LMU
LUDWIG-
MAXIMILIANS-
UNIVERSITÄT
MÜNCHEN

INSTITUT FÜR STATISTIK

Marcus Scherl

Benchmarking of Cluster Indices

Diploma Thesis
Department of Statistics, LMU Munich
Supervisor: Prof. Dr. Friedrich Leisch, Dipl.-Ing. Manuel Eugster
Closing Date: September, 30^{th} 2010

Hiermit erkläre ich, dass ich die Diplomarbeit selbstständig angefertigt und nur die angegebenen Quellen verwendet habe.

München, 15. September 2010 (Marcus Scherl)

Acknowledgement

Firstly I would like to thank Prof. Dr. Friedrich Leisch, who gave me the chance to write my diploma thesis in his Workgroup Computational Statistics. I would like to thank Dipl. Ing. Manuel Eugster for the opportunity to develop this thesis at the Department of Statistics. Furthermore I want to thank him for the qualified support to incorporate my own ideas into this thesis. I would like to thank as well Dipl. Stat. Sebastian Kaiser, who helped me in implementation problems. Special thanks to the working group, which was developed for such theses, for discussion the results and solving any problems. In addition, special thanks to my friend Juliane Manitz, who helped me during this time and discussed with me nearly every problem, that I have got a better point of view. I would like to thank all the other people who supported and assisted me in writing this thesis.

Contents

Chapter 1

Introduction

The aim of cluster analysis is to identify groups with similar objects and discover patterns, subsets and correlations of certain data sets. The main task of the analysis is to classify objects into respective categories, which is known as a cluster. The data points of the objects shall be more similar to each other in the same cluster than points of different groups, i.e. the greater the homogeneity within a group, the more distinct the clustering. As a method of unsupervised learning cluster analysis has applications in different fields such as Market Research, Medical Science, and Engineering. For example, in Market Research the segmentation of customers is very important for knowing their affiliation. The clustering procedures can group customers with similar buying interests into the same cluster.

As cluster analyses provide a sufficient way to discover structures within data without necessarily providing underlying reasons, many cluster algorithms have been developed over time. Examples are hierarchical clustering, k-means or k-medoids algorithm for partitioned clustering and the EM-algorithm for density-based clustering. Advantages and disadvantages of these techniques gives for example Hastie et al. (2009).

Cluster validation techniques are used to evaluate an applied cluster algorithm, i.e. to describe the quality of clustering. Hence, cluster algorithms need to be validated in terms of their general goodness of fit. Literature provides a wide range of techniques with different approaches. The main distinctive feature lies between external and internal performance measures. External validation techniques sources to the stability of the cluster partition, meaning that the focus of these techniques is based on the knowledge of the correct class labels. Major disadvantage of these measures is the true assignment has to be known for external indices. Therefore, internal validation techniques are used when their true cluster structure or correct class labels are unknown. No additional knowledge for the cluster assignments is required, but is generally based on their quality of labeling of data alone. The purpose of these validation techniques is to evaluate several cluster algorithms to obtain the best algorithm for the present data with an adequate result of a performance measure.

Most of the techniques and their achievement describes Halkidi et al. (2001) and Handl et al. (2005).

Currently, cluster analysis consists of three steps (Handl et al., 2005): (1) Data transformation (variable selection, normalization and choice of distance functions); (2) Selection of the cluster algorithm, associated with their parameters and, of course, the application of the algorithm to the data; (3) Evaluation of the cluster algorithm as an useful check of the partitioning. For the last step, cluster validation techniques are needed to improve the algorithms as well as the data transformations from the first two steps. This thesis will focus on the cluster validation techniques, gained by the strength and weakness of each validation measure.

Benchmarking experiments in statistical learning are empirical studies with the aim of comparing and ranking algorithms with respect to a performance measure. In these experiments, the primary intention is to evaluate certain algorithms. In particular, the aim of these studies is to find out which algorithm has the best fit for a given data set. Hothorn et al. (2005) define a statistically sound framework; Eugster et al. (2008) use the framework and give a practical direction for benchmark experiments with supervised learning algorithms. Following this direction, this thesis investigates benchmark experiments in case of unsupervised learning techniques. More specific, and in combination to common benchmark experiments, it validate the performance measure and not the candidate algorithm. Common validation measures in up-to-date publications are the *Rand-Index* as an external validation index (used, e.g. in Dolnicar and Leisch (2010) and Campello (2007)) and the *Calinski-Index* as an internal one (used, e.g. in Dolnicar and Leisch (2010) and Kryszczuk and Hurley (2010)). However, there exist a set of other indices, and no evidence is given that the mentioned measures are the best. In fact, the authors believe is, that the behaviour may depends on the data set. While discussing the strengths and weaknesses of each single validation index by changing small steps of the data with the respect to the cluster algorithm. Furthermore, it is aimed to find groups of validation measures and to test these measurements against each others.

This thesis is structured as follows: Section 2 gives an introduction to the data generating process and a review of several cluster validation indices, external as well as internal ones. The main focus is the reproducibility of these indices given by Hothorn et al. (2005) . Section 3 describes the generated data, which are developed suitable for benchmarking experiments, and the practical application of the cluster algorithm with `flexclust` (Leisch, 2006) is shown as well. Then, the external cluster indices are computed by using methods from the package `clv` (Nieweglowski., 2009). The internal measures are calculated using the package `clValid` (Brock et al., 2008) and `cclust` (Dimitriadou, 2009). Furthermore, the first descriptive analysis of the validation indices is done. Section 4 tests the indices among each others. In Section 5 the description of implemented R package is introduced to combine all further assumptions. Section 6 draws conclusions based on the previous chapters.

Chapter 2

Design of the Benchmark Experiment

The first part in chapter 2 shows the structure of the data generating process for reproducibility the data for the benchmarking experiment. Afterwards the cluster algorithm is introduced. This algorithm, in particular the *k-means* algorithm does not change in the whole experiment. The number of clusters only modifies the structure of the algorithm. Then, the validation indices are computed, either the external or the internal ones. By changing the generated data regarding their dimensions such as the standard deviation, i.e. from widely spaced clusters to completely noisy ones, it is assumed that the measures are acting differently. According to the computation, each group of validation indices (external and internal) is computed differently by certain samples of the generated data sets. The external indices are based on a pre-specified structure of the data and the internal indices are compared by the separation and compactness of each cluster. Caused to this computation, the reproducibility of the measurements are different of internal and external ones.

2.1 Bootstrapping Segmentation

The benchmark experiment follows the general framework by Hothorn et al. (2005), i.e. the *real*-world problem. The benchmark study consists of the following elements:

1. The data set $X = (x_1, \dots, x_N)$, where B samples of size N are drawn using the sampling method with replacement (bootstrapping):

$$\begin{aligned} X^1 &= \{x_1^1, \dots, x_N^1\} \sim \mathcal{X} \\ &\vdots \\ X^B &= \{x_1^B, \dots, x_N^B\} \sim \mathcal{X} \end{aligned}$$

2. The cluster algorithm a with $a(\cdot \mid X^b)$ is the fit of a learning sample X^b.

3. Then, the performance measure is given by a scalar function s:

$$s^b = s(a, X^b) \sim S = S(\mathcal{X})$$

Thus, s^b are samples drawn from the distribution $S(\mathcal{X})$ of the performance measures for the algorithm a on $\mathcal{X}$. As already noted, the interests lies in different performance measures s_i $(i = 1, \ldots, I)$:

$$s_i^b = s_i(a, X^b) \sim S_i = S_i(\mathcal{X})$$

4. A method to draw the test data set $\hat{X}^b$ is to empirically determine s_i^b:

$$\hat{s}_i^b = s_i(a, \hat{X}^b) \sim \hat{S}_i = \hat{S}_i(\mathcal{X})$$

Hence, X^b is computed through the following method $t = 1, \ldots, 3$:
1.) All observations of the original data are used, i.e. X on size N.
2.) The data points of a new sampled data are sampled from the original data set, i.e. $X^{b_{new}}$ on size N.
3 ext.) An out-of-bootstrap sample is drawn for the external validity indices, i.e. $(X \setminus X^{b_1}) \cap (X \setminus X^{b_2}) = X_{ext}^{OOB}$, which is a ninth of the original size.
3 int.) An out-of-bootstrap sample is drawn for the internal validity indices, i.e. $X \setminus X^b = X_{int}^{OOB}$, which is a third of the original data size.
That implies the definition for each performance measure in the respect to their sample method, announced as follows:

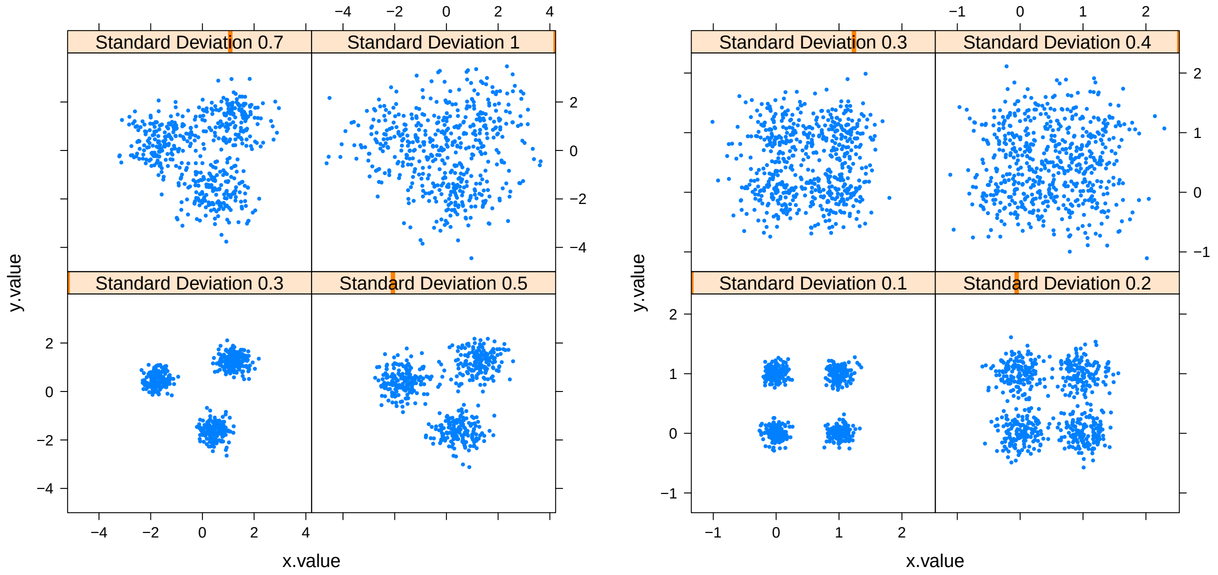
$$\hat{s}_i^{bt} = s_i(a, \hat{X}^{bt}) \sim \hat{S}_i = \hat{S}_i(\mathcal{X}).$$

Furthermore note, that S looks different for the external and internal validation indices, which are defined in section 2.3.1 and 2.4.2, respectively.

2.1.1 Structure of the data sets

This section shows the replication of the data sets in the simulation study by the `R` package `mlbench`. The description bases on Leisch and Dimitriadou (2010). Further functions of generating artificial data sets for benchmark experiments are listed there as well.

All of these artificial data sets are Gaussian distributed with several standard deviations. This should show the cluster problem from a well-separated dataset to a completely noisy data with no identifiable pattern caused by an increasing standard deviation. Figure 2.1 shows an example with several standard deviations, whereas the visualization is very simple. The first two items are visualized in two figures, while the third and fourth item are difficult for visualisation. By taking a look to Figure 2.1b, the last two explanations are more understandable.



(a) Three cluster partition for circled scenario

(b) Four cluster partition for cubed scenario

Figure 2.1: Generated data sets on certain standard deviations

The data set X is generated by four certain scenarios:

Three clusters in two dimensions: The clusters are Gaussian distributed and the centers of each cluster are placed in a circle around the origin with the radius $\sqrt{3}$. This data set consists of 500 data points, each cluster of around 166 points. Figure 2.1a shows three different Gaussian clusters of four standard deviations. As seen in this figure the well-separated cluster solution becomes more noisy with increasing variance. These deviations are defined from 0.1 up to 1 in the interval of 0.1 steps, i.e. 10 certain datasets are computed for each cluster problem.

Four clusters in two dimensions: These clusters are as well Gaussian distributed and the centers of each cluster are placed in each corner of a 2-dimensional cube. The number of clusters are computed by 2^d with d as the number of the dimension, in particular for this case: $2^2 = 4$. The data set consists of 624 data points, each cluster of 156 points. Figure 2.1b is visualizing the four different Gaussian clusters of four several standard deviations, i.e. the grid lies between 0.1 up to 0.4. For the following scenarios, these four standard deviations are chosen in the benchmark study.

- **8 clusters in three dimensions:** These clusters are Gaussian distributed such as in the previously computed dataset. The cluster centers are placed in each corner such as above, i.e. $2^3 = 8$ clusters in each corner of a cube. Furthermore, the data set consists of 1,248 data points, i.e. each cluster exists of exactly 156 points.

- **32 clusters in five dimensions:** The cluster centers of this data set are in the corners of a 5-dimensional hypercube and each cluster is Gaussian distributed, i.e. $2^5 = 32$ cluster in such a hypercube. Hence, this data set should consist of an increased number of data points as the previously generated data sets, in particular it consists of 4,992 data items, i.e. as well 156 data points within each cluster. This last cluster problem is added to look at the limits of the performance measures in high-dimensional datasets.

The choice of the inadequate looking size in all the data sets lies in the cluster assignment of the data points. In this case, the same number of points in the clusters is given for each dimension of the artificial data for comparing different dimensions. Formerly choices of the size in the data were 600 items for the two-dimensional cube, 1,500 items for the three-dimensional cube and 4,992 items for the five-dimensional hypercube. The five-dimensional data should be generated with 5,000 data points, but the functions passes a value which is dividable for the indicated number of clusters.

2.2 *k*-means Algorithm

In practice certain algorithms for cluster analysis are utilized. As announced in the previous chapter one cluster algorithm is used for the evaluation. Hence, the most popular partitioning method to locate natural clusters in a dataset is the k-means algorithm. Therefore, it is helpful to present this algorithm in a well-known form. This description is based on Hastie et al. (2009) and parts from Leisch (2006), especially for benchmark experiments.

The first step of this procedure is to start with a random set of K initial center points of the cluster $C(i)$, which the user pre-specified. These are the current cluster centers, i.e. the cluster centers are a set of K data points $\{m_1, \ldots, m_K\}$. In a second step each data points are assigned to the closest cluster center and the updated cluster centers are the current cluster means for the observed partition. For the third step a current set of means, the within cluster variance, will be minimized by using the euclidean distance, which is often used as a dissimilarity measure:

$$C(i) = \underset{1 \le k \le K}{\operatorname{argmin}} \parallel x_i - m_k \parallel^2$$

Then, the steps are an iterative repeated algorithm until convergence, i.e. the assignments of the data points do not change anymore. With a finite number of partitions to a local optimum the convergence in a finite number of iterations is guaranteed (Leisch, 2006).

The k-means algorithm can also generate empty clusters, which are identified and randomly re-initialized in every iteration by the used implementation. If there are empty cluster convergence would not reach in finite. By setting a maximum (100 times) of iterations the convergence is enforced. Furthermore, to reduce the non-optimal cluster solution the k-means algorithm runs repeatedly (10 times) by using randomized initializations. Only the best result with the smallest within cluster variance is returned (Handl et al., 2006).

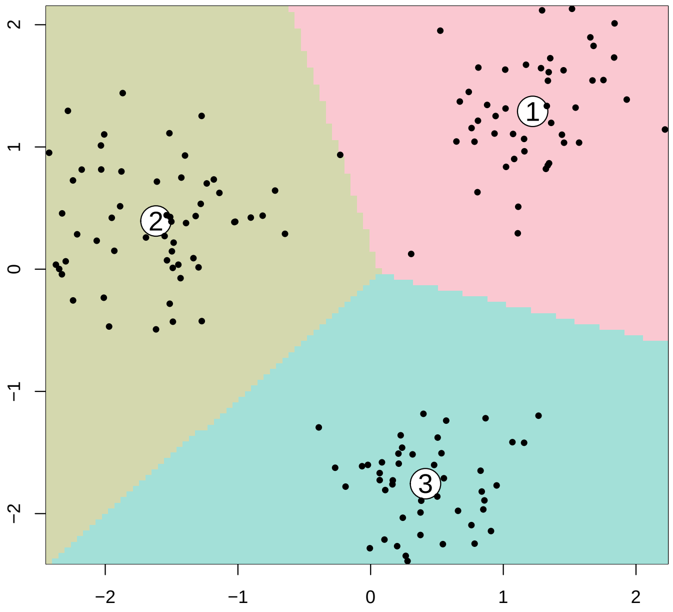


Figure 2.2: Example for a three cluster solution with their cluster centers

In Figure 2.2 an example for a three cluster problem is shown. The cluster centers are labeled randomly in the algorithm. This will be very important for computation of the external cluster indices. The data example is well-separated. Therefore, the convergence is reached in a small number of iterations.

