User Clustering Based on Keystroke Dynamics

Maximiliano Bertacchini[1,2], Carlos E. Benitez[1] and Pablo I. Fierens[2,3]

[1] Si6 Labs - CITEFA - Inst. de Investigaciones Científicas y Técnicas para la Defensa
San J. B. de La Salle 4397 (B1603ALO),Villa Martelli, Buenos Aires, Argentina
Tel: (5411) 4709-8285, Fax: (5411) 4709-5363
{mbertacchini,cbenitez}@citefa.gov.ar
[2] ITBA - Instituto Tecnológico Buenos Aires
Av. Eduardo Madero 399 (C1106ACD), Buenos Aires, Argentina
Tel: (5411) 6393-4822
pfierens@itba.edu.ar
[3] CONICET - Consejo Nacional de Investigaciones Científicas y Técnicas
Avda. Rivadavia 1917 (C1033AAJ), Buenos Aires, Argentina
Tel: (5411) 5983-1420

Abstract. The PAM clustering algorithm is applied on the Si6 keystroke dataset in order to identify sessions of the same users. A number of heuristical outlier filters based on statistical properties of keystroke latencies are proposed and run on the dataset. Different tests are performed varying the number of digraphs that compose each observation and its dimensionality, in order to verify the assumption that more data gives a better quality of clustering and to estimate the minimum required number of dimensions. The number of clusters is estimated through the silhouette algorithm. Resulting clustering accuracy is measured by means of the F-measure, showing the viability of user identification through keystroke analysis.

1 Introduction

Keystroke dynamics are a set of biometric techniques that allow to identify a person according to her keystroke pattern, that is, her typing rythm on a computer keyboard. These techniques are based on the detailed timing that describes when each key was depressed and when it was released. This biometric information is supposedly as unique for each individual as the handwritten signature.

This technique has been mostly used in improving user authentication mechanisms (e.g. in the computer login process) or in anomaly detection. Nevertheless, in this paper we describe a clustering analysis performed on our own keystroke dataset in order to verify whether it is possible to identify users, and under which conditions, based on their keystroke patterns, and under which conditions. The ultimate goal is the identification of computer intruders.

This paper is structured as follows: Section 2 lists some of the previous work this paper is based on; Section 3 describes the dataset as well as the outlier elimination process; Section 4 explains the clustering algorithm applied in order to group same user data in clusters; results are shown in Section 5. Finally, conclusions and possible future work are presented in Section 6

2 Related Work

2.1 Keystroke Analysis

A large number of articles have been published in the past 20 years related to keystroke dynamics techniques. Most of them focus on user authentication reinforcement or in anomaly detection in user behavior. Starting in 1990, a number of articles describe several techniques for user authentication reinforcement based on measuring the distance between the user keystroke pattern and the already enrolled user keystroke profile [1,2,3,4,5,6,7]. On the other hand, other group of investigators analyzed and studied free text user authentication methods [8,9,10,11,12].

Almost every published article related to keystroke dynamics is analyzed and compared by Killourhy and Maxion in [13].

2.2 Data Clustering

Clustering algorithms group a set of elements into subsets or *clusters*, so that similar elements are assigned to the same cluster [14]. This requires the definition of a distance measure, the most common of which is the *Euclidean distance*. Clustering is a method of unsupervised learning, meaning that no human expert is involved in the process. Partitional or flat clustering algorithms, as opossed to hierarchical clustering, define no structure or relation among clusters [15]. The most popular flat clustering algorithm is k-means[15], which assigns observations to the cluster with the nearest mean. The k-medoids algorithm is similar to k-means, except that it chooses data points as cluster centers, which are called *medoids*. A medoid is the element of a cluster with the minimum average dissimilarity to all the objects in the same cluster; it is the most centrally located point in that cluster. The *Partitioning Around Medoids* (PAM) algorithm is an implementation of k-medoids, first proposed by Kaufman and Rousseeuw [16]. It has the advantage over k-means that it does not need a definition of mean, as it works on dissimilarities; for this reason, any arbitrary measure distance can be used. It is also more robust to noise and outliers as compared to k-means because it minimizes a sum of dissimilarities instead of a sum of squared Euclidean distances.

