Philipp KULICH, BSc.

Clustering of Biological Data

MASTER'S THESIS

to achieve the university degree of

Diplom-Ingenieur

Master's degree programme: **Biomedical Engineering**

submitted to

Graz, University of Technology

Supervisor

Dr. Gerhard Thallinger

Institute of Neural Engineering

Institute of Computational Biotechnology

Graz, **April, 2018**

I would like to thank my supervisor, Dr. Gerhard Thallinger, for the patient guidance, encouragement and advice he has provided throughout my time as his student.

I would also like to thank my family and my girlfriend who not just supported me through my whole study but also advised me through the process of researching and writing this thesis. This accomplishment would not have been possible without them.

Thank you.

AFFIDAVIT[1]

I declare that I have authored this thesis independently, that I have not used other than the declared sources/resources, and that I have explicitly indicated all material which has been quoted either literally or by content from the sources used. The text document uploaded to TUGRAZonline is identical to the present master's thesis.

Graz, _____ _____

Date Signature

[1]Beschluss der Curricula-Kommission für Bachelor-, Master- und Diplomstudien vom 10.11.2008; Genehmigung des Senates am 1.12.2008

Contents

Abstract

Gene expression analysis is getting better over years and so the amount of data produced increases. The range of different methods for processing data is also getting bigger each year. Clustering is an analysis method which deals with large datasets and tries to group the data with different distance measure methods. This approach leads to a better overview of the dataset and assists in creating results and drawing conclusions from the data.

Clustering is a good attempt to deal with big datasets and therefore it is often used in gene expression analysis. To get the best clustering results it is necessary to have a wide range of clustering possibilities like Hierarchical Clustering (HC) or k-means (KM) clustering.

In this thesis the clustering application Genesis was expanded by implementing the Mahalanobis Distance method, the Ward's linkage and two new clustering algorithms, Partitioning Around Medoids (PAM) and Transitive Clustering (TC).

To ensure correct implementation six data sets were used to check the new functions and features. Three datasets were taken from the already existing Genesis test files and were used to compare the clustering results from Hierarchical Clustering with different linkage measures and Mahalanobis distance measure with R and PAST. The remaining three datasets were used to validate the new clustering functions PAM and TC. These methods are using a precalculated similarity matrix to accelerate the clustering.

Comparable Silhouette value and F_1-score was yielded with different datasets. *Dataset 1* has a Silhouette value of 0.45 for TC and 0.4760 for PAM. The classification of the *Dataset 2* by TC achieves a F_1-score of 0,88 compared to 0.93 by ClustEval which is used as reference. PAM achieves a F_1-score of 0.91 by Genesis and 0.92 by ClustEval. The synthetic *Dataset 3* with 16 clusters has a F_1-score of 1.00 for TC and 0.76 for PAM. The *MCC* for *Dataset 3* is 1 for TC and 0.75 for PAM.

Key Words:

clustering, Genesis, Medoids, Transitive, Mahalanobis, Ward's

1 Introduction

The amount of data generated dramatically increased in the past years and the processing is getting more and more difficult. Over years multiple data processing methods like neural networks, clustering or deep learning developed. Some of these methods need to be trained and for that, huge amounts of annotated data are needed. If deep learning is applied, multiple neural networks are trained on classified data under supervision. Clustering can be done with several methods which does not need annotated training sets. With many different clustering methods for example hierarchical clustering or k-means clustering big datasets can be clustered in groups of interest and get analyzed. With the increase of data, also the complexity of the datasets increases and so multiple methods are needed to achieve the best clustering result.

1.1 Basic Clustering Approaches

To cluster different types of data, a huge variety of clustering methods has developed. Algorithms can be classified in five main types: *k-means*, *hierarchical*, *density-based*, *model-based* and *graph-based* [1]. Clustering is a technique which groups data, based on similarity. The similarity between data points is based on different distance measures for example the euclidean distance, the mahalanobis distance or the cosine correlation. All clustering algorithms can be divided in two main groups which are Agglomerative clustering and Divisive clustering. Hierarchical Clustering is agglomerative because in the beginning each data point represents a cluster which are merged and Transitive clustering is divisive because at the beginning the whole dataset is one cluster and then is spitted in multiple clusters.