2.3 Evaluating Reproducibility of External Cluster Indices

In the following, the external indices are calculated in a similar way as in Dolnicar and Leisch (2010), such as the reproducibility of these indices. Given the partitions X_1 and X_2 of two different bootstrap samples X^b, the cluster membership is predicted on the basis of the cluster algorithm by assigning each data point to the closest cluster center. Then, in order to the different samples of X^b, the k-means algorithm is computing the partition C^b. Using the predicted partitions $\tilde{C}_1$ and $\tilde{C}_2$, whereas respectively $\tilde{C}_1$ is computed by the sampling method of X^{1t}, the validation indices can be computed, as described in the following section 2.3.1. Each measurement is defined as

$$s_i^{bt} = \left(s_i^{1t} = s(\tilde{C}_1, \tilde{C}_2), \ldots, s_i^{Bt} = s(\tilde{C}_{2B-1}, \tilde{C}_{2B})\right) \sim S_i^t = S_i(\mathcal{X}^t)$$

where i stands for external validation indices, which are defined in the next section 2.3.1. The index t describes the certain sample methods. In addition, with $2B$ bootstrap partitions B independent and identically distributed replications of the external validation indices are computed, i.e. the algorithm runs twice and measures afterwards external validation index. The k-means algorithm runs 100 times, that altogether 50 passes are utilized for the training data set and 50 passes for the test data set. For calculating the external indices, one training data set and one test data set has to be drawn. In the following section, this process is described.

2.3.1 External Validation Indices

The external indices are calculated in a similar matter. The data points are counted in a pairwise co-assignment. Given two partitions named C_1 and C_2, the quantities a, b, c and d are computed for the pairs of the data points x_i and x_j and their cluster assignments $c_{C_1(i)}$, $c_{C_1(j)}$, $c_{C_2(i)}$ and $c_{C_2(j)}$ (Handl et al., 2005):

$$\begin{aligned}
a &= \left|\{x_i, x_j \mid c_{C_1(x_i)} = c_{C_1(x_j)} \wedge c_{C_2(x_i)} = c_{C_2(x_j)}\}\right| \\
b &= \left|\{x_i, x_j \mid c_{C_1(x_i)} = c_{C_1(x_j)} \wedge c_{C_2(x_i)} \neq c_{C_2(x_j)}\}\right| \\
c &= \left|\{x_i, x_j \mid c_{C_1(x_i)} \neq c_{C_1(x_j)} \wedge c_{C_2(x_i)} = c_{C_2(x_j)}\}\right| \\
d &= \left|\{x_i, x_j \mid c_{C_1(x_i)} \neq c_{C_1(x_j)} \wedge c_{C_2(x_i)} \neq c_{C_2(x_j)}\}\right|
\end{aligned}$$

The numbers a and d are counted as the agreements between the two cluster partitions C_1 and C_2, whereas b and c are the disagreements of these two partitions. Due to the random selection of the clusters, the pairs of data points are calculated. The cluster assignments are shown in Table 2.1 (Albatineh et al., 2006).

		Partition C_2	
	Number of pairs	same cluster	different cluster
Partition C_1	same cluster	a	b
	different cluster	c	d

Table 2.1: Similarity table of the cluster partitions

Table 2.2 (Albatineh et al., 2006) lists the external validation indices, which are counted by a, b, c and d. Most of the indices are between a range of 0 and 1, but however, the validation indices *Hamann, McConnaughey, Peirce, Gamma, Kruskal* and *Pearson* are defined on $[-1, 1]$ and *Fager* ranges between $-\frac{1}{2}$ and 1. Caused to the similarity of each other, these indices are self-explanatory.

Name	Symbol	Formula	Range
Rand	R	$\frac{a+d}{a+b+c+d}$	$[0, 1]$
Hamann	H	$\frac{(a+d)-(b+c)}{a+b+c+d}$	$[-1, 1]$
Czekanowski	CZ	$\frac{2a}{2a+b+c}$	$[0, 1]$
Kulczynski	K	$\frac{1}{2}(\frac{a}{a+b} + \frac{a}{a+c})$	$[0, 1]$
McConnaughey	MC	$\frac{a^2-bc}{(a+b)(a+c)}$	$[-1, 1]$
Peirce	PE	$\frac{ad-bc}{(a+c)-(b+d)}$	$[0, 1]$
Fowlkes and Mallows	FM	$\frac{a}{\sqrt{(a+b)(a+c)}}$	$[0, 1]$
Wallace (1)	$W1$	$\frac{a}{a+b}$	$[0, 1]$
Wallace (2)	$W2$	$\frac{a}{a+c}$	$[0, 1]$
Gamma	Γ	$\frac{ad-bc}{\sqrt{(a+b)(a+c)(c+d)(b+d)}}$	$[-1, 1]$
Sokal and Sneath (1)	$SS1$	$\frac{1}{4}(\frac{a}{a+b} + \frac{a}{a+c} + \frac{d}{d+b} + \frac{d}{d+c})$	$[0, 1]$
Russel and Rao	RR	$\frac{a}{a+b+c+d}$	$[0, 1]$
Fager and McGowan	FMG	$\frac{a}{\sqrt{(a+b)(a+c)}} - \frac{1}{2\sqrt{(a+b)}}$	$[-\frac{1}{2}, 1]$
Pearson	P	$\frac{ad-bc}{(a+b)(a+c)(c+d)(b+d)}$	$[-1, 1]$
Jaccard	J	$\frac{a}{a+b+c}$	$[0, 1]$
Sokal and Sneath (2)	$SS2$	$\frac{a}{a+2(b+c)}$	$[0, 1]$
Sokal and Sneath (3)	$SS3$	$\frac{ad}{\sqrt{(a+b)(a+c)(c+d)(b+d)}}$	$[0, 1]$
Gower and Legendre	GL	$\frac{a+d}{a+\frac{1}{2}(b+c)+d}$	$[0, 1]$
Rogers and Tanimoto	RT	$\frac{a+d}{a+2(b+c)+d}$	$[0, 1]$
Goodman and Kruskal	GK	$\frac{ad-bc}{ad+bc}$	$[-1, 1]$

Table 2.2: List of External Cluster Indices

2.4 Evaluating Reproducibility of Internal Cluster Indices

In the following, the internal indices are calculated in a similar way as in section 2.3. Given the partition X^b, which is drawn by a sample method with replacement, the cluster membership is predicted on the basis of the cluster algorithm by assigning each data point to the closest cluster center. Then, the k-means algorithm is computing the partition C_b. Using the predicted partition $\tilde{C}_b$, whereas respectively $\tilde{C}_b$ is computed by the sampling method of X^{bt}, the validation indices can be computed, as described in the following section 2.4.2. Each measurement is defined as

$$s_i^{bt} = \left(s_i^{1t} = s(\tilde{C}_1), \dots, s_i^{2Bt} = s(\tilde{C}_{2B})\right) \sim S_i^t = S_i(\mathcal{X}^t)$$

where i stands for internal validation indices, which are defined in the next section 2.4.2. The index t describes the certain sample methods. Hence, with $2B$ bootstrap partitions, $2B$ independent and identically distributed replications of the internal validation indices can be computed, i.e. the algorithm runs once and measures afterwards the internal validation indices, respectively. In total, the k-means algorithm runs 100 times, that altogether 100 passes are utilized for the different data samples and 100 certain validation indices can be computed.

2.4.1 Preview of the Internal Validation Indices

Other than external validation indices, the internal ones take the clustering on the basis of the used partition as the input. They use the obtained results information intrinsic of the data to assess the quality of clustering. The internal indices can be categorized into three certain types, as well as a combination of the categorization (Handl et al., 2005):

Compactness The data points of the same cluster should be as close to each other as possible, in particular regarding their homogeneity. The intra-cluster variance is one of the representatives of this category. Another measure is the sum-of-squared errors variance criterion, which is locally optimized by the k-means. Both of these measures are yielding to minimize their value for optimization.

Connectedness This type of validation technique attempts to assess that the neighbouring data points should share the same cluster. In principle density-based cluster algorithms are chosen for such a grouping problem. This is well-suited for arbitrary shaped clusters, but in fact it loses robustness in spatial separation.

Separation The clusters should be widely spaced between each other. There are certain distance possibilities to measure the space between two clusters.

Besides the internal cluster validation indices, there exist the relative criteria. This group of indices is evaluating cluster structure by comparing other cluster validation schemes, i.e. evaluating of the same cluster algorithm by using validation measures with different and changeable parameter values. Halkidi et al. (2001) announced the *Dunn-like indices*, which is similar to the below listed *Dunn-Index*, e.g. by using other linkage distances for separation of several clusters. As well announced is the *SD-Index* by using different weighting measures for the relation of the inter- or intra-cluster variance. In this benchmark experiment, the relative criteria is not considered in this thesis.

2.4.2 Internal Validation Indices

Before introducing the internal validation indices, let define the maximization or minimization criteria in the respect to their differences. Some of the validation indices have their criteria regarding the computed values of the performance measure, but however, other validation indices have the stopping rule at the maximum or minimum difference between the investigated cluster solutions, either regarding the first or second differences. Hence, the stopping rule is also known as the "elbow" criteria, i.e. there is a positive or negative jump between the cluster solutions. For a better understanding of the differences, Figure 2.3 gives an overview of the first and second differences.

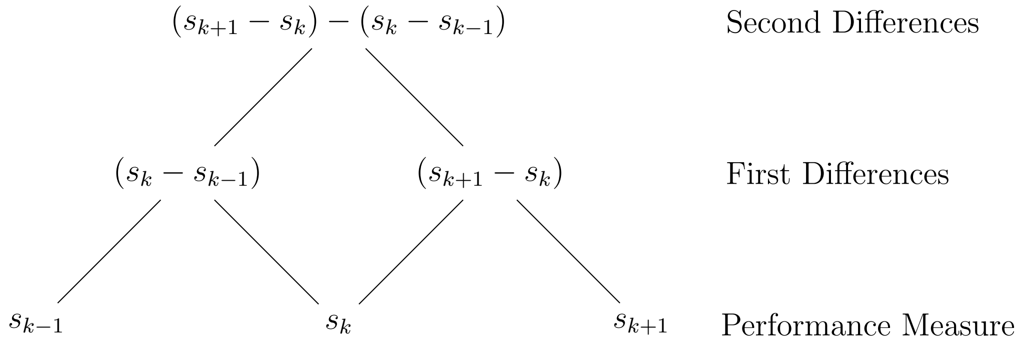


Figure 2.3: Hierarchy level of the first two differences in respect to the computed validation measurement.

The lowest hierarchy level is the computed performance measure, which are used either to minimize or to maximize regarding their criteria. However described in the definitions, a couple of the validation measures has to be maximized in the respect to the first differences. To reach either a positive or a negative "elbow", the maximum or minimum value of the second differences has to be reached.

The following description of the internal validity indices is based on the description by Handl et al. (2005), Brock et al. (2008) and Halkidi et al. (2001). The first four internal indices are most-common as validation techniques for the previous announced problems in order to investigate the correct number of clusters. Afterwards, the internal validation indices of Weingessel et al. (2002) are presented.

Connectivity Measure

The *Connectivity Index* shows the degree of connectedness of each cluster by evaluating its degree to each of all neighbouring data points in the same cluster. Let $nn_{i(j)}$ be the jth neighbour of the data point x_i; so $x_{i,nn_{i(j)}}$ is zero if x_i and $nn_{i(j)}$ are in the same cluster, otherwise the value is computed by $\frac{1}{j}$. For the cluster partition $\mathcal{C} = \{C_1, \ldots, C_K\}$ of each observation with the respect to the size N into K disjoint clusters, the measurement is computed as

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

where as mentioned above

$$x_{i,nn_{i(j)}} = \begin{cases} 1/\text{j} & \text{if } \nexists C_k : i \in C_k \wedge nn_{i(j)} \in C_k \\ 0 & \text{otherwise.} \end{cases}$$

Furthermore, L determines the number of neighbours that contribute to the connectivity measure. The measure is lying between zero and ∞ and has to be minimized.

Silhouette Index

The *Silhouette Width* for a particular partition is computed as the average Silhouette value for each data point. The value measures the degree of confidence in a particular cluster assignment for each individual data point x_i and is computed as

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

where

$$a_i = \frac{1}{|C(x_i)|} \sum_{x_j \in C(x_i)} dist(x_i, x_j) \quad \text{and} \quad b_i = \min_{C_k \in \mathcal{C} \setminus C(x_i)} \sum_{x_j \in C_k} \frac{dist(x_i, x_j)}{|C_k|}.$$

The value a_i is the average distance between the data point x_i to all the other observations in the same cluster; and b_i is the average distance between data point

x_i to all other points from the closest neighbouring cluster. In addition to this, b_i yields to minimum distance, of which the Euclidean distance is used as dissimilarity measure. Thus, certain distances can be chosen to get the minimum, such as the Manhattan distance. Furthermore, $C(x_i)$ denotes the cluster containing the data point x_i and $|C|$ is the number of elements in a particular cluster (cardinality). The *Silhouette Width* is limited between -1 and 1 and has to be maximized. The width is computed for each data point. To compare the width with other internal validation measures, the *Silhouette Index* is the average value of all the widths, but as well other measures of locations can be chosen, either median or mode.

Dunn Index

The *Dunn Index* measures the ratio of the shortest distance between each data point, which are not in the same cluster and the largest intra-cluster distance, i.e. it attempts to yield well-separated clusters. The measurement is computed as

$$D(\mathcal{C}) = \frac{\min\limits_{C_k, C_l \in \mathcal{C}, C_k \neq C_l} \left(\min\limits_{x_i \in C_k, x_j \in C_l} \operatorname{dist}(x_i, x_j) \right)}{\max\limits_{C_m \in \mathcal{C}} \operatorname{diam}(C_m)},$$

where $\operatorname{diam}(C_m)$ is maximum distance between each data item in the cluster C_m, and $\operatorname{dist}(x_i, x_j)$ is the distance between the pairs of the data points x_i and x_j within several clusters. This measurement lies between zero and ∞ and has to maximized.

Davies Bouldin index

The *Davies Bouldin index* is a similarity measure between two clusters C_i and C_j. This validity index measures the dispersion of each observed cluster (s_i) and the dissimilarity between two clusters (d_{ij}). Then, R_{ij} is defined as

$$R_{ij} = \frac{s_i + s_j}{d_{ij}}$$

and R_{ij} has to satisfy the following conditions:

1. $R_{ij} \geq 0$
2. $R_{ij} = R_{ji}$
3. if $s_i = 0$ and $s_j = 0$, then $R_{ij} = 0$
4. if $s_j > s_k$ and $d_{ij} = d_{ik}$, then $R_{ij} > R_{ik}$
5. if $s_j = s_k$ and $d_{ij} < d_{ik}$, then $R_{ij} < R_{ik}$

Thus, R_{ij} is the entry of a non-negative and symmetric matrix and the diagonal entries of this matrix have to be zero. The index is finally computed as

$$DB = \frac{1}{k}\sum_{i=1}^{k} R_i$$

where $R_i = \max_{i=1,\ldots,k \wedge i \neq j} R_{ij}$. The minimum value has to be taken as the proposed number of clusters.