The PAM algorithm is as follows [17]:

1. Randomly select k of the n data points as the medoids.
2. Associate each data point to the closest medoid (using any valid distance metric).
3. For each medoid m:
   (a) For each non-medoid data point o:
      i. Swap m and o and compute the total cost of the configuration (i.e. the summation of the distances from each data point to the medoid in its cluster).
4. Select the configuration with the lowest cost.
5. Repeat steps 2 to 5 until there is no change in the medoid.

3 The Si6 Dataset

The *Si6 dataset*[18] was collected by the Si6 Labs[4] through a web application named `k-profiler`[5]. It is publicly available at http://www.citefa.gov.ar/si6/k-profiler/dataset/.This dataset contains anonymized keystroke data of 66 labeled users. For this dataset texts were selected in order to obtain a large number of common digraphs. In this way, the most frequent digraphs in Spanish are typed many times by users in the selected texts.

The dataset contains a total of 66 files, one file per typing session, corresponding to 62 different users. The filename corresponds to the user name prefixed by a `UNIX` timestamp, e.g. `1266936532_user_000.finished`. Completed sessions have a `.finished` extension; incomplete sessions have a `.unfinished` extension.

Each typing session is composed by 15 sentences. Each sentence is identified by an identifier ranging from `ks_00` to `ks_14`. Invalid sentences are labeled with prefix `invalid-`. Following is a sample fragment of a typing session file:

```
...
ks_00 1279288592438 up 9
ks_00 1279288592902 dn 69
ks_00 1279288592980 dn 78
ks_00 1279288592986 up 69

ks_01 1279288591988 dn 190
ks_01 1279288591995 up 68
ks_01 1279288592083 up 190
ks_01 1279288592326 dn 9
...
```

Columns correspond to: sentence identifier, `UNIX` timestamp with milliseconds, keystroke event (released - `up`, or depressed - `dn`) and ASCII key code.

In order to eliminate outliers and make the clustering process as controlled as possible, some preprocessing tasks have been performed to the original data. Those tasks are described in the next subsections.

3.1 Data Preprocessing

We selected only those sessions which were complete (58 out of 66), i.e., sessions with a `.finished` file extension. There are three users (`user_000`, `user_001` and `user_103`) who typed the sentences more than once. In these cases, we arbitrarily chose a single session for each user. In all cases, only valid sentences were considered, i.e. those sentences named `ks_00` through `ks_14`. Invalid sentences (named `invalid-*`) and UNIX commands (named `cm_*`) were discarded. This leaves a total of 54 users with 15 complete sentences each.

Then a set of digraphs is computed for each sentence of each session of each user, taking the elapsed time between two consecutive down keystroke events (`dn`) measured in milliseconds. Digraphs that contain at least one non-printable ASCII character (e.g. tab, newline, carriage return, backspace, etc.) are discarded. This leaves a total of 88946 digraph latency times corresponding to 411 unique digraphs.

[4] http://www.citefa.gov.ar/si6/
[5] http://www.citefa.gov.ar/si6/k-profiler

3.2 Subsession Grouping

User sessions, each of which is composed of 15 sentences, were split into subsessions in order to assess the clustering accuracy in function of the subsession size. The idea here is that the larger the subsession size, the better the clustering accuracy. For this purpose, the subsession schemes show in Table 1 were used, grouping sentences arbitrarily by their ordering number.

Description	Distribution of Sentences	Avg. No. of digraphs per subsession
3 groups of 5 sentences per user	`ks_00-ks_04`, `ks_05-ks_09`, `ks_10-ks_14`	24554
5 groups of 3 sentences per user	`ks_00-ks_02`, `ks_03-ks_05`, `ks_06-ks_08`, `ks_09-ks_11`, `ks_12-ks_14`	14732
15 groups of 1 sentence per user	No grouping (all 15 sentences taken alone)	4910

Table 1. Sentence grouping schemes

3.3 Outlier Filtering

Outliers are observations that are distant from the rest of the data [19]. Filtering outliers is often the single most important step in clustering data.

Too short digraph times may be the result of measurement errors in the user's Internet browser. Too long digraph times may be due to user hesitation or distraction, and should not be taken into account as they are not representative of the user's typing behavior. This gives place to the following proposed digraph filters. These filters are essentially heuristics based on intuitive or expected statistical properties of the typing behavior of a person and as such, are subject to further discussion and improvement.

