired number of clusters k as a direct parameter the application of an internal validity index is inevitable. In these cases, k is varied in a wide range. Additionally, for each k tested, the remaining parameters of the tool are also tested in a as wide as possible range. For each of the clustering results, an internal validity index is evaluated and the parameter configuration yielding the best result is used for the final clustering. Nevertheless, there are validity indices which constantly increase (or decrease) with a growing number of clusters regardless of the actual clustering quality. For instance, the mean-squared-error (MSE) produces monotonically decreasing values with an increasing number of consequentially smaller cluster. In these cases the researcher can look for a so-called knee (or elbow) in the plot of the validity index, i.e., the point where after a rapid decrease the validity index's decrease levels off. Figure 4 depicts the elbow on a toy dataset. This concept was generalized and is explained below. A more exhaustive discussion can be found in the work of Halkidi *et al.* [110].

Statistical evaluation describes methods employing background models in order to statistically evaluate the clustering. A more generalized way of assessing the aforementioned elbow is the so-called gap statistic for determining the number of clusters [111]. Refer to Figure 4 for an example.The general principle is to create a background model so the goodness of the validity index can be objectively judged. This is done by creating random datasets with similar spread and principal components but with a lack of cluster structure. The cluster quality of the random dataset is compared to the quality of the actual dataset for a wide range of k. The k leading to the largest difference between the actual quality and randomized quality is suggested as the optimal parameter k.

Partial gold standard might help in determining clustering parameters. A partial gold standard of a dataset means that for a small subset of the dataset the actual labels are known. This fact can be exploited in two different ways: (1) the entire dataset can be clustered with a wide range of different parameters. For each clustering, an external measure is applied, only comparing the objects present in the dataset as well as in the gold standard. (2) If an extensive parameter testing is infeasible on the entire dataset, the parameter training can be performed on the reduced dataset containing only the objects also present in the gold standard. After deriving an optimal parameter set for the reduced dataset, the parameters might be directly applied or scaled to the size of the entire dataset. It is of fundamental importance that the effect of the parameter with respect to the dataset size is known. For example, when using k means, let k_o denote the optimal number of clusters for the small gold standard. Using k_o for the entire dataset might be entirely misleading, because this would assume that the entire dataset has the same number of clusters as the gold standard. Nevertheless, depending on the nature of the gold standard, a viable assumption might be that the average size of the clusters is an invariant and thus k_o can be scaled to fit the entire dataset. Parameters of other clustering tools are more independent of the dataset size. For example the threshold t of Transitivity Clustering defines the minimal average intra-cluster similarity. Assuming that the clusters in the entire dataset show a similar intra-cluster quality as the gold standard, the parameter t can directly be applied to the entire dataset [13].

Incorporating domain information can improve the cluster quality. In many cases, the researcher has specific domain information about the dataset which might be exploited for recovering the number of clusters or optimal parameters. The possibilities are manifold but can hardly be generalized. Typical traits of domain knowledge are for example objects which certainly belong in the same cluster or in a different cluster. A different example is that the objects to cluster originate from a known number of entities (e.g., patients or organisms) and thus clusters larger than the number of entities might indicate a too generous parameter setting. A comparable strategy was already employed for protein homology detection of actinobacteria: The clustering was optimized for maximizing the core-genome (i.e., clusters consisting of one protein of every organism) while minimizing the number of unrealistically large clusters [88].

Regardless of the type of additional domain information, it is important that the optimization procedure is independent of the actual research question. For example, when it is hypothesized that certain cancer genes cluster together, it would be highly biased to use the number of clustered cancer genes as optimization criteria.

The guidelines above give a basic understanding on how to evaluate the quality of a clustering. The evaluation is the first real point of feedback on the previously made decisions. It should be apparent, that all decision during a cluster analysis are highly interdependent. The data preprocessing influences the effectiveness of the similarity functions; the similarities have an impact on the effectiveness of the clustering tool. Only understanding the consequence of every single decision in the course of a cluster analysis enables the researcher to produce reliable and high-quality cluster results. Even when a thoughtful strategy is established, the evaluation of those results (especially with a lack of gold standard) remains a difficult task which is highly problem specific and requires a great amount of knowledge from the researcher. As Jain and Dubes put it in their book ”Algorithms for clustering data” [100]: ”The validation of clustering structures is the most difficult and frustrating part of cluster analysis. Without a strong effort in this direction, cluster analysis will remain a black art accessible only to those true believers who have experience and great courage.”