Further Internal Cluster Performance Measures

As announced in 2.4.2, coming internal validation indices are based on the description of Weingessel et al. (2002) and firstly together published in Milligan and Cooper (1985).
These validation indices are divided into two several groups. The first group of indices is based on the distances of the sum-of-squares within the clusters (SSW) and the sum-of-squares between each cluster (SSB) in the solution. To simplify the computation, the SST, which is the sum-of-squares total, is defined as

$$SST = \sum_{i=1}^{n}(x_i - \bar{x})^2$$

with the corresponding

$$SSW = \sum_{j=1}^{k}\sum_{i=1}^{n_g}(x_{ij} - \bar{x}_j)^2$$

and

$$SSB = \sum_{i=1}^{k} n_i(\bar{x}_i - \bar{x})^2.$$

Here, $g = 1, \ldots, k$ are the number of groups, $n = \sum_{i=1}^{k} n_i$ are the number of data points and $\bar{x}_j$ stands for the center point of each cluster. Furthermore, SST can be written in an identity form, which is given as $SST = SSW + SSB$.

- The **Calinski** index, computed as $\frac{SSB/(k-1)}{SSW/(n-k)}$, has to be minimized at the second difference to investigate the correct cluster solution.
- The **Hartigan** index is based on the ratio of the logarithm between the within and between sum-of-squares. It is computed as $\log \frac{SSB}{SSW}$. To indicate the correct cluster solution, the minimum at the second differences has to be reached.

- The **Ball** Index is only based on the average distance of all the data points to their corresponding cluster centroid and is defined as $\frac{SSW}{k}$. The maximum of the second differences has to be taken as the correct number of clusters.

- The Index provided by **Xu** is computed as $d\log(\sqrt{SSW/(dn^2)}) + \log(k)$ with d as the dimension of the data points. As well to indicate the correct number of clusters, the maximum value of the second differences should be taken.

- The last index of the group is the **Ratkowsky** index. This validation measure is also based on the sum-of-squares, but this time on the sum-of-squares of each variable by its own. Then, the index is computed as $\tilde{c}/\sqrt{k}$ with $\tilde{c} = \text{mean}\sqrt{\text{var}SSB/\text{var}SST}$. The abbreviation 'var' stands for each variable. In particular, varSST is the total sum-of-squared distance for each variable. Accordingly, the mean is calculated for the ratio between distance of the variables and total distance of each variable to the overall mean. As the correct number of the clusters the maximum difference at the right side is taken as the best solution.

The second part of the internal validation indices are based on the statistic of T, which is the scatter distance matrix of the data points. Hence, the total scatter matrix of n data items (Friedman and Rubin, 1967) is given by

$$T = \sum_{i=1}^{n}(x_i - \bar{x})(x_i - \bar{x})'$$

Furthermore, to compute the indices, the pooled-within groups scatter matrix is needed and is defined as W. To compute this matrix, the within scatter matrix of each group W_g is given by

$$W_g = \sum_{i=1}^{n_g}(x_{ig} - \bar{x}_g)(x_{ig} - \bar{x}_g)'$$

where $g = 1, \ldots, k$ the number of groups, $n = \sum_{i=1}^{k} n_i$ and $\bar{x}_g$ as the center point in each group. Then, W can defined as

$$W = \sum_{i=1}^{k} W_g$$

such as B, which is the between groups scatter matrix, given as

$$B = \sum_{i=1}^{k} n_i \bar{x}_i \bar{x}_i'.$$

Finally, the equation $T = W + B$ is valid, whereas the identity form is similar to the mentioned equation of SST.

- The **Scott** index is defined as $n \log \frac{\det(T)}{\det(W)}$, in which n is the number of the data points. Here, the maximum difference at the left has to be reached for the proposed number of clusters.
- **Marriot** is computed as $k^2 \det(W)$, where k is the number of clusters in the investigated cluster solution. The maximum of the second differences has to be reached to get the proposed number of clusters in the solution.
- For **trace(covW)** the minmum of the second differences is the best for the investigated cluster solution.
- Similar to the performance measure above the next index is defined as **trace(W)** and has to be maximized at the second differences for the proposed number in the cluster approach.
- The validation measure **Friedman** is computed as $trace(W^{-1}B)$. The difference reaches their maximum at the left side for the proposed number of cluster.
- **Rubin** is defined as $\det(T)/\det(W)$ and has to reach its minimum at the second differences for the proposed number of clusters.

In the `R` package `cclust`, three more indices were provided. One of these performance measures, known as the *C Index*, is for evaluating binary data sets, which are not considered in this benchmark study. Another one is known as *SSI*, an abbreviation for *Simple Structure Index*. For this performance measure, unfortunately the formula is not provided in Weingessel et al. (2002). As the last of these measurements, the *Likelihood*, shortly *NLL*, is also not considered in this experiment. This measurement has displayed to much errors for the test data sets, and therefore this validation measure is dropped out of the benchmark experiment.

Chapter 3

Explorative Analysis

Firstly, the beginning of this section deals with the gained data structure. Exemplary, the results for external validation indices is shown for the three-cluster solution in a representable way. The implementation results in a four dimensional array with the first dimension the sampling, the second the cluster size, the third the certain standard deviations of the generated data sets and the fourth dimension are the given validation indices. The structure of the internal validation indices are similar, but the first dimension is doubled to the structure of the external indices results; i.e. the external indices are computed 50 times, while the internal measures are computed 100 times.

```
> str(result.2dnorm.orig)

 num [1:50, 1:6, 1:10, 1:20] 1 0.149 1 0.149 0.149 ...
 - attr(*, "dimnames")=List of 4
  ..$ : NULL
  ..$ : chr [1:6] "2" "3" "4" "5" ...
  ..$ : chr [1:10] "0.1" "0.2" "0.3" "0.4" ...
  ..$ : chr [1:20] "Hamann" "Czekanowski" "Kulczinski" "McConnaughey" ...

> dim(result.2dnorm.orig)

[1] 50  6 10 20
```

Furthermore, to be mentioned before analysing the external and internal validation indices, the following results are only a small choice of the whole benchmark study. The analysis of all results would exceed the scope of the thesis, due to a high amount of the investigated results. For details of all solutions, the results are given in the electronic appendix of this thesis.

3.1 Discussion of the External Validation Measures

This section shows the external validation indices among each other. Furthermore, this section covers the strength and weakness of the external indices by a choice of the generated data sets and the corresponding standard deviation; and the sample method is selected, in which the indices are computed.

3.1.1 Proposed Number of Clusters Resulted by a Three Cluster Solution

Firstly, to keep the overview of the external validation indices, Figure 3.1 shows the Box-Wisker-Plots of the validation indices. As announced in previous sections, each external validation index has to reach its maximum value at 1 for the proposed number of clusters given by the algorithm and through the generating of the artificial data sets. In this figure, the three-cluster solution is the correct number of clusters. Already in this figure, it is peculiar, that only two measurements are acting kind of unusual as the other ones. Most of the external validation indices have their maximum at the correct number of clusters, i.e. the proposed number of clusters is, in fact, the correct number. Except two of them, in particular the *Pearson* and *Russel Index*, are proceeding unlike as favored.

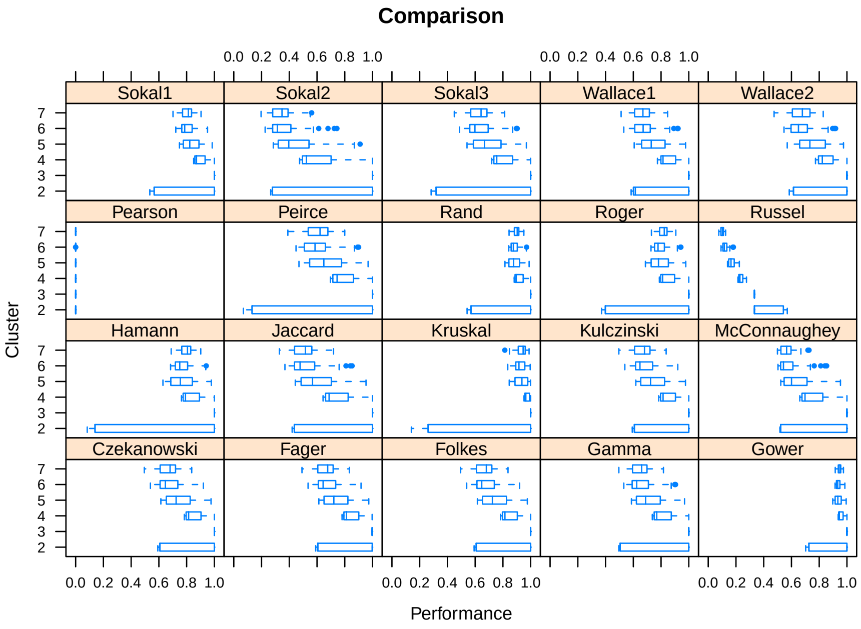


Figure 3.1: Comparison of the external validation measurements by their computed indices in well separated data

	2	3	4	5	6	7	2/3	2/4	3/4	2/3/4	2/3/4/5
Czekanowski	0	29	0	0	0	0	20	0	0	1	0
Fager	21	29	0	0	0	0	0	0	0	0	0
Folkes	0	29	0	0	0	0	20	0	0	1	0
Gamma	0	29	0	0	0	0	20	0	0	1	0
Gower	0	29	0	0	0	0	20	0	0	1	0
Hamann	0	29	0	0	0	0	20	0	0	1	0
Jaccard	0	29	0	0	0	0	20	0	0	1	0
Kruskal	0	29	0	0	0	0	20	0	0	1	0
Kulczinski	0	29	0	0	0	0	20	0	0	1	0
McConnaughey	0	29	0	0	0	0	20	0	0	1	0
Pearson	0	6	0	4	12	28	0	0	0	0	0
Peirce	0	29	0	0	0	0	20	0	0	1	0
Rand	0	29	0	0	0	0	20	0	0	1	0
Roger	0	29	0	0	0	0	20	0	0	1	0
Russel	21	0	0	0	0	0	29	0	0	0	0
Sokal1	0	29	0	0	0	0	20	0	0	1	0
Sokal2	0	29	0	0	0	0	20	0	0	1	0
Sokal3	0	29	0	0	0	0	20	0	0	1	0
Wallace1	0	29	0	0	0	0	20	0	0	1	0
Wallace2	0	29	0	0	0	0	20	0	0	1	0

Table 3.1: Proposed number of clusters through the indices, whereas the maximum values are counted. Indices are computed by a new sample data. The data is generated with a standard deviation of 0.3 and should reflect a two-dimensional 3-cluster solution.

To explain the weakness, such as the strength of the validation indices, further graphics are given to compare all the external indices among each other. Table 3.1 shows the proposed number of clusters for the first scenario in the simulation study. Here, the cluster assignment is predicted on the original data set. The three cluster solution is well-separated, in particular the data is generated with a standard deviation of 0.3. Figure 2.1a of the previous section shows the cluster solution for such a scenario. This table reflects the maximum value for each index and the investigated cluster solution, i.e. how often the index has its maximum value in different cluster scenarios. In some cases the maximum value is ambiguous, i.e. more than one value reaches the maximum value of 1 accurately. Then, these ties are counted as themselves between the proposed cluster solution, e.g. the maximum value reaches 1 in the two- and three-cluster solution. Hence, the maximum value is counted as a tie between the two solutions. Sometimes the maximum value is simultaneously existing in three different results. Thus, the tie is between all of the cluster solutions. Furthermore, the external indices differ in their ranges (compare section 2.3.1 for details), but in this example the maximum value is taken, either their ranges are between $[-1, 1]$ or $[0, 1]$. However, most of the indices choose the maximum value for the three cluster solution.

3.1.2 Proposed Number of Clusters Resulted by a Eight Cluster Solution

	4	6	8	10	12	4/6	4/8	6/8	4/6/8
Czekanowski	3	0	47	0	0	0	0	0	0
Fager	3	0	47	0	0	0	0	0	0
Folkes	3	0	47	0	0	0	0	0	0
Gamma	2	0	48	0	0	0	0	0	0
Gower	0	0	50	0	0	0	0	0	0
Hamann	0	0	50	0	0	0	0	0	0
Jaccard	3	0	47	0	0	0	0	0	0
Kruskal	1	0	49	0	0	0	0	0	0
Kulczinski	3	0	47	0	0	0	0	0	0
McConnaughey	3	0	47	0	0	0	0	0	0
Pearson	0	0	31	12	7	0	0	0	0
Peirce	2	0	48	0	0	0	0	0	0
Rand	0	0	50	0	0	0	0	0	0
Roger	0	0	50	0	0	0	0	0	0
Russel	36	6	8	0	0	0	0	0	0
Sokal1	2	0	48	0	0	0	0	0	0
Sokal2	3	0	47	0	0	0	0	0	0
Sokal3	2	0	48	0	0	0	0	0	0
Wallace1	3	0	47	0	0	0	0	0	0
Wallace2	3	0	47	0	0	0	0	0	0

Table 3.2: Proposed number of clusters through the indices, whereas the maximum values are counted. Indices are computed by a new sample data. The data is generated with a standard deviation of 0.2 and should reflect a 8-cluster solution in a cube.

Table 3.2 reflects the external validation indices are computed by the prediction to third data sample. The data is well-separated such as in the example above. In this solution the proposed number of the clusters should be eight. The membership of the data points can be either to a particular cluster or the cluster assignments of the points will be to a neighbouring cluster. Hence, the maximum values are located in the 8-cluster solution, however the maximum value will be reached by the weaker indices, which were announced in Table 3.1. From now on, further tables provides another stepwise solution of the performance measures for each number of clusters. That means, the identification of the maximum value can be either in 2 steps or as seen in the next example in 3 steps between the clusters. The *Fager Index* is looking much better for the three-dimensional generated data. Notable is, that nearly all indices reaches their maximum value for the true number of clusters; and none of the external validation indices reaches their maximum values with ties and every index reaches at least one of the maximum in the three dimensional scenario. This could be a relation to the curse of the dimensionality, i.e. the results are ambiguous in high-dimensional cluster solutions.

3.1.3 Proposed Number of Clusters Resulted by a 32 Cluster Solution

	26	29	32	35	38	26	29	32	35	38
Czekanowski	0	2	8	17	23	0	0	29	21	0
Fager	0	2	8	17	23	0	0	29	21	0
Folkes	0	2	8	17	23	0	0	29	21	0
Gamma	0	2	8	17	23	0	0	29	21	0
Gower	0	0	3	16	31	0	0	19	27	4
Hamann	0	0	3	16	31	0	0	19	27	4
Jaccard	0	2	8	17	23	0	0	29	21	0
Kruskal	0	0	6	21	23	0	0	27	23	0
Kulczinski	0	2	8	17	23	0	0	29	21	0
McConnaughey	0	2	8	17	23	0	0	29	21	0
Pearson	0	0	0	6	44	0	0	5	30	15
Peirce	0	1	3	17	29	0	0	29	21	0
Rand	0	0	3	16	31	0	0	19	27	4
Roger	0	0	3	16	31	0	0	19	27	4
Russel	41	7	2	0	0	13	12	25	0	0
Sokal1	0	2	8	17	23	0	0	29	21	0
Sokal2	0	2	8	17	23	0	0	29	21	0
Sokal3	0	2	8	17	23	0	0	29	21	0
Wallace1	0	1	8	20	21	0	0	29	21	0
Wallace2	0	1	3	18	28	0	0	29	21	0

Table 3.3: Left-sided the well-separated cluster solution (sd=0.1) and right-sided the noisier one (sd=0.3) of the original data set.

Table 3.3 shows an overview of the results in a well-separated dataset with $sd = 0.1$ (left-sided) and noisy dataset with $sd = 0.3$ (right-sided). For the proposed number of clusters should be chosen 32 groups as the number with the most maximum values. Not indicated for such a well-separated data set are the correct number of these groups, well seen in the table. Better results are provided by the noisy data, i.e. more external validation indices indicate their maximum values at the true number of clusters. However, it is unnecessary to compute the indices with ties. In such high-dimensional data sets, ties are implausible. Hence, if the case access as a result, ties between cluster solutions are disabled caused by choosing randomly the number of clusters. Furthermore, Figure 3.2 realizes the maximum value with a small peak at the correct number. Through the randomly chosen initial points of the k-means algorithm, some data points cannot assign to a particular cluster. The well-separation of the data sets could be a reason for it, because of the empty space between each cluster. Due to this, the respectively clusters, which is generated in the original data set, cannot be found in a finite time and the assignment to particular clusters is forced by the implementation of the algorithm.