Coarse Outlier Filtering As a first crude filter, digraphs with times lesser than 10ms or greater than 750ms are discarded. A second step filters out digraphs whose times are outside the range that comprises the median of the times of all remaining digraphs plus-minus the standard deviation of the times of all remaining digraphs, which is $140ms \pm 113ms \approx (25ms, 250ms)$. The median is used instead of the mean because it is less sensitive to outliers.

$$
\begin{aligned}
Filter_1(d_i): \ & \text{discard } d_i \text{ if } time(d_i) < 10ms \\
& \quad \text{or } time(d_i) > 750ms \\
Filter_2(d_i): \ & \text{discard } d_i \text{ if } time(d_i) < (m_D - \sigma_D = 25ms) \\
& \quad \text{or } time(d_i) > (m_D + \sigma_D = 250ms)
\end{aligned}
$$

where d_i is any digraph, $time(d_i)$ is the elapsed time between both keystrokes, m_D is the median of the time of all digraphs, and σ_D is the standard deviation of the time of all digraphs.

Both filters discard 15284 digraphs out of 88946 (17.18%).

Fine Outlier Filtering The remaining digraphs are run through another set of filters based on specific statistical properties of each subsession.

$Filter_3$ discards the "lonely" digraphs, i.e. those with only one appearance in each subsession. This is based on the assumption that digraphs with only one appearance possess little statistical support and as such are treated as outliers.

$Filter_4$ discards digraphs whose standard deviation of their times is greater than some certain threshold. This threshold was defined as twice the mean of the standard deviations of the times of all digraphs in that subsession. The idea here is to discard digraphs whose times deviate too much from the average deviation of the particular subsession.

Finally digraphs whose times are outside the range comprised by the median of that digraph for the particular subsession plus-minus twice its standard deviation in the same subsession are filtered out by $Filter_5$. This limits the possibility of confusing one user with another by discarding digraphs whose times deviate too much from the "average" behavior of that user. Again, the median is used instead of the mean because it is less sensitive to outliers.

$$
\begin{aligned}
Filter_3(d_i, S_j): &\text{ discard } d_i \text{ if } count(d, S_j) = 1 \\
Filter_4(d_i, S_j): &\text{ discard } d_i \text{ if } \sigma_{d,S_j} > \bar{\sigma}_{S_j} \times 2 \\
Filter_5(d_i, S_j): &\text{ discard } d_i \text{ if } time(d_i) < (m_{d,S_j} - \sigma_{d,S_j} \times 2) \\
&\qquad\qquad \text{or } time(d_i) > (m_{d,S_j} + \sigma_{d,S_j} \times 2)
\end{aligned}
$$

where d_i is an occurrence of digraph d in subsession S_j, $count(d, S_j)$ counts the number of occurrences of digraph d in that subsession, σ_{d,S_j} is the standard deviation of the times of all occurrences of digraph d in that subsession, m_{d,S_j} is the median of the times of all occurrences of that digraph in that subsession, and $\bar{\sigma}_{S_j}$ is the mean of the standard deviations of the times of all digraphs in subsession S_j.

4 Clustering

4.1 Feature Selection

The selection of relevant features or variables that represent the variability of the whole dimensionality of elements is a crucial step in clustering data. The idea is to build an n-dimensional vector for every element that will be fed into the clustering algorithm, which in this case are typing subsessions. The most prominent feature is, of course, the timing data of each digraph, which should uniquely characterize the user. The feature vector of each subsession was built from the mean of each digraph time, as in [1]. As a result, each subsession is represented by a vector of length 411 (the number of distinct digraphs in the dataset), with one dimension per digraph, each cell containing the mean of all appearances of that digraph in that subsession.

Digraphs are ordered decreasingly by the number of appearances among all users. Therefore, the values of the first dimensions in the feature vectors, which

correspond to the most frequent digraphs, represent the mean of a larger population of observations, and intuitively should be more relevant during the clustering process than the last dimensions, which correspond to seldom typed digraphs. This rises the question of how many digraphs, and which, are necessary to correctly identify the user of a subsession. The most convenient number of digraphs will be assessed based on the clustering quality by varying the feature vector size.