8 Discussion & Challenges in the Post Genomic Era

Clustering has been proven extremely useful in a wide variety of different applications. A high-quality cluster analysis can only be successfully conducted when all involved steps are carried-out thoughtfully and in synchronization with each other. Decisions on the data preprocessing influence the types and effectiveness of the similarity function which in turn influence the performance of the clustering tool. The complexity of a cluster analysis and early bad decisions are becoming an increasing nuisance since the dataset sizes will keep growing significantly; thus a simple trial-and-error approach will be too time-consuming. Furthermore, the biological datasets do not only grow in size but also in wealth and complexity posing fundamental problems to the researchers. In the vicinity of Big Data the growth of data is often described with the three Vs: Volume, Velocity and Variety, all describing different aspects of the challenges faced in the era of big data. A general problem of Big Data is the lack of a sharp definition as it is highly problem dependent and commonly a combination of available compute power, dataset size and complexity of the applied algorithms. As clustering in the biomedical context is often conducted by non-computer-experts with no or only limited access to large compute servers, we define datasets in the framework of this manuscript as Big Data when they are exceeding the capabilities of a normal desktop computer. With clustering being a typical first step in an entire analysis pipeline, it is one of the front lines facing this tremendous amount of data. If this step requires access to a compute server it will already exclude a significant number of researchers and thus limits the effective usage of the available data.

There are several very efficient clustering tools available, running in complexities of $O(n)$; the review of Xu *et al.* [15] already lists several of those tools: the aforementioned BIRCH [98], CLIQUE [112], fractal clustering (FC) [113] or WaveCluster [114]. All those tools require raw data, more specifically, data which can be interpreted in a n-dimensional numerical space. This is viable for many application scenarios but fails in others, many of them relevant for biomedical research.

In biomedical research, many available data types have no meaningful embedding in an n-dimensional space. For instance, protein sequences, DNA sequences, biological networks, structural protein information can all not be treated with the generalized methods mentioned above. Coupled with the recent advances in wet-lab technology, the amount of these datasets is also increasing drastically. One approach is the development of highly specialized clustering tools which serve exactly one purpose. One popular example is cd-hit [9], a fast tool for protein clustering. It is quite widely applied, for example, with the universal protein knowledgebase (Uniprot) [115] or with the protein database (PDB) [116].

Nevertheless, it is unrealistic for the researcher to develop a specific clustering tool for each dataset. In case the dataset cannot be embedded into a n-dimensional space or no specialized tool with integrated similarity calculation is available, all-purpose clustering tools handling a normal similarity/distance matrix have to be used. When resorting to standard clustering tools in those cases, a quite astonishing observation can be made: The actual bottleneck in terms of computational time is not the clustering itself but actually the calculation of the pairwise proximities [117, 118, 119, 45].

With respect to the pairwise proximity calculation, two points should be highlighted:

1. The calculation of all pairwise proximities is necessarily quadratic in runtime. Thus, even with a sub-quadratic clustering algorithm, the entire cluster analysis has a quadratic runtime, dominated by the proximity calculations.

2. Clustering tools benefit from more complex and better suited similarity functions. Especially in the biological context, it might be helpful to integrate, for instance, several data sources and domain knowledge to construct a superior proximity function. Advances in proximity functions might be even more critical to the clustering quality than an improved algorithm operating on a sub-optimal proximity function. For example, it has already been shown that protein clustering might be improved regardless of the utilized clustering when using Hidden Markov Models of the sequences instead of the mere sequence [120].

These points, together with the ever increasing volume and variety of the available datasets, will further shift the focus on developing more efficient and at the same time more complex proximity functions. That means incorporating a greater variety of data on a larger scale will pose one of the greatest challenges in biomedical cluster analysis. Thus, future research might concentrate on a more efficient use of the proximities. It has already been shown that many proximities are in fact redundant and are not strictly required for a good clustering [45]. For example, in the case a set of objects possesses very high pair-wise similarities, only a fraction of the similarities are required to make the decision to regard this set as a cluster; all additional similarities are in that sense redundant. Tools which are able to cope with missing values can be exploited such that not all pairwise proximities are calculated but only a fraction of them. Such an approach was already successfully tested with Transitivity Clustering [45]. Here, further developments exploiting the missing values strategically would allow for high cluster qualities with increasing dataset size while enabling more complex similarity functions.