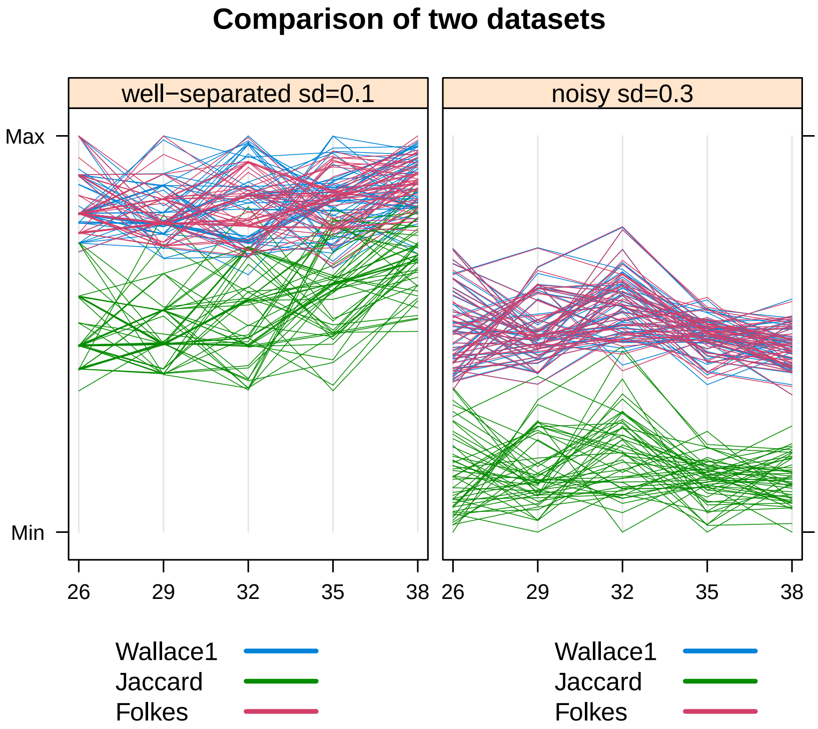


Figure 3.2: Overview of three chosen performance measures valued with two kinds of datasets

As an example, the *Rand Index* provides the correct grouping as the proposed number of clusters in the table. In details, Figure 3.3 shows the peak at the true number of clusters, but indeed the index works worser than the other ones. For the *Rand Index*, the proposed number of clusters should be chosen between 32 and 35 clusters. From the left side, it seems that the *Rand Index* increases more than on the right side at the correct number of clusters (32). The maximum difference on the left, like it seems, is reached for the proposed number of clusters as the true ones. Unfortunately, differences, however until now, are not an issue for external validation indices.

For aimless results of the external indices aspects the curse of the dimensionality as well. Due to large number of clusters and the corresponding small number of data items in the generated data sets, non-ambiguous results of the indices cannot be reached as the correct number of clusters. Here, only a group of number of clusters can be given for proposed numbers.

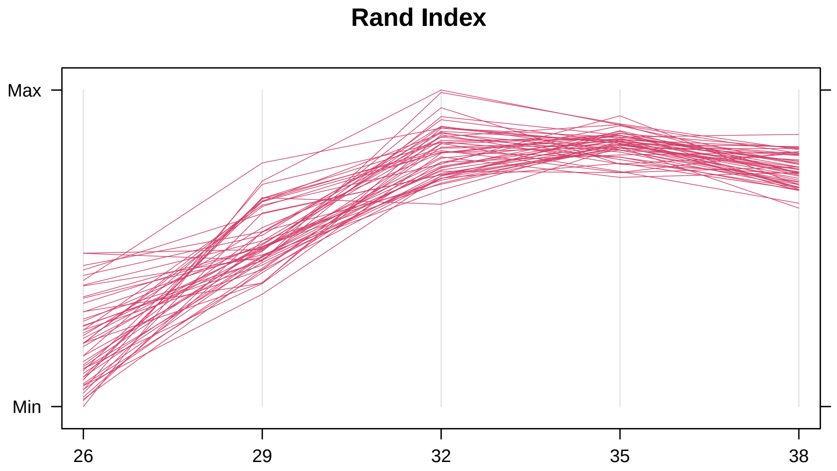


Figure 3.3: Detailed aspect of the Rand Index in a noisy generated dataset

3.1.4 Problems of the External Validation Indices

In Figure 3.4, six indices are selected to show badly fitting and well fitting validation indices. The validation indices are computed by the same data, as described above. The *Pearson* and *Russel Index* indicate neither their maximum nor minimum for the correct number of cluster. A better visual result is given in Figure 3.5. Here, it can be seen that the *Pearson Index* is not only a straight line, i.e. the index reaches its maximum at the three-cluster solution, but indeed dispers after it. Only the *Fager Index* seems to indicate most of its maximum values for the correct cluster solution, but has an ambiguous maximum between the two and three-cluster solution. Seen in Table 3.1, most maximum values are at ties between these two solutions. The *Rand Index* seems to be very similar to the *Fager Index*, but is better indicating its maximum value for the correct cluster solution. Possible explanations for the fluctuation of the maximum values in the computation such as in the formula of the validation indices are:

Russel Index

The weakness of *Russel Index* is described at first. This performance measure is very similar to the *Rand Index*. In fact, the denominator of both measurements are the same, but the numerator is different. *Russel* counts the sum of pairs in the same cluster, while ignoring pairs in different clusters of two partitions. The validation index is disable to reflect the randomness of cluster assignments in k-means such as other algorithms, either hierarchical or partitioned clustering. Due to the randomness of the assignment, well-performed results cannot be achieved, such as the value 1 for best-fitting clusters. Thus, the validation index might be useless for validation of cluster algorithms, however if there are provided better validation

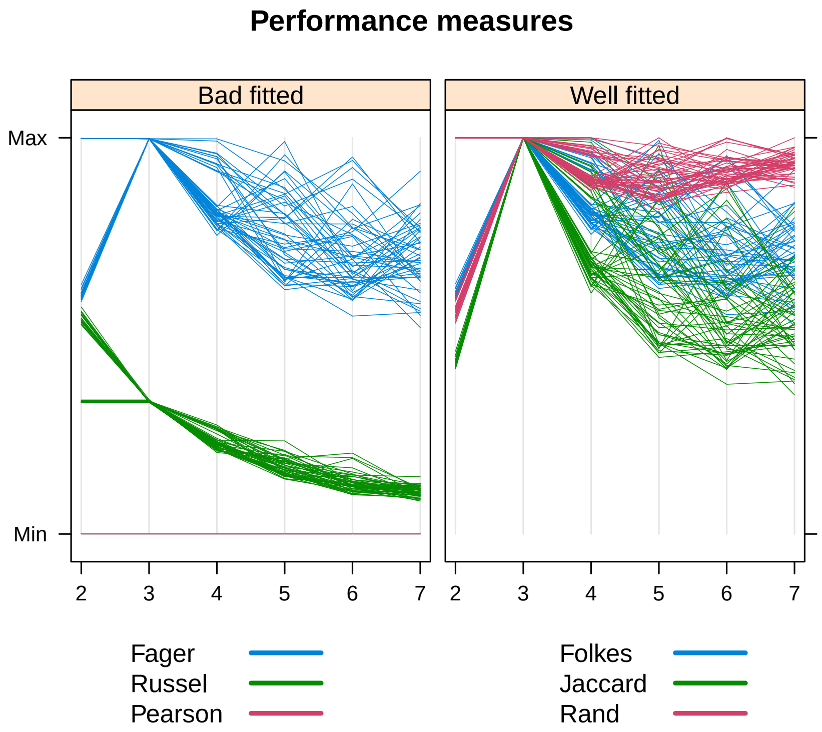


Figure 3.4: Selection of six performance measures, which are bad such as well fitted for the generated data in a three cluster solution. Illustrated are the validation indices computed by a new sampled dataset with the standard deviation of 0.3.

indices:

$$Russel = \frac{a}{a+b+c+d} \qquad Rand = \frac{a+d}{a+b+c+d}$$

Pearson Index

The *Pearson Index*, as the next validation index, is also achieving poor results for proposing the correct number of clusters. A better fit for the assignment with nearly the same formula is announced from the *Gamma Index*, in which the denominator differs in extracting the square root. Definitely, this results better solutions for the proposed number of clusters, such as ranges in the same limits of the validation index. Shown in a case study below, the example illustrates the better-working measurement:

$$Pearson = \frac{ad-bc}{(a+b)(a+c)(d+b)(d+c)}$$

$$Gamma = \frac{ad-bc}{\sqrt{(a+b)(a+c)(d+b)(d+c)}}$$

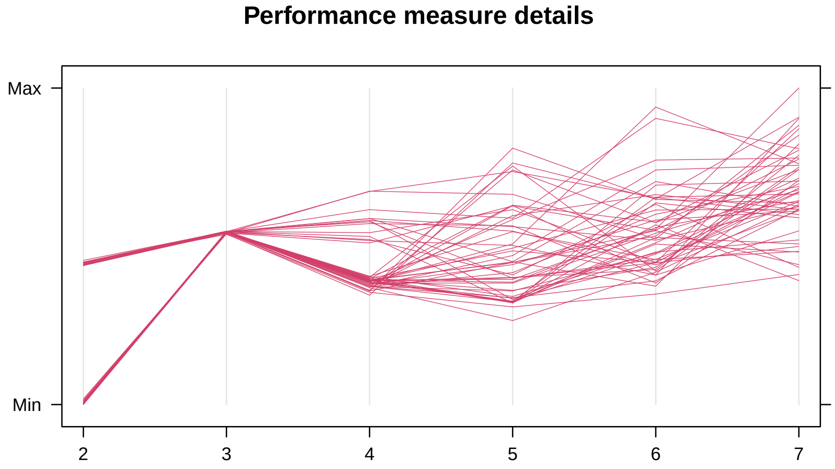


Figure 3.5: Detailed view to the Pearson Index of the previous provided figure 3.4

1. $a = d = 100$ and $b = c = 0$:
   $Pearson = 100^2/100^4 = 1/10000 \approx 0$
   $Gamma = 100^2/\sqrt{100^4} = 1$
   Here, a and d is computed with the same values, but in practice d results much higher values than a, caused by the random selection of cluster denotation by the algorithm. To simplify the case study, d is the same value as a.

2. $a = d = b = c = 100$:
   $Pearson = (100^2 - 100^2)/100^4 = 0$
   $Gamma = 0$

In fact the *Gamma Index* did not reach its maximum value of 1, mostly their results are below, but this index is almost better than *Pearson Index*.

Fager Index

The *Fager Index* fits almost the proposed cluster solution, but the index is weak for the proposed number of cluster in some cases. The *Folkes Index* with nearly the same formula, is not penalized by the second term. Therefore, this validation index is well-performed for finding the correct number of clusters.

$$Fager = \frac{a}{\sqrt{(a+b)(a+c)}} - \frac{1}{2\sqrt{a+b}} \qquad Folkes = \frac{a}{\sqrt{(a+b)(a+c)}}$$

3.2 Discussion of the Internal Validation Performance Measures

The design of this simulation study is very similar to the study of the external validation indices. In difference to the external ones, the internal validation indices should not only be maximized to propose the correct number of clusters. For the proposal of clusters the computing is slightly intricate, because the difference between the number of clusters has to be reviewed. The differences are an important tool to indicate the number of clusters. As seen in section 2.4.2, the first difference such as the second difference has to be determined.

3.2.1 Proposed Number of Clusters Resulted by a Three Cluster Solution

In order to the first example of the external validation indices, Tables 3.4, 3.5 and 3.5 shows the proposed number of clusters by their maximization or minimization criteria. Here again, the maximum or minimum values are accumulated for proposing the number.

	2	3	4	5	6	7
silhouette	0	100	0	0	0	0
dunn	0	100	0	0	0	0
db	0	100	0	0	0	0
connectivity	54	46	0	0	0	0

Table 3.4: Amount of the values, which should indicate the three cluster solution as the true number of clusters with the respect to their minimization or maximization criteria. Shown are the results of the predicted generated data with standard deviation 0.3.

In particular, in Table 3.4 the validation index is either maximized or minimized regarding its computed value. The *Silhouette*, the *Dunn* and the *Davies Bouldin Index* are performing the true numbers of clusters very well, while the *Connectivity Index* locates a lower number of clusters as the true one. The fluctuation of the measurement might indicate that only minimization is not well-fitting for the true number of clusters in these generated data sets.

In Table 3.5, the first difference between each cluster solution is shown. As announced in section 2.4.2, *Scott* such as *Friedman* should be maximized at the left side, i.e. if the maximum of the first difference lies between the two and three cluster-solution, i.e the three cluster solution should be chosen as the proposed number of

	2 to 3	3 to 4	4 to 5	5 to 6	6 to 7
friedman	86	0	0	6	8
ratkowsky	100	0	0	0	0
scott	100	0	0	0	0

Table 3.5: Amount of the values, which should indicate the three cluster solution as the true number of clusters with the respect to their first differences. Shown are the results of the predicted generated data with standard deviation 0.3.

clusters. Or rather, these validation indices are locating the true number of clusters in the generated data. Indeed, the same solution can be seen in each other generated data set. While maximization these validation measures on the left side, the *Ratkowsky Index* shall locate its maximum difference at the right side. Seen in this example such in each other one, this validation index has the high amount of the maximum values at the left side. This suggests that the choice of these values to the right side could be wrong.

	3	4	5	6
ball	100	0	0	0
marriot	100	0	0	0
xuindex	100	0	0	0
tracew	100	0	0	0
trcovw	0	27	28	45
rubin	93	0	0	7
calinski	100	0	0	0
hartigan	100	0	0	0

Table 3.6: Amount of the values, which should indicate the three cluster solution as the true number of clusters with the respect to their second differences. Shown are the results of the predicted generated data with standard deviation 0.3.

Last but not least, the measurements which either maximize or minimize their values at the second difference is shown in Table 3.6. Most of these validation indices find the high amount of the values at the true numbers of clusters. *Rubin* seems to be weaker than the other validation measures, but still minimization the second difference fits the proposed number of clusters correctly. However, $trace(\text{cov}(W))$ breaks the ranks. This validation index seems to bop around the proposed number of clusters announced by the algorithm. In other calculations of this index in each data set, the index is acting similar.

3.2.2 Proposed Number of Clusters Resulted by a Eight Cluster Solution

The next example as well in order to the declaration of the external validation indices, in which is shown the eight clustered solution in Tables 3.7, 3.8 and 3.9. The validation indices are computed with the respect to prediction to the new sampled data with a standard deviation of 0.2. Such a standard deviation represents a well-separated data set, i.e. the space between the clusters should be far enough for the internal measurements. As in the previous example the amounts are accumulated, in which the measurement shall be minimized or maximized in respect to their criteria, i.e. the measurement itself, the first difference or the second difference.

	4	5	6	7	8	9	10	11	12
silhouette	0	0	0	0	100	0	0	0	0
dunn	7	3	0	1	73	7	4	4	1
db	0	0	0	0	100	0	0	0	0
connectivity	22	1	0	0	77	0	0	0	0

Table 3.7: Amount of the values, which should indicate the eight cluster solution as the true number of clusters with the respect to their minimization or maximization criteria. Shown are the results of the predicted generated data with standard deviation 0.2.

In details, Table 3.7 shows the cumulated amounts, in which the validation indices should be either maximized or minimized regarding its definition. Particularly, noticeable is the well-performing of the *Connectivity Index*, whereas less clusters are indicated as the correct number, as in the previous example.

	4 to 5	5 to 6	6 to 7	7 to 8	8 to 9	9 to 10	10 to 11	11 to 12
friedman	3	6	2	89	0	0	0	0
ratkowsky	12	20	31	37	0	0	0	0
scott	9	12	13	66	0	0	0	0

Table 3.8: Amount of the values, which should indicate the eight cluster solution as the true number of clusters with the respect to their first differences. Shown are the results of the predicted generated data with standard deviation 0.2.

In Table 3.8, the indices are referenced, which has to be maximized in the respect to their first differences. *Scott* and *Friedman* are indicating the true number of clusters. *Scott* gives weaker values than in the three-cluster solution, but altogether the highest amount of these values are indicating the correct solution. *Ratkowsky* shows the same weak solution as in the previous example. As well, the same problem arise, i.e. the indication of the number on the right side are untouched and in particular in column notation ”8 to 9”, whereas the first difference should be maximized.