4.2 Cluster Cardinality

The k-means family of clustering algorithms, including PAM, requires a parameter commonly referred to as k that specifies the number of clusters to detect. Though in this paper the number of users is already known, a real-world scenario would require the calculation of the approximate number of expected clusters. Several algorithms exist that allow to estimate the value of k, such as the "elbow" method and the Akaike information criterion (AIC) (see, e.g. [15]). A straightforward method is called *Silhouette*[20]. Silhouette is a method of interpretation and validation of clusters of data, and provides a graphical representation of how well each element lies within its cluster. The silhouette of an element measures how close it is to the other elements of its cluster and how far it is from the neighboring cluster. The silhouette of element i, $s(i)$, is defined as follows:

$$s(i) = \frac{b(i) - a(i)}{max(a(i), b(i))} \qquad (-1 \leq s(i) \leq 1) \tag{1}$$

where $a(i)$ is the average dissimilarity of i with all other data within the same cluster, and $b(i)$ is the lowest average dissimilarity from i to any other cluster. A silhouette value of 1 implies an element appears to be appropriately clustered; a value of -1 means it appears to be in the wrong cluster.

The average silhouette width, i.e. the mean of $s(i)$ for all elements i in a particular cluster distribution, can be used to measure the overall quality of clustering. The optimum number of clusters can be determined by calculating the silhouette average for an increasing number of clusters and taking the one that yielded the highest value.

Figure 1 shows the average silhouette widths for a varying number of clusters, taking user subsessions in groups of 5 sentences. The highest value corresponds to 49 clusters, which is close to that of 54, the real number of users. For the sake of clarity, a fixed cardinality of 54 clusters was used in the tests.

4.3 Algorithm Implementation And Configuration

R[21] is a popular open-source programming language for statistical computing and graphics, based on the S programming language. The implementation of PAM provided by the `cluster` library[22] of the R language was used for the task of clustering the abovementioned data vectors. All default parameters of the `pam` function were used. Particularly, the distance used for calculating dissimilarities between observations is the *Euclidean metric*.

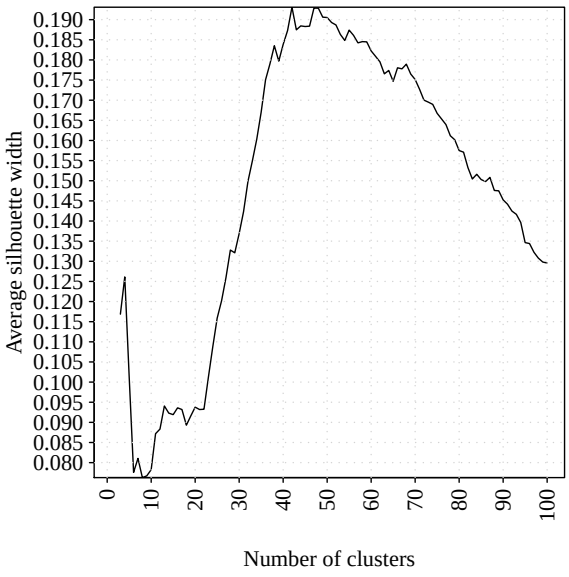

Fig. 1. Average silhouette width vs number of clusters. The highest value corresponds to 49 clusters, which is close to that of 54, the real number of users.

5 Evaluation

The process of clustering can be thought of as a series of decisions, one for each pair of elements [15]. A true positive (TP) decision assigns two similar elements (i.e. two subsessions belonging to the same user) to the same cluster; a true negative (TN) assigns two dissimilar elements (i.e. two subsessions belonging to different users) to different clusters. A false positive (FP) assigns two dissimilar elements to the same cluster; a false negative (FN) assigns two similar elements to different clusters. This results in the contingency table shown in Table 2.

	Same cluster	Different clusters	
Same class	TP	FN	$\rightarrow Recall$
Different classes	FP	TN	
	$\downarrow$ $Precision$		

Table 2. Classification contingency table

A number of statistical measures can be drawn from these values to measure the classification accuracy. *Precision* (see Eq. 2) measures the fraction of correct positive decisions (TP) among all the positive decisions ($TP + FP$). *Recall* (see Eq. 3) measures the fraction of correct positive decisions (TP) over decisions that should have been positive ($TP + FN$).

There is usually an inverse relationship between Precision and Recall, so that increasing one has the cost of reducing the other. *F-measure* (see Eq. 4) is a combination of both of them. An F-measure score of 1 means a perfect clustering, whereas the worst case is a score of 0.

$$Precision = \frac{TP}{TP + FP} \tag{2}$$

$$Recall = \frac{TP}{TP + FN} \tag{3}$$

$$F = 2 \times \frac{Precision \times Recall}{Precision + Recall} \tag{4}$$

A set of tests were run using an increasing number of digraphs (ordered decreasingly by their frequency among all users), for different grouping schemes. Figure 2 shows the resulting F-measure scores. As expected, the larger the subsession size, the better the quality of clustering. The scores start to stabilize at around 50 digraphs (i.e. the feature vector dimensionality), which is approximately $1/8^{th}$ of the complete digraph space, suggesting that only the most frequent digraphs are sufficient for an accurate classification.