The processing starts with calculating the overall similarity matrix for the whole dataset with n points. The similarity matrix consists of nxn entries with all distances between all data points. The standard for this distance calculation is the Euclidean distance.

When the distance matrix calculation is finished, the so called clustering, which can also be done by different algorithms, is processed. Hierarchical and k-means clustering are the most common methods and can be used for a huge variety of clustering tasks. In the beginning of the clustering process, the similarity matrix is searched for the two nearest points and this points are merged into a cluster representing a new point. For Hierarchical clustering a specific linkage, which calculates the best points to combine, is used. Common linkage types are Single Linkage, Avarage Linkage, Complete Linkage and Median Linkage. The last step is to recalculate the similarity matrix with the new point. This is repeated until the best clustering result is yielded (Figure 1). First the date is read to provide it for further processing. After this step, distances between the data points are being calculated and clusters can be created. There are many different

methods for creating the distances and clusters but these blocks should summarize all different types of clustering and show just a basic functionality. There are also different methods for quality measurement which decide whether the clusters are good enough not. If the quality is good, the data is written or displayed in some way but if the cluster has a bad quality, the data must be rearranged and the whole clustering algorithm starts recreating the distances and looping until the cluster quality is declared good enough. In some cases this can lead to an infinite loop so it is recommended to terminate after a defined number of iterations although the specified quality is not reached.

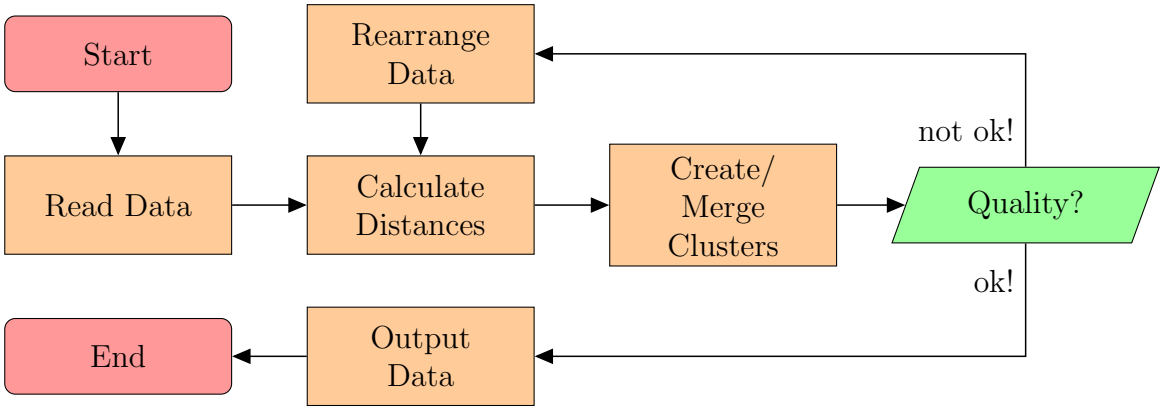


Figure 1: Overall clustering principle with a simple flowchart.

To find the correct clustering method is not easy because there are lots of different ways to cluster datasets. If we take hierarchical clustering, it is hard to say which way of calculating the similarity matrix will give the best result. There are many different types of distance measure algorithms and for each algorithm there are also different linkage methods to merge clusters. This leads to a huge amount of possibilities in parameter combination just for hierarchical clustering.

1.2 Clustering in Biomedical Research

The increasing amount of biomedical data is also pushed by the development of new research methods and the availability of easy accessible databases. Due to the dramatic increase of biological data and its electronic availability, data mining methods are getting more and more important over years [2]. Clustering provides a good opportunity to process this huge amounts of data. The clustered datasets are giving a fast and easy overview of possible dependencies or similarities. To get the best clustering results, it is necessary to know the different methods and functionalities of clustering algorithms. The number of publications with the keywords "*clustering*", "*gene*" and "*expression*" published in the last 35 years on PubMed [3] increased exponentially. The dramatic increase since 2000 shows the growing importance of gene expression clustering (Figure 2).