	5	6	7	8	9	10	11
ball	91	8	0	1	0	0	0
marriot	13	17	6	64	0	0	0
xuindex	0	0	0	100	0	0	0
tracew	0	0	0	100	0	0	0
trcovw	32	20	2	0	7	25	14
rubin	0	0	0	100	0	0	0
calinski	0	0	0	100	0	0	0
hartigan	0	0	0	100	0	0	0

Table 3.9: Amount of the values, which should indicate the eight cluster solution as the true number of clusters with the respect to their second differences. Shown are the results of the predicted generated data with standard deviation 0.2.

Last, Table 3.9 is visualizing the performance measure in respect to their second differences. The measurement *trace(cov W)* is as well not indicating the true number of clusters as it should propose regarding minimization the second difference. Not a value is locating the artificial generated clusters. Such as the index before, *Ball* is badly fitting yet, instead as proposed in the lower dimensional example. Nearly all of the values are proposing a five cluster solution as the true number of the groups. This could be due to the fact, that the generated data is not well performing for the validation measurement. All of the other validation measurements are indicating the true number of clusters, but it seems that the *Marriot Index* is weaker in higher dimensions than in the lower ones. In summary, only five validation indices are locating the correct number of clusters by their second differences.

3.2.3 Proposed Number of Clusters Resulted by a 32 Cluster Solution

Such in order to the results of the external validation measures, of course, the internal validation indices have to be interpreted for the 32 clustered solution. The following tables show the maximum or minimum values in order to their definition. At this time, the results are only providing a relatively well-separated data set, i.e the validation indices are computed on the basis of the original generated data with a standard deviation of 0.2. Trough better results of this standard deviation, other examples are disregarded for analyzing these indices.

	28	29	30	31	32	33	34	35	36
silhouette	0	2	14	20	30	25	9	0	0
dunn	64	28	6	0	0	0	2	0	0
db	0	0	4	3	14	33	46	0	0
connectivity	60	34	6	0	0	0	0	0	0

Table 3.10: Amount of the values, which should indicate the 32 cluster solution as the true number of clusters with the respect to their minimization or maximization criteria. Shown are the results of the predicted generated data with standard deviation 0.2.

In Table 3.10, the resulted validation indices give an overview to the true values in order to their criteria. Most of the *Silhouette Indices* are at the correct cluster solution and *Davies-Bouldin* has most values around the true number of clusters. But *Connectivity* and *Dunn* are proposing a lower number of clusters and differs in its solution. However, keep in mind that the results reflecting the search of the number in a high-dimensional data set with many clusters. Thus, *Silhouette Davies-Bouldin*, which are either maximized or minimized in their computed values, are working well in these cluster patterns.

	28/29	29/30	30/31	31/32	32/33	33/34	34/35	35/36
friedman	19	17	20	15	18	11	0	0
ratkowsky	3	11	9	15	34	28	0	0
scott	22	26	15	13	16	8	0	0

Table 3.11: Amount of the values, which should indicate the 32 cluster solution as the true number of clusters with the respect to their first differences. Shown are the results of the predicted generated data with standard deviation 0.2.

Not as good as provided before, the measurements on minimization their first differences are disabled to find the true number of clusters, seen in Table 3.11. *Scott* and *Friedman* are providing most of the maximum in their first difference for the lower number of clusters, as it would be correct. As the proposed number of clusters, these indices are indicating the real number as 30 clusters. Not as bad for such high number, but however, better results are given in the examples before. *Ratkowsky*, as the last of these three, differs a lot in the maximum value of its first difference. This index has to provide the cluster solution on the right side, but indeed the maximum difference are equally shared between the correct solution. Eventually, these indices indices are less informative for proposing the true number of the generated clusters.

Last but not least in discussion of the internal validation indices, Table 3.12 shows

	29	30	31	32	33	34	35
ball	30	34	16	10	10	0	0
marriot	50	38	11	1	0	0	0
xuindex	9	17	16	30	28	0	0
tracew	20	23	18	22	17	0	0
trcovw	23	22	28	18	9	0	0
rubin	3	8	16	38	35	0	0
calinski	5	12	18	34	31	0	0
hartigan	9	17	16	30	28	0	0

Table 3.12: Amount of the values, which should indicate the 32 cluster solution as the true number of clusters with the respect to second differences. Shown are the results of the predicted generated data with standard deviation 0.2

the overview to the performance measures, in which either the maximum or minimum value of their second differences has to be reached. Surprisingly, the *Xu*, *Rubin*, *Calinski*, and *Hartigan Index* are able to find the correct number of clusters in their second difference. Not as bad as announced before, the *trace(cov W)* illustrates better results in the higher level solution. Nearly, most minimum values are at the correct number of clusters; however, the distinction between 32 and 33 clusters as the true one in such dimensions is well-fitting for these kind of data. *Ball* provides as well bad predictions for the generated cluster patterns. Weaker than in previous examples, the *Marriot Index* turns to a lower number of clusters than the other validation measures.

3.2.4 Problems of Several Validation Measurements

Through the computation of the internal validation indices, some weakness follows in the performance measures from the generated data. In the next part, these problems are announced and if appropriate, it will be solved.

Ratkowsky

The *Ratkowsky Index* is not working very well in this Benchmark study. This validation index provides acceptable results, but once in a while the validity index gives a `NaN` as a result. Figure 3.6 shows an example, in which this validation index provides useless results. Represented is a cluster solution with two cluster centers in a data set generated with seven certain groups. This figure visualizes the problem, but does not appear in the benchmark study anymore. The groups are lying around zero, as the center point for the generated data. Here, the data represents separated cluster with noise inside. If these groups are underrated, i.e. the cluster algorithm choses less groups than in reality exists, then the validation index performances the group assignment not as requested. Major possibility is announced as follows. In

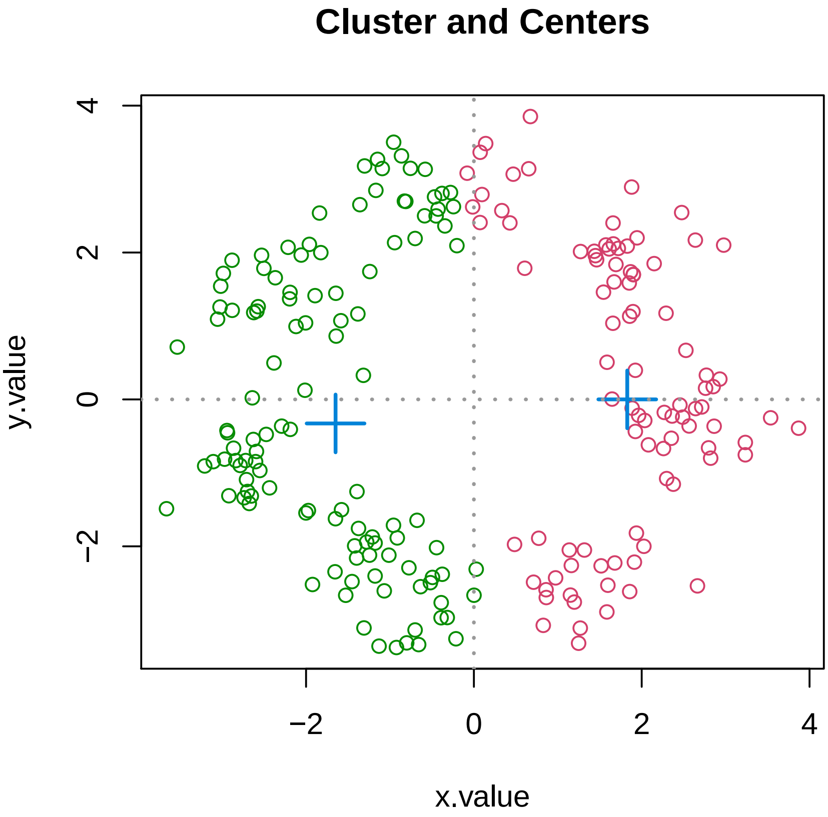


Figure 3.6: Officially 7-cluster solution with assignment in two clusters with their corresponding centers

an example, the cluster algorithm divides the data set in two individual groups on a straight line with $x = 0$ and the y-value of the center points in each cluster is zero. The x-values of the center points are widely spaced. Then, the validation index is taken the within sum-of-squares of each variable, shortly varSSW. Afterwards, the column sum of each within sum-of-squares is calculated. The varSSW of the y-value is rather small, whereas the varSSW of the x-value is larger in relation. Furthermore, the total sum-of-squares of each variable, shortly varSST, is computed by the squared sum of the distance to the overall mean in the whole data set. Hence, the varSST is nothing more than the Euclidean distance to the mean of the variable. For computing the *Ratkowsky Index*, the varSSB is demanded, which is simply the difference of varSST to the varSSW, i.e. varSST = varSSW + varSSB. Thus, it occurs that the varSSW of one variable can be larger than the varSSW of the same variable, and the computed varSSB can be a negative value. The conclusion for this inconsistency in the validation index is the accidentally sensitivity to the generated data. The main problem for this validation index is the generated data set around zero and only robust for more noisy data in the same area. On the other side, it is a general problem, if the centers of each variable have nearly the same value around the zero point. Such a validation index should be independent to the data sets, but indeed the *Ratkowsky Index* is improper for validation.

	2d-norm	2d-corn	3d-corn	5d-corn
original generated data	6	263	21	2
bootstrapped data	22	243	35	5
out-of-bootstrap data	64	315	21	8

Table 3.13: Sum of NA returning values over all results

Due to the announced problem, this validation index cannot be computed, caused through the abort of the computation. In such cases, the implementation of this validation index is caught by a function, which returns NA instead of abortion. Table 3.13 shows the results, how often NA is counted for every data set in this empirical benchmark experiment. Most NA-values are produced in the four-grouped cluster solution. It is produced, while the k-means algorithm investigated two-clusters as the best solution for the four-cluster data. The center points are located totally between these two clusters in well-separated data sets, as seen in the earlier mentioned example.

SD	0.1	0.2	0.3	0.4
original generated data	100	89	67	7
bootstrapped data	92	78	63	10
out-of-bootstrap data	94	96	93	32

Table 3.14: Sum of NA returning values for the several standard deviation in the two-dimensional cubed scenario

Table 3.14 shows the sum of the produced NA's in details. Here, the most of the NA-values are produced for the well-separated data, while less of these values appear in the noisy data sets. Main reason for the announced problem can be, that the centers are in empty areas, completely between the clusters, because most NA's were produced in the well-separated data. In such a case, both center points in the variable are located nearly at the same value. Thus, the above announced problem occurred. For noisy data, i.e. by increasing the standard deviation, the problem disappears by degrees.

Connectivity

The focus is on the *Connectivity Index*, which gives bad results for the number of clusters by its minimization criteria. As seen in the previous sections, this validation index did not reach most values of its minimum rank for the proposed number of clusters.

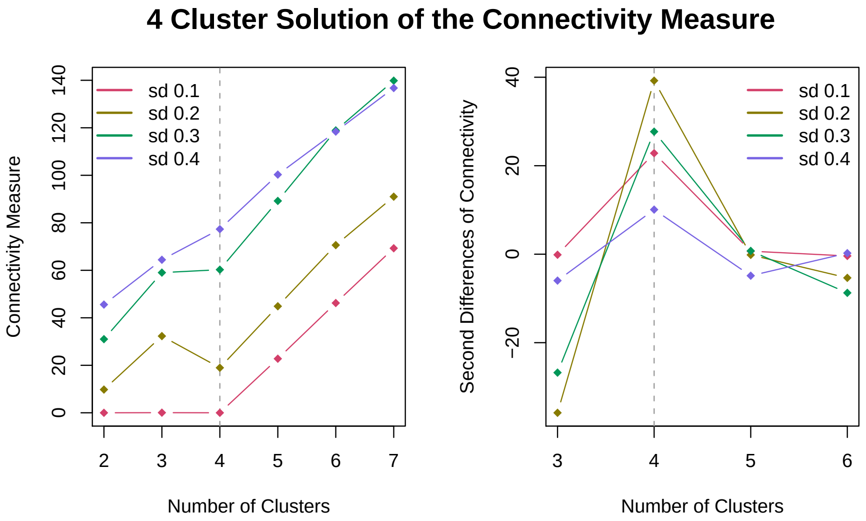


Figure 3.7: Computation of the Connectivity Index on the four cluster solution of the two-dimensional original data with different standard deviation.

In Figure 3.7 on the left side, the average value for each proposed number of clusters is shown. Due to the high amount of the validation index, the average value is taken. On this reason, the visualization is better working to compare this validation measure for certain standard deviations of the generated data in the four cluster solution. The validation index is computed to the corresponding original generated data, and this validation index is performing similar to the computation regarding each other sampled data, i.e. to sampled or out-of-bootstrapped data. Hence, a conclusion cannot suggested in the respect to the minimization criteria. Noticeable, it seems that the performance measure has a sharp curvature at the true number of cluster. At least, this is shown for the well-separated data, visualized with the standard deviations of 0.1 and 0.2. Therefore, the second difference should be reach its maximum value for the highest curvature. Thus, the second difference measures a positive "elbow" caused to the original minimization criteria. Such as for the well-separated data, the noisy generated data also measures a positive "elbow" for the validation measurement.

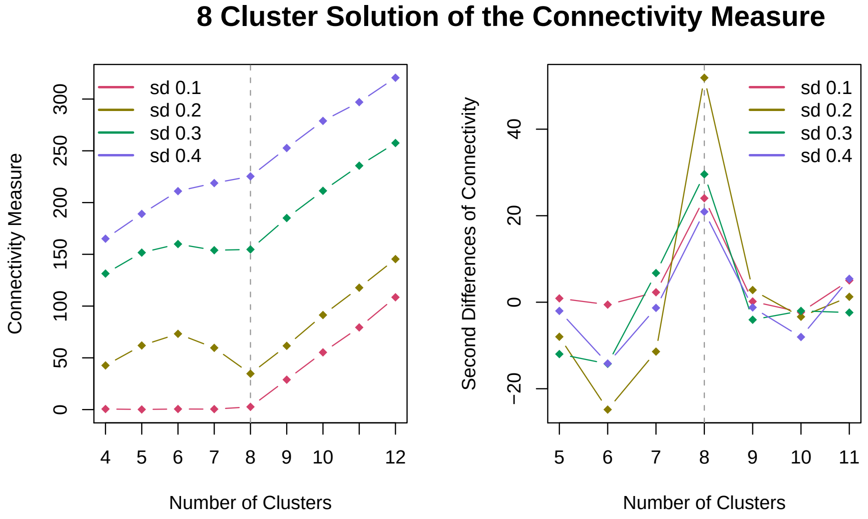


Figure 3.8: Computation of the Connectivity Index on the eight cluster solution of the three-dimensional out-of-bootstrapped data with different standard deviation.

Likewise as suggested in Figure 3.7, the eight cluster solution is given the same solution, comparisons to this are shown in Figure 3.8. Hence, the second difference has to be maximized as well to reach its positive ”elbow”. In this example, the performance measure is computed regarding the out-of-bootstrap data. The results are not changing significantly on the other generated data, such as the original data and the new sampled one. The best fit for the internal validation index gives the generated data with the standard deviation of 0.2, represented by the high peak in the illustration. However, each of the standard deviations gives an obvious peak for the true number of clusters.

Trace(cov W)

Next, the internal validation index *trace(cov W)* is badly fitting to search the correct number of clusters. The identification for minimum of the second difference could not be the best solution, i.e. the validation index provides different solutions for the proposed number of clusters.