The case with no subsession grouping is special in that the PAM algorithm was unable to successfuly clusterize the input data. A closer analysis revealed that the cause of this anomaly is the "lonely" digraph filter ($Filter_3$), which is too aggressive on small datasets, as in the case with no data grouping, where many digraphs appear just once in each sentence. As a result, the previous tests where run again, but with $Filter_3$ deactivated.

Figure 3 shows the newly obtained results. As expected, the quality of clustering data with no sentence grouping is improved, though results are still unsatisfactory. On the other hand, the other tests with grouped sentences give worse results, due to the fact that the lack of $Filter_3$ introduced too many "lonely" digraphs with little statistical support, which hinders the ability of the clustering algorithm to distinguish data from different users.

6 Conclusions and Future Work

We showed the feasibility of user identification by applying the PAM clustering algorithm on the Si6 keystroke dataset. The amount of available data proved to be paramount in that process. Which digraphs are most valuable and what is the minimum amount of data necessary for an accurate clustering of users is subject to further investigation. Outlier filters should be improved as well.

Though the Euclidean metric gave good results, other distances should also be tried. The estimation of the number of clusters should be extended by applying other algorithms in order to improve its accuracy.

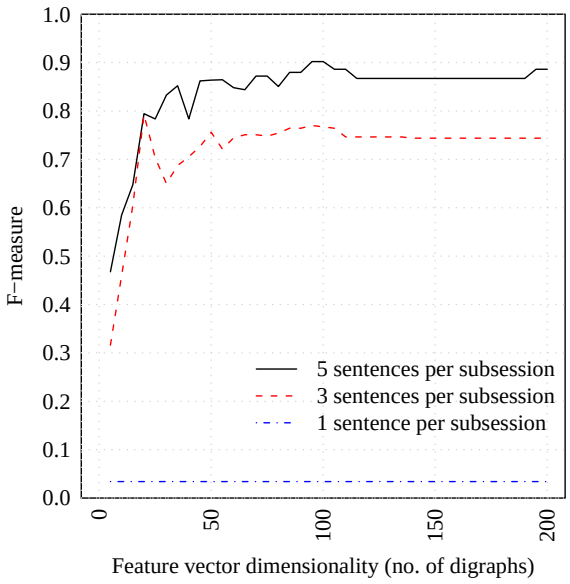

Fig. 2. F-measure of clustering different grouping schemes, taking an increasing number of digraphs (ordered by decreasing frequency in the dataset).

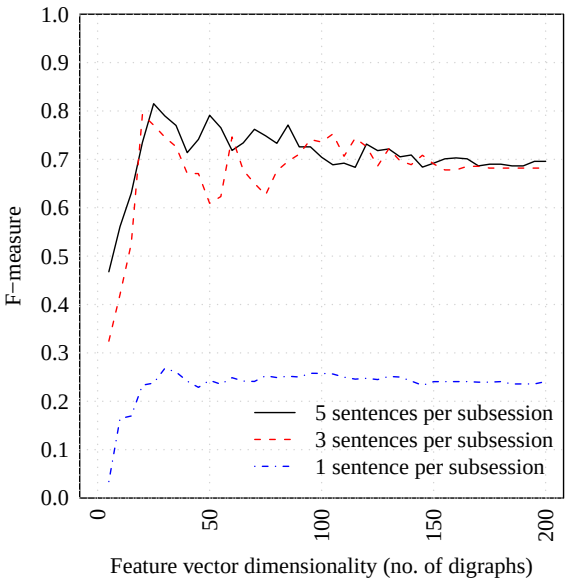

Fig. 3. F-measure of clustering different grouping schemes, taking an increasing number of digraphs (ordered by decreasing frequency in the dataset). "Lonely" digraph filter ($Filter_3$) not applied.