To conclude, cluster analysis is a long standing problem in computer science and is applied in virtually any scientific area. Despite of the countless efforts published in this area, it remains a challenging task to perform a high-quality cluster analysis and should be carried out with greatest caution and expertise knowledge. Furthermore, the era of big data adds new exciting challenges to the picture which need to be accounted for by the research community in the future.

References

[1] M. L. Metzker. Sequencing technologiesthe next generation. *Nature reviews genetics*, 11(1):31–46, 2010.

[2] J. Naisbitt. Megatrends warner books. *New York*, pages 192–195, 1982.

[3] B. S. Everitt, S. Landau, M. Leese and D. Stahl. *Cluster Analysis*. Wiley Series in Probability and Statistics. John Wiley & Sons, Ltd, 5th edition edition, 2011.

[4] A. Sturn, J. Quackenbush and Z. Trajanoski. Genesis: cluster analysis of microarray data. *Bioinformatics*, 18(1):207–208, 2002.

[5] W. Shannon, R. Culverhouse and J. Duncan. Analyzing microarray data using cluster analysis. *Pharmacogenomics*, 4(1):41–52, 2003.

[6] J. Quackenbush. Computational analysis of microarray data. *Nature reviews genetics*, 2(6):418–427, 2001.

[7] M. C. de Souto, I. G. Costa, D. S. de Araujo, T. B. Ludermir and A. Schliep. Clustering cancer gene expression data: a comparative study. *BMC bioinformatics*, 9(1):1, 2008.

[8] W. Li, L. Jaroszewski and A. Godzik. Clustering of highly homologous sequences to reduce the size of large protein databases. *Bioinformatics*, 17(3):282–283, 2001.

[9] W. Li and A. Godzik. Cd-hit: a fast program for clustering and comparing large sets of protein or nucleotide sequences. *Bioinformatics*, 22(13):1658–1659, 2006.

[10] F. Servant, C. Bru, S. Carrère, E. Courcelle, J. Gouzy, D. Peyruc and D. Kahn. Prodom: automated clustering of homologous domains. *Briefings in bioinformatics*, 3(3):246–251, 2002.

[11] A. J. Enright and C. A. Ouzounis. Generage: a robust algorithm for sequence clustering and domain detection. *Bioinformatics*, 16(5):451–457, 2000.

[12] R. C. Edgar. Search and clustering orders of magnitude faster than blast. *Bioinformatics*, 26(19):2460–2461, 2010.

[13] T. Wittkop, D. Emig, A. Truss, M. Albrecht, S. Böcker and J. Baumbach. Comprehensive cluster analysis with transitivity clustering. *Nature protocols*, 6(3):285–295, 2011.

[14] R. Xu and D. Wunsch. Survey of clustering algorithms. *IEEE Transactions on Neural Networks*, 16(3):645–678, 2005.

[15] R. Xu and D. Wunsch. Clustering algorithms in biomedical research: a review. *IEEE Rev Biomed Eng*, 3:120–154, 2010. URL http://dx.doi.org/10.1109/RBME.2010.2083647.

[16] G. W. Milligan. Clustering validation: results and implications for applied analyses. *Clustering and Classification*, 1:341–375, 1996.

[17] C. Wiwie, J. Baumbach and R. Röttger. Comparing the performance of biomedical clustering methods. *Nature methods*, 12(11):1033–1038, 2015.

[18] M. R. Anderberg. *Cluster analysis for applications: probability and mathematical statistics: a series of monographs and textbooks*, volume 19. Academic press, 2014.

[19] A. K. Jain, M. N. Murty and P. J. Flynn. Data clustering: a review. *ACM computing surveys (CSUR)*, 31(3):264–323, 1999.

[20] K. C. Gowda and E. Diday. Symbolic clustering using a new similarity measure. *IEEE Transactions on Systems, Man, and Cybernetics*, 22(2):368–378, 1992.

[21] H. Liu and L. Yu. Toward integrating feature selection algorithms for classification and clustering. *IEEE Transactions on knowledge and data engineering*, 17(4):491–502, 2005.

[22] I. Guyon and A. Elisseeff. An introduction to variable and feature selection. *Journal of machine learning research*, 3(Mar):1157–1182, 2003.