On the left side of Figure 3.9, there is seen the average of the original computed validation measure on certain standard deviations. Furthermore, the eight cluster solution on the predicted sample data is visualized in the figure. The right side is showing the second difference of the measurement, in which the validation index should be reach its minimum value for the correct number of clusters. The second difference of the validation index tries to get a solution in order to the elbow crite-

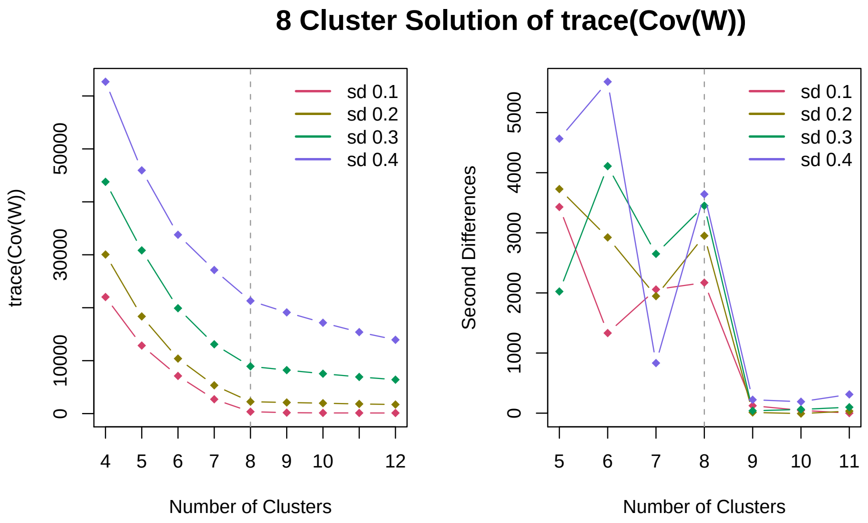


Figure 3.9: Computation the average of trace(Cov(W)) on the eight cluster solution of the three-dimensional new sampled data with different standard deviation.

ria. Thus, the elbow is not identifying the true clusters, i.e. the local minimum of the difference has not been found in the artificial generated data. In fact, the right side of the figure shows that the index is delayed in locating the correct number of clusters. As a reason for that, minimization the second differences could be useless as the correct criteria for such an index.

Ball Index

The *Ball Index* is such badly proposing for the number of clusters as the previous performance measures. Likewise, this validation index has to reach its maximum value at the second difference, but is not indicating the correct number of clusters in the three-dimensional scenario.

In Figure 3.10, the eight cluster solution of the validation index is shown on the left side. To verify the solutions, the figure shows the validation index for the prediction on the original data set on several standard deviations. Due to the similarity of this index with *Connectivity* in regard to their trend, the second difference has to be visualizing for comparing the solutions. On the right side of Figure 3.10, it can be seen that the maximum values of the second difference lies for slightly number of clusters. Nevertheless, the local maximum of these differences are indicating the correct number of clusters, i.e. the positive ”elbow” is around the correct number of clusters. In fact, only counting the maximum of second differences cannot solve the cluster identification problem. Thus, visualization of the results is very helpful to

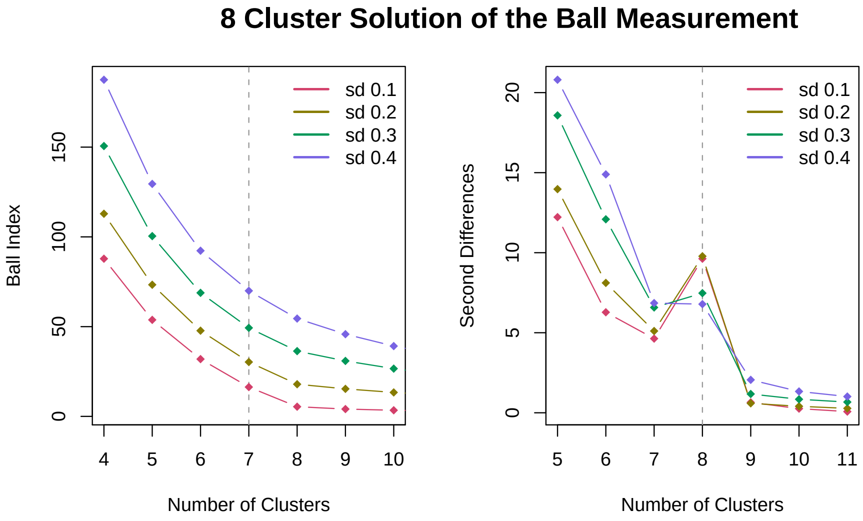


Figure 3.10: Computation the average of the Ball Index on the eight cluster solution of the three-dimensional new sampled data with different standard deviation.

get an idea of the correct number of clusters.

Chapter 4

Testing and Modelling

In this analysis, a model is searched for getting the correct number of clusters in dependence of the validation indices, the used sample for computing the performance measurement, the standard deviation and the dimension. Therefore, the question of such an analysis is to find out the parameter for the influence of these factors. Hereby, in such an order, a logit model can be used to predict the probability for finding the correct number of clusters by each index.

4.1 Data Preparation

Before analysing, the data has to be prepared for modelling. Firstly, each index has to be evaluated regarding their maximization or minimization criteria, either in its computed value, first or second differences. Afterwards, the true number of clusters have been retained in the prepared data, i.e. other proposed number of clusters by the validation indices are cut for modelling. Other cluster solutions cannot be reviewed in the analysis, because the true number of clusters are changing in every dimension. Following, the included factors of the prepared data are listed below:

1. *Success for correct clusters found:* This value gives the success, that the true number of cluster is found by each criterion, i.e. 0 stands for finding other number of clusters and 1 stands for finding the correct number of clusters.

2. *Validation Index:* The name of each index is given, i.e. in total, there are 35 certain validation indices in the data for modelling the success for finding the true number of clusters. Furthermore, there is no distinction for external or internal validation indices.

3. *Data sample for computing the index:* Each validation index is computed by several data samples, i.e. from the original generated data (*orig*), a new bootstrapped data (*sam*) and the out-of-bootstrapped data (*oob*).

4. *Dimension of the generated data set:* The data sets are generated in three certain dimensions. For modelling, the Gaussian distributed clusters with the centers in each corner of a hypercube are used. Hence, there are the two-dimensional (*2dcorn*), the three-dimensional (*3dcorn*), and the five-dimensional generated data sets (*5dcorn*).

5. *Standard deviation of the generated data:* The standard deviation are in a grid from 0.1 till 0.4. For different deviations are used to show the difference from well-separated to completely noisy data.

A better understanding of the last two items was given in section 2.1.1. Furthermore, the data is assumed from the simulation and finally structured for the analysis, seen in Figure 4.1:

	success	name	sample	sd	dimension
1	0:52793	ball : 3600	orig:30000	0.1:22500	2dcorn:30000
2	1:37207	calinski : 3600	sam :30000	0.2:22500	3dcorn:30000
3		connectivity: 3600	oob :30000	0.3:22500	5dcorn:30000
4		db : 3600		0.4:22500	
5		dunn : 3600			
6		friedman : 3600			
7		(Other) :68400			

Table 4.1: Structure of the data for analysing

A detail also ought to be mentioned. Certain internal validation indices are producing `NA`'s. Previous sections discussed some reasons for such purposes. Due to the reason, the maximum or minimum validation indices are ambiguous regarding their criteria. The value in the variable *success* is zero in such a situation.

4.2 Modelling and Results of the Logit Model

The dependent variable in this model is *success*, which indicates the true number of clusters. As announced, this variable is either *zero*, if other numbers of clusters are found or *one*, if the true number corresponds to the proposed number of clusters, given by the validation index. One of the independent variable is *name*, which reflects each index. Another independent variable is *sd*, which is nothing more than the standard deviation of each single data set, whether *dimension* is chosen. The squared value of *sd* is used as well to get more flexibility in the model. Last but not least, the *sample* is another parameter in the model.

Thus, the probability for finding the correct number of clusters can be given by the formula according to the definitions of Fahrmeir et al. (2007):

$$\begin{aligned}\pi_i &= P(success_i = 1 \mid name_{i1}, sample_{i2}, sd_{i3}, sd_{i4}^2, dimension_{i5}) \\ &= \frac{\exp(\eta_i)}{1+\exp(\eta_i)}.\end{aligned}$$

Hence, the linear predictor is given by

$$\eta_i = \beta_0 + \beta_1 name_{i1} + \beta_2 sample_{i2} + \beta_3 sd_{i3} + \beta_4 sd_{i4}^2 + \beta_5 dimension_{i5}.$$

4.2.1 Discussion of the Logit Model

Table 4.2 shows the estimators of each coefficient in the logit model. The standard deviation is highly significant and its behaviour is completely different to the reference category with the standard deviation of 0.1. Indeed, the linear term of the standard deviation is positive, but however, the quadratic term is negative. Hence, the quadratic term prevails the linear one and a higher standard deviation implies bad results for finding the correct number of clusters. Also, a higher dimension of the generated data sets implies worser results as in the two-dimensional reference category, due to highly significance of the variable *dimension*. In higher dimensions, the prediction for the success of finding the correct number is considerably more difficult than in lower dimensions. This dimension effect, such as the effect of the standard deviation can be already seen in the previous sections. Unlike as expected, the sample method *oob* makes no difference as the sample *orig*. The new bootstrapped sample is differently, but however not that highly significant to the reference category.

Nevertheless, most of the external validation indices behaves similar to the *Czekanowski Index*, which is classified as the reference category. As already shown in the explorative analsyis, three of the external indices are acting different to the reference category, in particular, the *Fager*, the *Pearson* and the *Russel Index* are highly significant and the negative prefix of the estimators implies a worser result for finding the correct number of clusters in the benchmark study. All of the internal indices are acting different to the reference category, seen in the statistically significance of the variables. The negative prefix of *Connectivity*, *Ratkowsky*, *Ball* and *trace(cov W)* implies worser predictions for the success of finding the true number of clusters; and these internal validation indices are already known by their weaknesses. Notable, the estimators of *Scott* and *Silhouette* are covered as well with a negative prefix. The negative value of *Scott* can be explained with some non-produced values. In particular, some values are covered with `NA`'s due to the implementation of the index in the package `cclust`, e.g. by finding empty clusters in the sample method. Then, the first differences cannot be calculated and finding the true number of clusters is proving to

	Estimate	Std. Error	z value	Pr(>|z|)
(Intercept)	0.2231	0.0857	2.60	0.0092
nameFager	-0.5936	0.0878	-6.77	0.0000
nameFolkes	0.0000	0.0893	0.00	1.0000
nameGamma	0.1329	0.0897	1.48	0.1387
nameGower	0.2065	0.0900	2.29	0.0218
nameHamann	0.2065	0.0900	2.29	0.0218
nameJaccard	0.0000	0.0893	0.00	1.0000
nameKruskal	0.2519	0.0902	2.79	0.0052
nameKulczinski	0.0000	0.0893	0.00	1.0000
nameMcConnaughey	-0.0080	0.0893	-0.09	0.9289
namePearson	-1.9101	0.0906	-21.09	0.0000
namePeirce	0.1451	0.0898	1.62	0.1061
nameRand	0.2065	0.0900	2.29	0.0218
nameRoger	0.2065	0.0900	2.29	0.0218
nameRussel	-3.1375	0.1088	-28.84	0.0000
nameSokal1	0.1492	0.0898	1.66	0.0967
nameSokal2	0.0000	0.0893	0.00	1.0000
nameSokal3	0.1410	0.0898	1.57	0.1163
nameWallace1	0.0721	0.0895	0.81	0.4204
nameWallace2	0.0160	0.0893	0.18	0.8582
nameball	-2.6538	0.0832	-31.89	0.0000
namecalinski	1.4742	0.0808	18.24	0.0000
nameconnectivity	-2.5423	0.0822	-30.93	0.0000
namedb	0.4220	0.0782	5.39	0.0000
namedunn	-0.7172	0.0764	-9.39	0.0000
namefriedman	0.6626	0.0788	8.41	0.0000
namehartigan	0.9259	0.0795	11.65	0.0000
namemarriot	0.4711	0.0783	6.01	0.0000
nameratkowsky	-4.4214	0.1227	-36.04	0.0000
namerubin	1.2418	0.0802	15.48	0.0000
namescott	-4.5998	0.1303	-35.31	0.0000
namesilhouette	-3.4352	0.0943	-36.44	0.0000
nametracew	0.6120	0.0787	7.78	0.0000
nametrcovw	-4.3266	0.1190	-36.36	0.0000
namexuindex	1.5928	0.0812	19.63	0.0000
sd	18.5394	0.5171	35.85	0.0000
I(sd^2)	-48.7077	1.0255	-47.50	0.0000
samplesam	-0.0577	0.0244	-2.37	0.0180
sampleoob	-0.0036	0.0244	-0.15	0.8836
dimension3dcorn	-0.4165	0.0224	-18.58	0.0000
dimension5dcorn	-3.9888	0.0307	-130.11	0.0000

Table 4.2: Logit-Estimations

be difficult regarding its maximization criteria. Through the statistically significance and positive estimators of *Calinski*, *Rubin* and the *Xu Index*, these validation indices provides better results for the success as the reference category. These validation indices were already discussed and made a positive impression in the previous sections.

Interpretation of the Probabilities in the Logit Model

For better overview of the logit model, the probabilities by the prediction are given for each dimension. In the following figures, the probabilities are shown for the success of finding the true number of cluster divided by the standard deviation in each dimension.

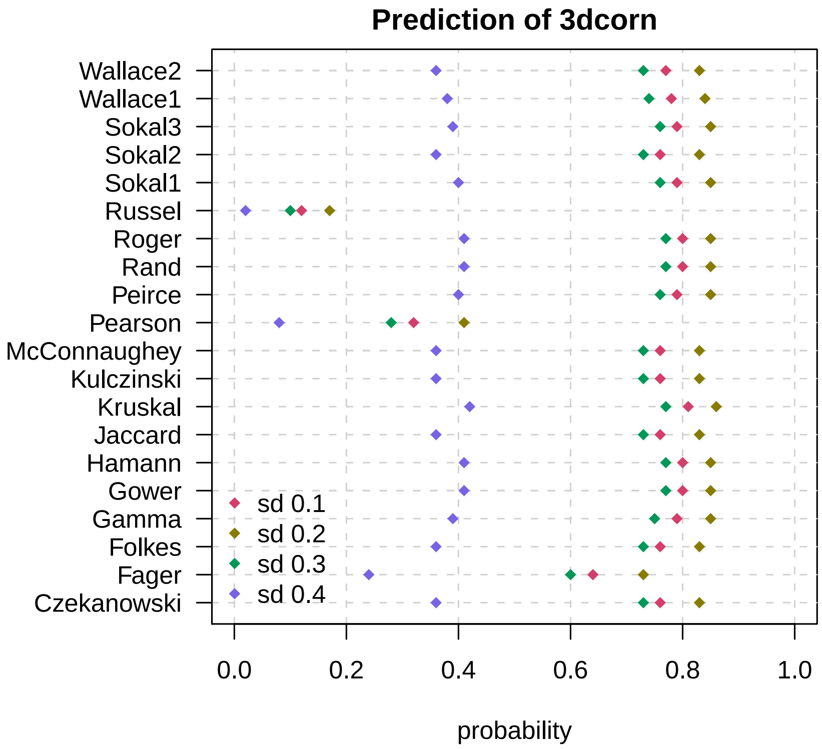


Figure 4.1: Prediction of the probabilities for success in finding the correct cluster solution of the external validation indices divided by the standard deviation.

Due to the similarity in the results of the probabilities for the two- and the three-dimensional solution, Figures 4.1 and 4.2 show the probability of the three-dimensional data set. Here, eight clusters should be found as the correct number of clusters concerning their certain standard deviations. Firstly, the success of finding the correct number attempts better with a higher standard deviation, but indeed, through the quadratic term, the probability drops rapidly by noisier data structure. As announced, nearly all external indices implies good results, except the mentioned ones in the previous section. *Calinski*, *Rubin* and *Xu*, as representatives for well-fitting internal validation indices, provides better results as the mentioned internal indices with the negative prefix.

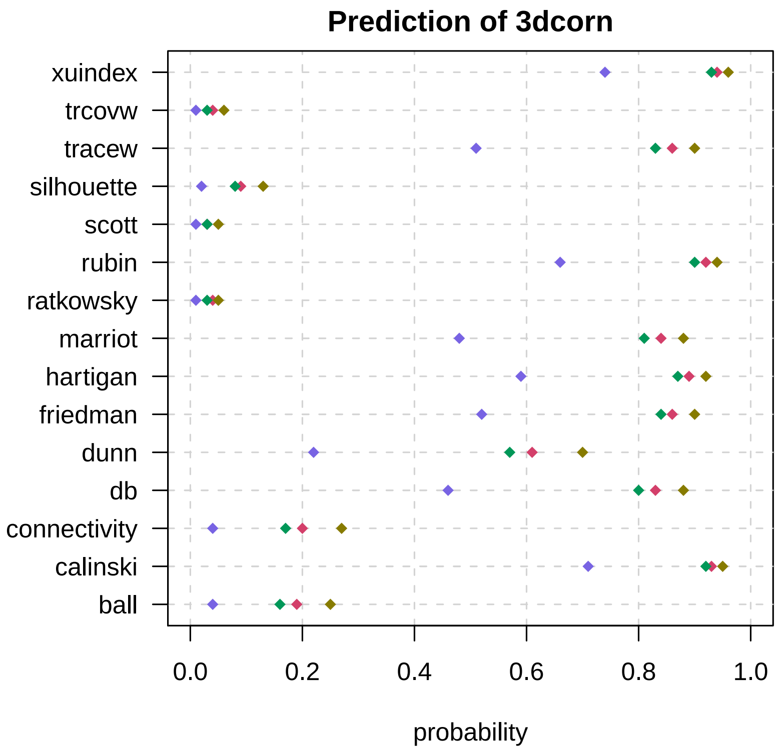


Figure 4.2: Prediction of the probabilities for success in finding the correct cluster solution of the internal validation indices divided by the standard deviation.