7 Acknowledgements

This work is part of a more extendend research project which is supported in part by ANPCyT (National Agency of Scientific Promotion and Technology of Argentina) under the grant PICTO CITEFA 2004 18623. We gratefully acknowledge financial support from ANPCyT. We would also like to thank all members of CITEFA's Si6 Lab and the peer reviewers for their advice and support.

References

1. Joyce, R., Gupta, G.: Identity authentication based on keystroke latencies. Communications of the ACM **33**(2) (1990) 168–176
2. Monrose, F., Rubin, A.: Authentication via keystroke dynamics. In: Proceedings of the Fourth ACM Conference on Computer and Communications Security, Zurich, Suiza (1997) 48–56
3. Obaidat, M., Sadoun, B.: Verification of computer users using keystroke dynamics. IEEE Transactions on Systems, Man, and Cybernetics, Part B **27**(2) (1997) 261–269

4. Tapiador, M., Sigüenza, J.: Fuzzy keystroke biometrics on web security. In: Proceedings Workshop on Automatic Identification Advanced Technologies -AutoID 99, IEEE (1999) 133–136
5. Yu, E., Cho, S.: Ga-svm wrapper approach for feature subset selection in keystroke dynamics identity verification. In: Proceedings of the International Joint Conference on Neural Networks, 2003. Volume 3. (2003) 2253–2257
6. Chen, W., Chang, W.: Applying hidden markov models to keystroke pattern analysis for password verification. In: IEEE International Conference on Information Reuse and Integration, IRI - 2004. (2004) 467–474
7. Jiang, C., Shieh, S., Liu, J.: Keystroke statistical learning model for web authentication. In: Proceedings of the 2nd ACM symposium on Information, computer and communications security, ACM New York, NY, USA (2007) 359–361
8. Gunetti, D., Picardi, C.: Keystroke analysis of free text. ACM Transactions on Information and System Security (TISSEC) **8**(3) (2005) 312–347
9. Sheng, Y., Phoha, V., Rovnyak, S.: A parallel decision tree-based method for user authentication based on keystroke patterns. IEEE Transactions on Systems, Man, and Cybernetics-Part B: Cybernetics **35**(4) (2005) 826–833
10. Quinlan, J.: Induction of decision trees. Machine learning **1**(1) (1986) 81–106
11. Sim, T., Janakiraman, R.: Are digraphs good for free-text keystroke dynamics? In: IEEE Conference on Computer Vision and Pattern Recognition, 2007. CVPR'07. (2007) 1–6
12. Hu, J., Gingrich, D., Sentosa, A.: A k-nearest neighbor approach for user authentication through biometric keystroke dynamics. In: Proceedings of the IEEE International Conference on Communications. (2008) 1556–1560
13. Killourhy, K., Maxion, R.: Comparing anomaly-detection algorithms for keystroke dynamics. In: IEEE/IFIP International Conference on Dependable Systems & Networks, 2009. DSN'09. (2009) 125–134
14. Tan, P.N., Steinbach, M., Kumar, V.: Introduction to Data Mining. Addison-Wesley (2006)
15. Manning, C.D., Raghavan, P., Schütze, H.: An Introduction to Information Retrieval. Press, Cambridge U. (2008)
16. Kaufman, L., Rousseeuw, P.J.: Finding groups in data: An introduction to cluster analysis. Wiley, NY (1990)
17. Theodoridis, S., Koutroumbas, K.: Pattern recognition. Academic Press (2006)
18. Bello, L., Benitez, C., Bertacchini, M., Pizzoni, J.C., Cipriano, M.: Collection and publication of a fixed text keystroke dynamics dataset. submitted to CACIC 2010 (2010)
19. Barnett, V., Lewis, T.: Outliers in statistical data. John Wiley & Sons (1994) ISBN 0-471-93094-6.
20. Rousseeuw, P.J.: Silhouettes: A graphical aid to the interpretation and validation of cluster analysis. Journal of Computational and Applied Mathematics **20** (1987) 53 – 65
21. R Development Core Team: R: A Language and Environment for Statistical Computing. R Foundation for Statistical Computing, Vienna, Austria. (2010) ISBN 3-900051-07-0.
22. Maechler, M., Rousseeuw, P., Struyf, A., Hubert, M.: Cluster analysis basics and extensions. Rousseeuw et al provided the S original which has been ported to R by Kurt Hornik and has since been enhanced by Martin Maechler: speed improvements, silhouette() functionality, bug fixes, etc. See the 'Changelog' file (in the package source) (2005)