[23] M. Dash and H. Liu. Feature selection for clustering. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 110–121. Springer, 2000.

[24] V. Roth and T. Lange. Feature selection in clustering problems. In *Advances in Neural Information Processing Systems*, pages 473–480. 2004.

[25] S. Alelyani, J. Tang and H. Liu. Feature selection for clustering: A review. *Data Clustering: Algorithms and Applications*, 29, 2013.

[26] I. Jolliffe. *Principal component analysis*. Wiley Online Library, 2002.

[27] J. B. Kruskal and M. Wish. *Multidimensional scaling*, volume 11. Sage, 1978.

[28] T. F. Cox and M. A. Cox. *Multidimensional scaling*. CRC press, 2000.

[29] M. A. Kramer. Nonlinear principal component analysis using autoassociative neural networks. *AIChE journal*, 37(2):233–243, 1991.

[30] T. Hastie and W. Stuetzle. Principal curves. *Journal of the American Statistical Association*, 84(406):502–516, 1989.

[31] D. Dong and T. J. McAvoy. Nonlinear principal component analysisbased on principal curves and neural networks. *Computers & Chemical Engineering*, 20(1):65–78, 1996.

[32] C. C. Aggarwal and C. K. Reddy. *Data clustering: algorithms and applications*. CRC Press, 2013.

[33] I. Guyon and A. Elisseeff. An introduction to feature extraction. In *Feature extraction*, pages 1–25. Springer, 2006.

[34] N. K. Visalakshi and K. Thangavel. Impact of normalization in distributed k-means clustering. *International Journal of Soft Computing*, 4(4):168–172, 2009.

[35] L. Al Shalabi, Z. Shaaban and B. Kasasbeh. Data mining: A preprocessing engine. *Journal of Computer Science*, 2(9):735–739, 2006.

[36] G. Gan, C. Ma and J. Wu. *Data clustering: theory, algorithms, and applications*, volume 20. Siam, 2007.

[37] K. Jajuga and M. Walesiak. Standardisation of data set under different measurement scales. In *Classification and information processing at the turn of the millennium*, pages 105–112. Springer, 2000.

[38] S. Y. Kim, J. W. Lee and J. S. Bae. Effect of data normalization on fuzzy clustering of dna microarray data. *BMC bioinformatics*, 7(1):134, 2006.

[39] Y. Fujikawa and T. Ho. Cluster-based algorithms for dealing with missing values. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 549–554. Springer, 2002.

[40] O. Troyanskaya, M. Cantor, G. Sherlock, P. Brown, T. Hastie, R. Tibshirani, D. Botstein and R. B. Altman. Missing value estimation methods for dna microarrays. *Bioinformatics*, 17(6):520–525, 2001.

[41] T. H. Bø, B. Dysvik and I. Jonassen. Lsimpute: accurate estimation of missing values in microarray data with least squares methods. *Nucleic acids research*, 32(3):e34–e34, 2004.

[42] R. J. Hathaway and J. C. Bezdek. Fuzzy c-means clustering of incomplete data. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 31(5):735–744, 2001.

[43] D.-Q. Zhang and S.-C. Chen. Clustering incomplete data using kernel-based fuzzy c-means algorithm. *Neural Processing Letters*, 18(3):155–162, 2003.

[44] D. Li, J. Deogun, W. Spaulding and B. Shuart. Towards missing data imputation: a study of fuzzy k-means clustering method. In *International Conference on Rough Sets and Current Trends in Computing*, pages 573–579. Springer, 2004.

[45] R. Röttger, C. Kreutzer, T. D. Vu, T. Wittkop and J. Baumbach. Online transitivity clustering of biological data with missing values. In *OASIcs-OpenAccess Series in Informatics*, volume 26. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2012.

[46] S. Pandit, S. Gupta et al. A comparative study on distance measuring approaches for clustering. *International Journal of Research in Computer Science*, 2(1):29–31, 2011.

[47] J. Mao and A. K. Jain. A self-organizing network for hyperellipsoidal clustering (hec). *IEEE transactions on neural networks*, 7(1):16–29, 1996.

[48] C. C. Aggarwal, A. Hinneburg and D. A. Keim. On the surprising behavior of distance metrics in high dimensional space. In *International Conference on Database Theory*, pages 420–434. Springer, 2001.