Last, the probabilities of the five-dimensional data set is given in Figures 4.3 of the external indices and Figure 4.4 of the internal ones. Such as in the example above, the indices are divided by certain standard deviations, i.e. from well-separated data to completely noisy ones. Here, the probabilities decreased in its values by increasing the dimensions. The proposed number of clusters would be 32, but however in such dimensions, finding the true number of clusters is very difficult. The external validation indices are acting similar to each other. Finding the correct number is nearly impossible. Due to higher dimensions, the external indices are not that good as in lower dimensions. The three representatives of the internal indices are providing better results according their probabilities. Indeed, these indices are the weaker as in the three-dimensional data set, but their probability lies between 0.3 and 0.4. These indices the best-fitted in the benchmark study, even in such high dimensional data sets.

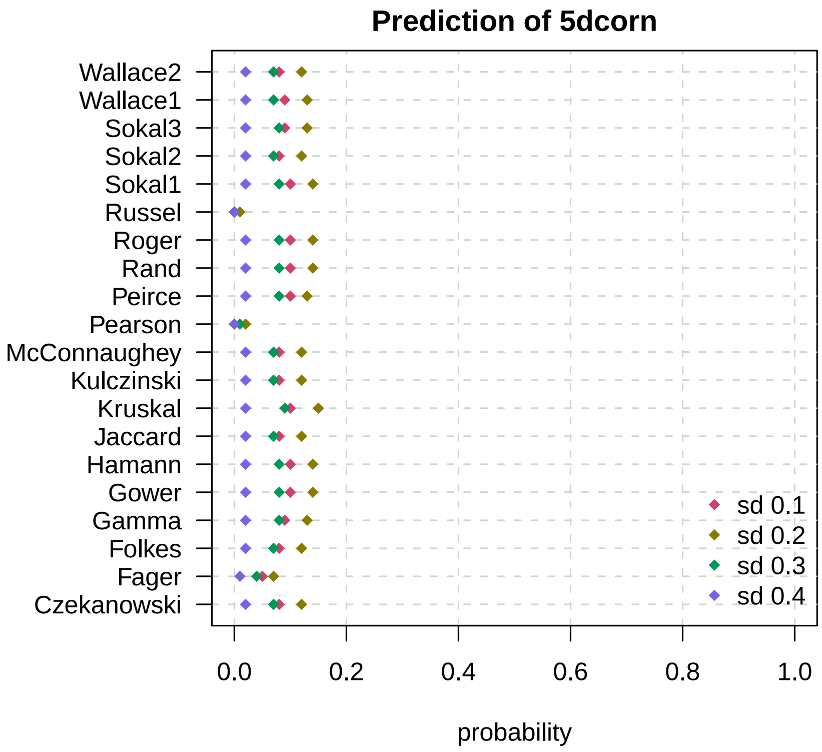


Figure 4.3: Prediction of the probabilities for success in finding the correct cluster solution of the external validation indices divided by the standard deviation.

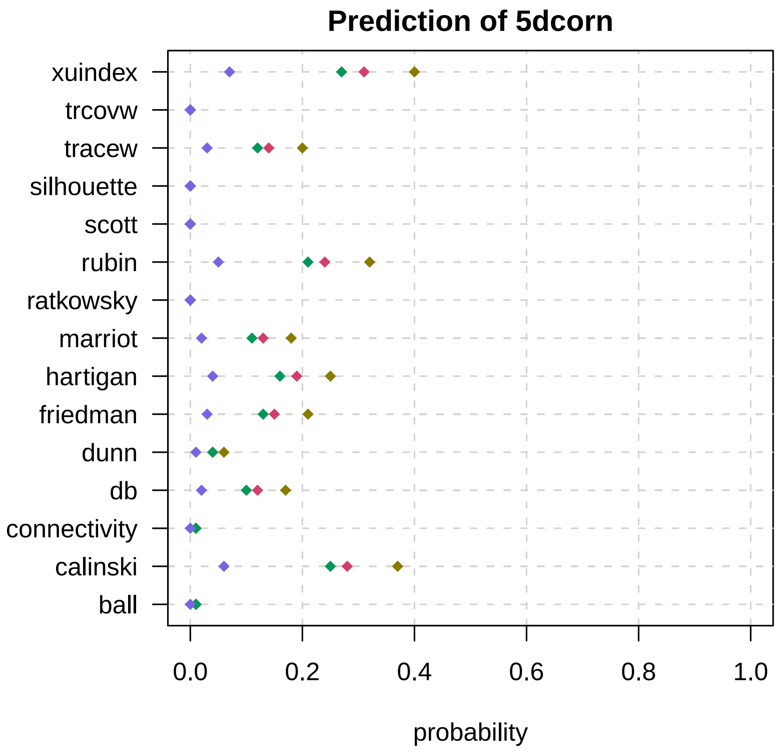


Figure 4.4: Prediction of the probabilities for success in finding the correct cluster solution of the internal validation indices divided by the standard deviation.

Chapter 5

Implementation in R

The first part in this chapter shows the changes, which was done to run the benchmark experiment of the internal indices. Afterwards, the implementation of the validation functions is described. In this package are implemented the external and internal validation indices, which were used during the simulation study.

5.1 Implementation of the Internal Indices in R

The first modification of the function clustIndex exists in the pass of the training data set. In this implementation, however a function to predict the new cluster assignments of the data items was not provided in the package. For such a benchmark experiment, an implementation should exist for predicting the data points to a particular cluster. The training data for the algorithm, which is not the same as test data, provides problems in the computation of these internal indices. This problem is solved in the implementation through the function predict of the R package flexclust. Furthermore, clustIndex needs objects generated by S3 class systems, while functions of the package flexclust are providing objects in the new S4 class system. These methods does not match among each other. Further problems are listed below.

5.1.1 Sum-of-Squares Within

The first problem in the implementation of clustIndex appears in the pass of the sum-of-squares within the clusters by the cluster object. These sum-of-squares within the clusters are not the same as in the training data set and has to be computed by its own. However, this function already existed in clustIndex, but was untouched the certain internal validation indices. The sum-of-squares should handed in by the cluster object for the particular algorithm, which is indeed wrong. Due to the different size and data points of the test data set, NA's were produced and the function maybe interrupt. In order to circumvent this, the already implemented function is attached into clustIndex, and the sum-of-squares corresponds to their cluster size. The cluster centers remains in their original computation by the cluster algorithm.

5.1.2 Distance Matrices

Further difficulties occurs in the second group of the performances measures, announced previously in section 2.4.2. In certain cases, it appears that the clusters exists only of a few data points which are assigned to this particular cluster. In the new predicted cluster assignment for a test data set (e.g. the out-of-bootstrapped data), several clusters are allocated with non or at least one data point. Through the implementation in the function `clustIndex` the scatter matrices are computed via `cov(x)` corrected with the size of the test data set. For the total scatter matrix T, this problem in the computation cannot appear. In certain groups, the within scatter matrix W_g does not exist, caused by computation with the help of the already announced penalized covariance. Though, the pooled within groups scatter matrix W does not exist with all the scatter matrices of each cluster. If this inconvenience occurs, the cluster should be crossed out of the investigated cluster solution. For this reason, the performance measures in the second group returns `NA`, i.e. *Scott*, *Friedman*, *Rubin*, *Marriot*, such as *trace(covW)* and *traceW* are not defined in these cases.

5.1.3 Davies Bouldin

Another weakness of this performance measurement lies again in the computation in `clustIndex`. As announced previously, some cluster are empty through the assignment of the `predict` function. Through the assignment of the data points in weakly engaged clusters, empty groups can be generated. However, in the computation of the function, these problems are not caught by an auxiliary function. In these cases, the calculation are stopped for all the internal validation indices. Thus, the problem is solved through an auxiliary function, which `NA` returns, and so on, the computation of the internal validation indices are not stopped anymore.

5.2 `R` package - ”`validator`”

The functions, which are used in the benchmark study, are implemented in an `R` package, called `validator`. The package `validator` is already uploaded on `CRAN` and can be viewed on the website http://CRAN.R-project.org/package=validator. The package follows on the description of Leisch (2008). There are two function calls in the package, which includes the functions of the external and internal validation indices. These function calls are listed in the appendix for further details of implementation.

5.2.1 Function Call extVal

Firstly, the external validation indices are based on the similarity table from the package clv. Only four of the twenty used validation indices were implemented, which is completed by the implementation of the function extVal. Even the weaker validation indices were implemented in the new package. On this reason, the user can decide between these indices to reproduce the gained results of this thesis. The package validator is supplemented with the new external validation indices, which are described in this thesis and based on Albatineh et al. (2006). Through the function call extVal, all of the external validation indices are selected. Furthermore, each of these indices can be called by its own. The function call only needs vectors with the true and the proposed cluster assignments, also the agreement of two cluster partitions can be evaluated, such as done in this thesis. Then, the external validation indices were computed on the base of the similarity table of clv.

In example on the base of the R package mlbench, an artificial two-dimensional and gaussian distributed data set is generated in the following way:

```
> x <- mlbench.2dnormals(500, 3)
> str(x)

List of 2
 $ x      : num [1:500, 1:2] 1.984 -1.938 0.746 1.352 1.628 ...
 $ classes: Factor w/ 3 levels "1","2","3": 1 2 1 3 1 1 2 1 2 2 ...
 - attr(*, "class")= chr [1:2] "mlbench.2dnormals" "mlbench"
```

The generated data is structured as follows. The first part of the data set includes the artificial data and the second part is the group assignment of the generated data set. This is the correct cluster assignment of the generated data. Afterwards, the k-means cluster algorithm is running on the data for a three cluster solution, either the function kmeans or such as in this thesis with kcca from the package flexclust. Then, the data points are assigned to a particular cluster given by the cluster centers of the data.

```
> cl <- kcca(x$x, 3)
> pred <- predict(cl, x$x)
```

Thereafter, the external validation indices can be computed and these indices have to be evaluated as described in section 2.3.1. The predicted cluster assignments are compared to the true cluster assignment.

```
> round(extVal(pred, x$class), 4)

      Hamann  Czekanowski   Kulczynski McConnaughey       Peirce
      0.6998       0.7749       0.7749       0.6512       0.6627
    Wallace1     Wallace2        Gamma       Sokal1        Fager
      0.7742       0.7757       0.7130       0.8312       0.7725
      Sokal2       Sokal3        Gower        Roger      Kruskal
      0.4626       0.6877       0.9189       0.7390       0.9289
     Pearson         Rand      Jaccard       Folkes       Russel
      0.0000       0.8499       0.6326       0.7749       0.2584
```

5.2.2 Function Call intVal

Secondly, the internal validation indices were implemented as well in the package. Through the function call intVal, the validation indices of this benchmark study can be reproduced. The internal validation indices, which bases on the description of Weingessel et al. (2002), are taken from the package cclust with some small changes, e.g. changing for-loops with the command apply. Furthermore, the sum-of-squares within are recalculated to the passed data points of the test data set. Then, better results of the internal validation indices can be reproduced by the new function intVal. The problem of the empty or rather engaged cluster are not solved by the function in the package. Not all of the implemented indices were taken to the new package validator. Only the announced internal validation indices are implemented in the package. Sixteen validation indices can be called by the function, such as the indices by its own. Weaker validation indices are observed as well to reproduce the gained results of this thesis. On this reason, the user can decide between the strengths and weaknesses of the internal validation indices by its own.

For improvements in understanding, the reproducing of the internal validation is shown as follows. Here again, the previous generated data set can be used as well for this example (for details see section 5.2.1). For these validation indices, the assignment of the data points to a particular group is not required. However, the function intVal is only implemented for kcca objects, e.g. the function does not work for kmeans objects. In this example, the k-means cluster algorithm of a three-cluster solution is performed by the function kmeans. Afterwards, the result is converted to an object of the class kcca by the conversion function as.kcca, shown as follows:

```
> cl <- kmeans(x$x, 3)
> cl <- as.kcca(cl, x$x)
```

Hence, after the converting the class assignment, the internal validation indices can be computed. The validation indices only need the performed clusters and the data set. Here, the test data set does not have to be the same as the training data set, on which the cluster algorithm is running through. Then, the values can be used to determine the number of clusters in the test or training data set.

```
> round(intVal(cl, x$x), 4)

    calinski           db     hartigan    ratkowsky        scott
    480.5859       1.1033       0.6596       0.4686    1075.8589
     marriot         ball       trcovw       tracew     friedman
1556832.5781     277.4970  186552.6716     832.4910       3.8824
       rubin      xuindex         dunn connectivity   silhouette
      8.5996      -5.2993       0.0064      64.4147       0.4347
```

Chapter 6

Conclusions

This thesis discussed a benchmark experiment on comparing and ranking cluster validation indices, either external indices or internal indices. The empirical study showed the strengths and weaknesses in the validation indices on different data sets. Notable was the weaknesses of three external validation indices, on which bad results in the study were recognized. Even the logit model showed their weaknesses in the probabilities for finding the correct number of clusters. All the other external validation indices were reacting similar to each cluster problem. The internal validation indices were fluctuating in their results for finding the correct cluster solution. Indeed, notable for the internal indices three of them resulted their proposed number of clusters very well, even in higher dimensions.

The investigated model in this thesis is non-optimal for fitting the results. However, it reflects through the prefix of the coefficients their behaviour in changing the dimensions with the corresponding standard deviations and even the sample method. The probabilities of the success for finding the correct cluster solution was not been bad as well. On this advantages the framework of the benchmark study can be diversified. The cluster scenarios can be extend in changing the true number of clusters within each dimension. Then, the empirical study can be run through the proposed number of clusters in each true cluster solution. Thus, even the model can be extend in finding the correct number of clusters with their incorrect detection for the proposed number from the validation indices. Indeed, such extensions in simulation studies can be confronted with memory and runtime problems, when changing the data sets into higher dimensions.

Furthermore, the relative criteria of the internal validation indices, which has been ignored in this thesis, could be an extension for a benchmark study. The behaviour of changing their parameters can be simulated and, maybe, it can be derive any rules for such indices in the simulation study. This could be interesting for next benchmark studies on cluster validation.

Next, the electronic appendix has to be added to the `R` package `validator`. Then, anyone can reproduce the results of this thesis by its own. That means, the package has to be updated with the investigated data sets and the functions, which were needed for evaluation the indices, has to be added. Furthermore, the package has to be upgraded with improvement suggestions by each user of the package.