[49] A. Brazma and J. Vilo. Gene expression data analysis. *FEBS letters*, 480(1):17–24, 2000.

[50] D. J. Rogers and T. T. Tanimoto. A computer program for classifying plants. *Science (New York, NY)*, 132(3434):1115–1118, 1960.

[51] J. C. Gower and P. Legendre. Metric and euclidean properties of dissimilarity coefficients. *Journal of classification*, 3(1):5–48, 1986.

[52] P. Jaccard. Nouvelles recherches sur la distribution floral. *Bull. Soc. Vard. Sci. Nat*, 44:223–270, 1908.

[53] P. H. Sneath and R. R. Sokal. Numerical taxonomy. the principles and practices of numerical classification. *WF Freeman and Co., San Francisco*, 573, 1973.

[54] D. W. Goodall. A new similarity index based on probability. *Biometrics*, pages 882–907, 1966.

[55] S. Boriah, V. Chandola and V. Kumar. Similarity measures for categorical data: A comparative evaluation. *red*, 30(2):3, 2008.

[56] J. C. Gower. A general coefficient of similarity and some of its properties. *Biometrics*, pages 857–871, 1971.

[57] T. H. Jukes and C. R. Cantor. Evolution of protein molecules. *Mammalian protein metabolism*, 3(21):132, 1969.

[58] S. F. Altschul, W. Gish, W. Miller, E. W. Myers and D. J. Lipman. Basic local alignment search tool. *Journal of molecular biology*, 215(3):403–410, 1990.

[59] A. R. Ortiz, C. E. Strauss and O. Olmea. Mammoth (matching molecular models obtained from theory): an automated method for model comparison. *Protein Science*, 11(11):2606–2621, 2002.

[60] T. Hastie, R. Tibshirani and J. Friedman. Unsupervised learning. In *The elements of statistical learning*, pages 485–585. Springer, 2009.

[61] A. K. Jain, R. P. W. Duin and J. Mao. Statistical pattern recognition: A review. *IEEE Transactions on pattern analysis and machine intelligence*, 22(1):4–37, 2000.

[62] G. W. Milligan and M. C. Cooper. Methodology review: Clustering methods. *Applied psychological measurement*, 11(4):329–354, 1987.

[63] D. Jiang, C. Tang and A. Zhang. Cluster analysis for gene expression data: a survey. *IEEE Transactions on knowledge and data engineering*, 16(11):1370–1386, 2004.

[64] B. Andreopoulos, A. An, X. Wang and M. Schroeder. A roadmap of clustering algorithms: finding a match for a biomedical application. *Briefings in Bioinformatics*, 10(3):297–314, 2009.

[65] M. E. Celebi, H. A. Kingravi and P. A. Vela. A comparative study of efficient initialization methods for the k-means clustering algorithm. *Expert Systems with Applications*, 40(1):200–210, 2013.

[66] J. M. Pena, J. A. Lozano and P. Larranaga. An empirical comparison of four initialization methods for the k-means algorithm. *Pattern recognition letters*, 20(10):1027–1040, 1999.

[67] L. Kaufman and P. Rousseeuw. *Clustering by means of medoids*. Statistical Data Analysis Based on the L1 Norm and Related Methods. North-Holland, 1987.

[68] J. C. Bezdek, R. Ehrlich and W. Full. Fcm: The fuzzy c-means clustering algorithm. *Computers & Geosciences*, 10(2-3):191–203, 1984.

[69] D. Dembélé and P. Kastner. Fuzzy c-means method for clustering microarray data. *Bioinformatics*, 19(8):973–980, 2003.

[70] D. L. Massart, L. Kaufman, P. J. Rousseeuw and A. Leroy. Least median of squares: a robust method for outlier and model error detection in regression and calibration. *Analytica Chimica Acta*, 187:171–179, 1986.

[71] P. Langfelder, B. Zhang and S. Horvath. Defining clusters from a hierarchical cluster tree: the dynamic tree cut package for r. *Bioinformatics*, 24(5):719–720, 2008.

[72] F. Corpet. Multiple sequence alignment with hierarchical clustering. *Nucleic acids research*, 16(22):10881–10890, 1988.

[73] J. Wang, M. Li, J. Chen and Y. Pan. A fast hierarchical clustering algorithm for functional modules discovery in protein interaction networks. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 8(3):607–620, 2011.

[74] Y. Loewenstein, E. Portugaly, M. Fromer and M. Linial. Efficient algorithms for accurate hierarchical clustering of huge datasets: tackling the entire protein space. *Bioinformatics*, 24(13):i41–i49, 2008.

[75] R. Jothi, E. Zotenko, A. Tasneem and T. M. Przytycka. Coco-cl: hierarchical clustering of homology relations based on evolutionary correlations. *Bioinformatics*, 22(7):779–788, 2006.

[76] D. G. Saunders, J. Win, L. M. Cano, L. J. Szabo, S. Kamoun and S. Raffaele. Using hierarchical clustering of secreted protein families to classify and rank candidate effectors of rust fungi. *PLoS One*, 7(1):e29847, 2012.

[77] M. B. Eisen, P. T. Spellman, P. O. Brown and D. Botstein. Cluster analysis and display of genome-wide expression patterns. *Proceedings of the National Academy of Sciences*, 95(25):14863–14868, 1998.

[78] B. J. Frey and D. Dueck. Clustering by passing messages between data points. *Science*, 315(5814):972–976, 2007.

[79] T. Nepusz, H. Yu and A. Paccanaro. Detecting overlapping protein complexes in protein-protein interaction networks. *Nature methods*, 9(5):471–472, 2012.

[80] S. Van Dongen. A cluster algorithm for graphs. *Report-Information systems*, (10):1–40, 2000.

[81] U. Von Luxburg. A tutorial on spectral clustering. *Statistics and computing*, 17(4):395–416, 2007.

[82] T. Wittkop, D. Emig, S. Lange, S. Rahmann, M. Albrecht, J. H. Morris, S. Böcker, J. Stoye and J. Baumbach. Partitioning biological data with transitivity clustering. *Nature methods*, 7(6):419–420, 2010.

[83] T. Aittokallio and B. Schwikowski. Graph-based methods for analysing networks in cell biology. *Briefings in bioinformatics*, 7(3):243–255, 2006.

[84] J. Vlasblom and S. J. Wodak. Markov clustering versus affinity propagation for the partitioning of protein interaction graphs. *BMC bioinformatics*, 10(1):1, 2009.

[85] V. Satuluri, S. Parthasarathy and D. Ucar. Markov clustering of protein interaction networks with improved balance and scalability. In *Proceedings of the First ACM International Conference on Bioinformatics and Computational Biology*, pages 247–256. ACM, 2010.

[86] Y.-K. Shih and S. Parthasarathy. Identifying functional modules in interaction networks through overlapping markov clustering. *Bioinformatics*, 28(18):i473–i479, 2012.

[87] C.-S. Liao, K. Lu, M. Baym, R. Singh and B. Berger. Isorankn: spectral methods for global alignment of multiple protein networks. *Bioinformatics*, 25(12):i253–i258, 2009.

[88] R. Röttger, P. Kalaghatgi, P. Sun, S. de Castro Soares, V. Azevedo, T. Wittkop and J. Baumbach. Density parameter estimation for finding clusters of homologous proteins-tracing actinobacterial pathogenicity life styles. *Bioinformatics*, page bts653, 2012.

[89] M. Ester, H.-P. Kriegel, J. Sander, X. Xu et al. A density-based algorithm for discovering clusters in large spatial databases with noise. In *KDD*, volume 96, pages 226–231. 1996.

[90] A. Hinneburg and D. A. Keim. An efficient approach to clustering in large multimedia databases with noise. In *KDD*, volume 98, pages 58–65. 1998.

[91] L. Rabiner and B. Juang. An introduction to hidden markov models. *IEEE ASSP Magazine*, 3(1):4–16, 1986.

[92] M. M. Van Hulle. Self-organizing maps. In *Handbook of Natural Computing*, pages 585–622. Springer, 2012.

[93] C. Fraley and A. E. Raftery. Model-based clustering, discriminant analysis, and density estimation. *Journal of the American statistical Association*, 97(458):611–631, 2002.

[94] K. Y. Yeung, C. Fraley, A. Murua, A. E. Raftery and W. L. Ruzzo. Model-based clustering and data transformations for gene expression data. *Bioinformatics*, 17(10):977–987, 2001.

[95] M. Medvedovic and S. Sivaganesan. Bayesian infinite mixture model based clustering of gene expression profiles. *Bioinformatics*, 18(9):1194–1206, 2002.