Appendix A

List of Functions

A.1 Function extVal

> extVal

```
function (x, y, index = "all")
{
    x <- as.integer(x)
    y <- as.integer(y)
    sim <- std.ext(x, y)
    Hamann <- function(sim) {
        ham <- ((sim$SS + sim$DD) - (sim$SD + sim$DS))/(sim$SS +
            sim$SD + sim$DS + sim$DD)
        return(ham)
    }
    Czekanowski <- function(sim) {
        cze <- 2 * sim$SS/(2 * sim$SS + sim$SD + sim$DS)
        return(cze)
    }
    Kulczynski <- function(sim) {
        kul <- 0.5 * ((sim$SS/(sim$SS + sim$SD)) + (sim$SS/(sim$SS +
            sim$DS)))
        return(kul)
    }
    McConnaughey <- function(sim) {
        mcconn <- ((sim$SS)^2 + sim$SD * sim$DS)/((sim$SS + sim$SD) *
            (sim$SS + sim$DS))
        return(mcconn)
    }
    Peirce <- function(sim) {
        pei <- (sim$SS * sim$DD - sim$SD * sim$DS)/((sim$SS +
            sim$DS) * (sim$SD + sim$DD))
        return(pei)
    }
    Wallace1 <- function(sim) {
        wall1 <- (sim$SS)/(sim$SS + sim$SD)
        return(wall1)
    }
    Wallace2 <- function(sim) {
        wall2 <- (sim$SS)/(sim$SS + sim$DS)
        return(wall2)
    }
    Gamma <- function(sim) {
        gam <- (sim$SS * sim$DD + sim$SD * sim$DS)/sqrt((sim$SS +
            sim$SD) * (sim$SS + sim$DS) * (sim$DS + sim$DD) *
            (sim$SD + sim$DD))
        return(gam)
    }
    Sokal1 <- function(sim) {
        sok1 <- 0.25 * ((sim$SS/(sim$SS + sim$SD)) + (sim$SS/(sim$SS +
            sim$DS)) + (sim$DD/(sim$DD + sim$SD)) + (sim$DD/(sim$DD +
            sim$DS)))
        return(sok1)
    }
    Fager <- function(sim) {
        fag <- (sim$SS/sqrt((sim$SS + sim$SD) * (sim$SS + sim$DS))) -
            (0.5/sqrt(sim$SS + sim$SD))
        return(fag)
    }
    Sokal2 <- function(sim) {
        sok2 <- (sim$SS/(sim$SS + 2 * (sim$SD + sim$DS)))
        return(sok2)
    }
    Sokal3 <- function(sim) {
        sok3 <- ((sim$SS * sim$DD)/sqrt((sim$SS + sim$SD) * (sim$SS +
            sim$DS) * (sim$DS + sim$DD) * (sim$SD + sim$DD)))
        return(sok3)
    }
    Gower <- function(sim) {
        gow <- ((sim$SS + sim$DD)/(sim$SS + 0.5 * (sim$SD + sim$DS) +
            sim$DD))
        return(gow)
    }
    Roger <- function(sim) {
        rog <- ((sim$SS + sim$DD)/(sim$SS + 2 * (sim$SD + sim$DS) +
            sim$DD))
        return(rog)
    }
    Kruskal <- function(sim) {
        goo <- ((sim$SS * sim$DD - sim$SD * sim$DS)/(sim$SS *
            sim$DD + sim$SD * sim$DS))
        return(goo)
    }
    Pearson <- function(sim) {
        phi <- (sim$SS * sim$DD + sim$SD * sim$DS)/((sim$SS +
            sim$SD) * (sim$SS + sim$DS) * (sim$DS + sim$DD) *
            (sim$SD + sim$DD))
        return(phi)
    }
    index <- pmatch(index, c("Hamann", "Czekanowski", "Kulczynski",
        "McConnaughey", "Peirce", "Wallace1", "Wallace2", "Gamma",
        "Sokal1", "Fager", "Sokal2", "Sokal3", "Gower", "Roger",
        "Kruskal", "Pearson", "Rand", "Jaccard", "Folkes", "Russel",
        "all"))
    if (is.na(index))
        stop("invalid clustering index")
    if (index == -1)
        stop("ambiguous index")
    vecallindex <- numeric(20)
    if (any(index == 1) || (index == 21))
        vecallindex[1] <- Hamann(sim)
    if (any(index == 2) || (index == 21))
        vecallindex[2] <- Czekanowski(sim)
    if (any(index == 3) || (index == 21))
        vecallindex[3] <- Kulczynski(sim)
    if (any(index == 4) || (index == 21))
        vecallindex[4] <- McConnaughey(sim)
    if (any(index == 5) || (index == 21))
        vecallindex[5] <- Peirce(sim)
    if (any(index == 6) || (index == 21))
        vecallindex[6] <- Wallace1(sim)
    if (any(index == 7) || (index == 21))
        vecallindex[7] <- Wallace2(sim)
    if (any(index == 8) || (index == 21))
        vecallindex[8] <- Gamma(sim)
    if (any(index == 9) || (index == 21))
        vecallindex[9] <- Sokal1(sim)
    if (any(index == 10) || (index == 21))
        vecallindex[10] <- Fager(sim)
    if (any(index == 11) || (index == 21))
```

```
        vecallindex[11] <- Sokal2(sim)
    if (any(index == 12) || (index == 21))
        vecallindex[12] <- Sokal3(sim)
    if (any(index == 13) || (index == 21))
        vecallindex[13] <- Gower(sim)
    if (any(index == 14) || (index == 21))
        vecallindex[14] <- Roger(sim)
    if (any(index == 15) || (index == 21))
        vecallindex[15] <- Kruskal(sim)
    if (any(index == 16) || (index == 21))
        vecallindex[16] <- Pearson(sim)
    if (any(index == 17) || (index == 21))
        vecallindex[17] <- clv.Rand(sim)
    if (any(index == 18) || (index == 21))
        vecallindex[18] <- clv.Jaccard(sim)
    if (any(index == 19) || (index == 21))
        vecallindex[19] <- clv.Folkes.Mallows(sim)
    if (any(index == 20) || (index == 21))
        vecallindex[20] <- clv.Russel.Rao(sim)
    names(vecallindex) <- c("Hamann", "Czekanowski", "Kulczynski",
        "McConnaughey", "Peirce", "Wallace1", "Wallace2", "Gamma",
        "Sokal1", "Fager", "Sokal2", "Sokal3", "Gower", "Roger",
        "Kruskal", "Pearson", "Rand", "Jaccard", "Folkes", "Russel")
    if (index < 21)
        vecallindex <- vecallindex[index]
    return(vecallindex)
}
```

A.2 Function intVal

```
> intVal

function (y, x, index = "all")
{
    clres <- y
    cluster <- predict(clres, x)
    x <- as.matrix(x)
    clsize <- table(cluster)
    centers <- clres@centers
    varwithinss <- function(x, centers, cluster) {
        x <- (x - centers[cluster, ])^2
        varwith <- aggregate(x, by = list(cluster), FUN = sum)
        varwithins <- as.matrix(varwith[, -1])
    return(varwithins)
}
withinss <- function(varwithins) {
    withins <- apply(varwithins, 1, sum)
    return(withins)
}
gss <- function(x, clsize, withins) {
    n <- sum(clsize)
    k <- length(clsize)
    allmean <- apply(x, 2, mean)
    dmean <- sweep(x, 2, allmean, "-")
    allmeandist <- sum(dmean^2)
    wgss <- sum(withins)
    bgss <- allmeandist - wgss
    zgss <- list(wgss = wgss, bgss = bgss)
    return(zgss)
}
vargss <- function(x, clsize, varwithins) {
    nvar <- dim(x)[2]
    varallmean <- apply(x, 2, mean)
    vardmean <- (sweep(x, 2, varallmean, "-"))^2
    varallmeandist <- apply(vardmean, 2, sum)
    varwgss <- apply(varwithins, 2, sum)
    vartss <- varallmeandist
    varbgss <- vartss - varwgss
    zvargss <- list(vartss = vartss, varbgss = varbgss)
    return(zvargss)
}
ttww <- function(x, clsize, cluster) {
    n <- sum(clsize)
    k <- length(clsize)
    w <- 0
    tt <- cov(x) * (n - 1)
    for (l in 1:k) w <- w + cov(x[cluster == l, ]) * (clsize[l] -
        1)
    zttw <- list(tt = tt, w = w)
    return(zttw)
}
calinski <- function(zgss, clsize) {
    n <- sum(clsize)
    k <- length(clsize)
    vrc <- (zgss$bgss/(k - 1))/(zgss$wgss/(n - k))
    return(vrc = vrc)
}
```

```
cindex <- function(withins, minmaxd, clsize) {
    dw <- sum(withins * clsize)
    cindex <- (dw - minmaxd$mindw)/(minmaxd$maxdw - minmaxd$mindw)
    return(cindex)
}
db <- function(withins, centers, cluster) {
    mse <- withins/table(cluster)
    r <- outer(mse, mse, "+")/as.matrix(dist(centers, diag = TRUE))
    diag(r) <- 0
    db <- mean(apply(r, 1, max))
    return(db)
}
hartigan <- function(zgss) {
    hart <- log(zgss$bgss/zgss$wgss)
    return(hart)
}
ratkowsky <- function(zvargss, clsize) {
    k <- length(clsize)
    rat <- mean(sqrt(zvargss$varbgss/zvargss$vartss))
    rat <- rat/sqrt(k)
    return(rat)
}
scott <- function(zttw, clsize) {
    n <- sum(clsize)
    dettt <- prod(eigen(zttw$tt)$values)
    detw <- prod(eigen(zttw$w)$values)
    scott <- n * log(dettt/detw)
    return(scott)
}
marriot <- function(zttw, clsize) {
    k <- length(clsize)
    detw <- prod(eigen(zttw$w)$values)
    mar <- (k^2) * detw
    return(mar)
}
ball <- function(withins, clsize) {
    ball <- sum(withins)/length(clsize)
}
tracecovw <- function(zttw) {
    trcovw <- sum(diag(cov(zttw$w)))
    return(trcovw)
}
tracew <- function(zttw) {
    tracew <- sum(diag(zttw$w))
    return(tracew)
}
friedman <- function(zttw) {
    b <- zttw$tt - zttw$w
    fried <- sum(diag(solve(zttw$w) %*% b))
    return(fried)
}
rubin <- function(zttw) {
    dettt <- prod(eigen(zttw$tt)$values)
    detw <- prod(eigen(zttw$w)$values)
    friedm <- dettt/detw
    return(friedm)
}
xu <- function(x, clsize, zgss) {
    n <- sum(clsize)
    k <- length(clsize)
    d <- dim(x)[2]
    xuindex <- d * log(sqrt(zgss$wgss/(d * (n^2)))) + log(k)
    return(xuindex)
}
varwithins <- varwithinss(x, centers, cluster)
withins <- withinss(varwithins)
zgss <- gss(x, clsize, withins)
zttw <- ttww(x, clsize, cluster)
index <- pmatch(index, c("calinski", "db", "hartigan", "ratkowsky",
    "scott", "marriot", "ball", "trcovw", "tracew", "friedman",
    "rubin", "xuindex", "dunn", "connectivity", "silhouette",
    "all"))
if (is.na(index))
    stop("invalid clustering index")
if (index == -1)
    stop("ambiguous index")
vecallindex <- numeric(15)
if (any(index == 1) || (index == 16))
    vecallindex[1] <- calinski(zgss, clsize)
if (any(index == 2) || (index == 16))
    vecallindex[2] <- db(withins, centers, cluster)
if (any(index == 3) || (index == 16))
    vecallindex[3] <- hartigan(zgss)
if (any(index == 4) || (index == 16)) {
    zvargss <- vargss(x, clsize, varwithins)
    vecallindex[4] <- ratkowsky(zvargss, clsize)
}
if (any(index == 5) || (index == 16))
```

```
        vecallindex[5] <- scott(zttw, clsize)
    if (any(index == 6) || (index == 16))
        vecallindex[6] <- marriot(zttw, clsize)
    if (any(index == 7) || (index == 16))
        vecallindex[7] <- ball(withins, clsize)
    if (any(index == 8) || (index == 16))
        vecallindex[8] <- tracecovw(zttw)
    if (any(index == 9) || (index == 16))
        vecallindex[9] <- tracew(zttw)
    if (any(index == 10) || (index == 16))
        vecallindex[10] <- friedman(zttw)
    if (any(index == 11) || (index == 16))
        vecallindex[11] <- rubin(zttw)
    if (any(index == 12) || (index == 16))
        vecallindex[12] <- xu(x, clsize, zgss)
    if (any(index == 13) || (index == 16)) {
        vecallindex[13] <- dunn(clusters = cluster, Data = x)
    }
    if (any(index == 14) || (index == 16)) {
        vecallindex[14] <- connectivity(clusters = cluster, Data = x)
    }
    if (any(index == 15) || (index == 16)) {
        dist <- dist(x)
        vecallindex[15] <- summary(silhouette(cluster, dist))$si.summary[4]
    }
    names(vecallindex) <- c("calinski", "db", "hartigan", "ratkowsky",
        "scott", "marriot", "ball", "trcovw", "tracew", "friedman",
        "rubin", "xuindex", "dunn", "connectivity", "silhouette")
    if (index < 16)
        vecallindex <- vecallindex[index]
    return(vecallindex)
}
```

List of Figures

List of Tables

Bibliography

Ahmed N Albatineh, Magdalena Niewiadomska-Bugaj, and Daniel Mihalko. On similarity indices and correction for chance agreement. *Journal of Classification*, 23:301–313, 2006.

Guy Brock, Vasyl Pihur, Susmita Datta, and Somnath Datta. `clValid`: An `R` package for cluster validation. *Journal of Statistical Software*, 25(4):1–22, 3 2008. URL `http://www.jstatsoft.org/v25/i04`.

Ricardo J.G.B. Campello. A fuzzy extension of the rand index and other related indexes for clustering and classification assessment. *Pattern Recognition Letters*, 28(7):833–841, May 2007.

Evgenia Dimitriadou. *cclust: Convex Clustering Methods and Clustering Indexes*, 2009. URL `http://CRAN.R-project.org/package=cclust`. R package version 0.6-16.

Sara Dolnicar and Friedrich Leisch. Evaluation of structure and reproducibility of cluster solutions using the bootstrap. *Marketing Letters*, 21:83–101, 2010.

Manuel J. A. Eugster, Torsten Hothorn, and Friedrich Leisch. Exploratory and inferential analysis of benchmark experiments. Technical Report 30, Department of Statistics, Ludwig-Maximilians-University Munich, Germany, 2008.

Ludwig Fahrmeir, Thomas Kneib, and Stefan Lang. *Regression. Modelle, Methoden und Anwendungen.* Springer, 1st edition, 2007.

H. P. Friedman and J. Rubin. On some invariant criteria for grouping data. *Journal of the American Statistical Association*, 62(320):1159–1178, 1967.

Maria Halkidi, Yannis Batistakis, and Michalis Vazirgiannis. On clustering validation techniques. *Journal of Intelligent Information Systems*, 17:107–145, 2001.

Julia Handl, Joshua Knowles, and Douglas B. Kell. Computational cluster validation in post-genomic data analysis. *Bioinformatics*, 21(15):3201–3212, 2005.

Julia Handl, Joshua Knowles, and Douglas B. Kell. Supplementary material to computational cluster validation in post-genomic data analysis. pages 1-3, 2006. URL `http://dbkgroup.org/handl/clustervalidation/`.

Trevor Hastie, Robert Tibshirani, and Jarome Friedman. *The Elements of Statistical Learning*. Springer Series in Statistics, 2nd edition, 2009.

Torsten Hothorn, Friedrich Leisch, Achim Zeileis, and Kurt Hornik. The design and analysis of benchmarking experiments. *Journal of Computational and Graphical Statistics*, 14(3):675–699, 2005.

Krzysztof Kryszczuk and Paul Hurley. Estimation of the number of clusters using multiple clustering validity indices. *Lecture Notes in Computer Science*, 5997: 114–123, 2010.

Friedrich Leisch. Sweave: Dynamic generation of statistical reports using literate data analysis. In Wolfgang Härdle and Bernd Rönz, editors, *Compstat 2002 - Proceedings in Computational Statistics*, pages 575–580. Physica Verlag, Heidelberg, 2002. URL `http://www.stat.uni-muenchen.de/~leisch/Sweave`.

Friedrich Leisch. A toolbox for k-centroids cluster analysis. *Computational Statistics and Data Analysis*, 51(2):526–544, 2006.

Friedrich Leisch. Creating r packages: A tutorial. In Paula Brito, editor, *Compstat 2008 - Proceedings in Computational Statistics*. Physica Verlag, 2008. URL `http://epub.ub.uni-muenchen.de/6175/`.

Friedrich Leisch and Evgenia Dimitriadou. *mlbench: Machine Learning Benchmark Problems*, `R` package version 2.0-0 edition, 2010. URL `http://CRAN.R-project.org/package=mlbench`.

Glenn W. Milligan and Martha C. Cooper. An examination of procedures for determining the number of clusters in a dataset. *The Psychometric Society*, 50(2): 159–179, June 1985.

Lukasz Nieweglowski. *clv: Cluster Validation Techniques*, 2009. URL `http://CRAN.R-project.org/package=clv`. R package version 0.3-2.

R Development Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2010. URL `http://www.R-project.org`. ISBN 3-900051-07-0.

Andreas Weingessel, Evgenia Dimitriadou, and Sara Dolnicar. An examination of indexes for determining the number of clusters in binary data sets. *Psychometrica*, 67(1):137–160, March 2002.