[96] G. J. McLachlan, R. Bean and D. Peel. A mixture model-based approach to the clustering of microarray expression data. *Bioinformatics*, 18(3):413–422, 2002.

[97] P. Smyth et al. Clustering sequences with hidden markov models. *Advances in neural information processing systems*, pages 648–654, 1997.

[98] T. Zhang, R. Ramakrishnan and M. Livny. Birch: an efficient data clustering method for very large databases. In *ACM Sigmod Record*, volume 25, pages 103–114. ACM, 1996.

[99] S. Guha, R. Rastogi and K. Shim. Cure: an efficient clustering algorithm for large databases. In *ACM SIGMOD Record*, volume 27, pages 73–84. ACM, 1998.

[100] A. K. Jain and R. C. Dubes. *Algorithms for clustering data.* Prentice-Hall, Inc., 1988.

[101] W. M. Rand. Objective criteria for the evaluation of clustering methods. *Journal of the American Statistical association*, 66(336):846–850, 1971.

[102] J. Handl, J. Knowles and D. B. Kell. Computational cluster validation in post-genomic data analysis. *Bioinformatics*, 21(15):3201–3212, 2005.

[103] E. B. Fowlkes and C. L. Mallows. A method for comparing two hierarchical clusterings. *Journal of the American statistical association*, 78(383):553–569, 1983.

[104] L. Hubert and P. Arabie. Comparing partitions. *Journal of classification*, 2(1):193–218, 1985.

[105] J. C. Dunn. A fuzzy relative of the isodata process and its use in detecting compact well-separated clusters. 1973.

[106] J. C. Dunn. Well-separated clusters and optimal fuzzy partitions. *Journal of cybernetics*, 4(1):95–104, 1974.

[107] D. L. Davies and D. W. Bouldin. A cluster separation measure. *IEEE transactions on pattern analysis and machine intelligence*, (2):224–227, 1979.

[108] Y. Liu, Z. Li, H. Xiong, X. Gao, J. Wu and S. Wu. Understanding and enhancement of internal clustering validation measures. *IEEE transactions on cybernetics*, 43(3):982–994, 2013.

[109] M. Halkidi, Y. Batistakis and M. Vazirgiannis. Cluster validity methods: part i. *ACM Sigmod Record*, 31(2):40–45, 2002.

[110] M. Halkidi, Y. Batistakis and M. Vazirgiannis. Clustering validity checking methods: part ii. *ACM Sigmod Record*, 31(3):19–27, 2002.

[111] R. Tibshirani, G. Walther and T. Hastie. Estimating the number of clusters in a data set via the gap statistic. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 63(2):411–423, 2001.

[112] R. Agrawal, J. Gehrke, D. Gunopulos and P. Raghavan. *Automatic subspace clustering of high dimensional data for data mining applications*, volume 27. ACM, 1998.

[113] D. Barbará and P. Chen. Using the fractal dimension to cluster datasets. In *Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 260–264. ACM, 2000.

[114] G. Sheikholeslami, S. Chatterjee and A. Zhang. Wavecluster: A multi-resolution clustering approach for very large spatial databases. In *VLDB*, volume 98, pages 428–439. 1998.

[115] R. Apweiler, A. Bairoch, C. H. Wu et al. Uniprot: the universal protein knowledgebase. *Nucleic acids research*, 32(suppl 1):D115–D119, 2004.

[116] P. E. Bourne, K. J. Addess, W. F. Bluhm et al. The distribution and query systems of the rcsb protein data bank. *Nucleic acids research*, 32(suppl 1):D223–D225, 2004.

[117] Y. Chen, K. D. Reilly, A. P. Sprague and Z. Guan. Seqoptics: a protein sequence clustering system. *BMC bioinformatics*, 7(4):1, 2006.

[118] M. Hauser, C. E. Mayer and J. Söding. kclust: fast and sensitive clustering of large protein sequence databases. *BMC bioinformatics*, 14(1):1, 2013.

[119] A. Krause, J. Stoye and M. Vingron. Large scale hierarchical clustering of protein sequences. *BMC bioinformatics*, 6(1):1, 2005.

[120] J. S. Bernardes, F. R. Vieira, L. M. Costa and G. Zaverucha. Evaluation and improvements of clustering algorithms for detecting remote homologous protein families. *BMC bioinformatics*, 16(1):1, 2015.