A large number of clustering software applications are available. Popular are PAST [4] which was originally developed for paleontological research but is now used for different data analysis tasks. Cluster 3.0 [5] was developed by Michael Eisen at Stanford University and is used for clustering genome-wide expression data. WebMeV [6] is a cloud based web application which supports analysis visualization and stratification of genomic data. This means that the data is not processed locally but submitted to a server (cloud) which does the calculation with a high amount of calculating power. TM4 Mev [7] is the stand alone client for this application. There is also a Python [8] application called Pycluster [9] which provides similar functionalities like Matlab [10] with the Clustering Toolbox [11] or R [12] with the cluster package [13]. For Matlab and R there are different packages and tools which provide clustering functions. ClustEval [14] is a server based application which compares the output of different clustering algorithms of different datasets and parameters and generates a standardized evaluation for easy comparison. Genesis [15, 16] is a software package which already supports some major clustering functions like hierarchical clustering, selforganizing maps, k-means and also some classification methods like principal component analysis and support vector machines [15]. Every clustering method can be started with ten different distance measure algorithms which gives a huge amount of possibilities.

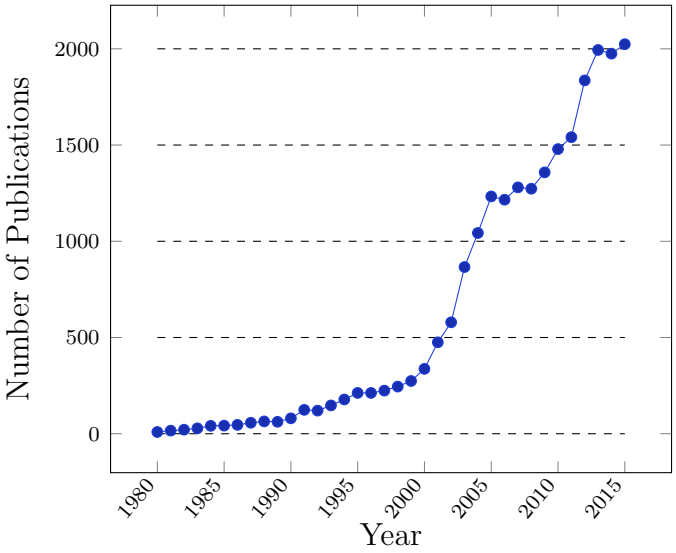


Figure 2: Increase of Publications with the keywords "*clustering*", "*gene*" and "*expression*" over the last 35 years.

In the context of this thesis, Genesis is extended with two new methods, Partitioning Around Medoids (PAM) [17] and Transitive Clustering (TC) [18] which are giving good results on biological data according to Wiwie et all. [14].

1.3 Aims of the Thesis

The overall goal of this thesis was to extend Genesis with a new distance measure, additional linkage calculation methods for hierarchical clustering and two clustering methods which are relevant in biomedical research. Specifically the following should be achieved:

- Implementation
  - of Mahalanobis Distance
  - of Ward's Linkage
  - of Partitioning Around Medoids clustering
  - of Transitive Clustering
- Testing the implemented code by comparison with established applications.
- Creating and uploading a new Genesis release.

2 Methods

2.1 Gene Expression Data

Gene expression data are generated in general via DNA-Microarrays [19] or RNA-seq [20]. These methods are used to measure expression levels of a large number of genes simultaneously and to study cellular responses to extracellular stimuli, for example the impact of medication.
DNA-Microarrays consist of microscopic DNA spots attached to a surface. Each spot contains different specific DNA sequences which are used as reporters. These sequences can be short gene sections or other DNA elements that hybridize with cRNA or cDNA. The hybridization of reporter and sample is detected and quantified by fluorophore labeled targets (figure 3) [21].

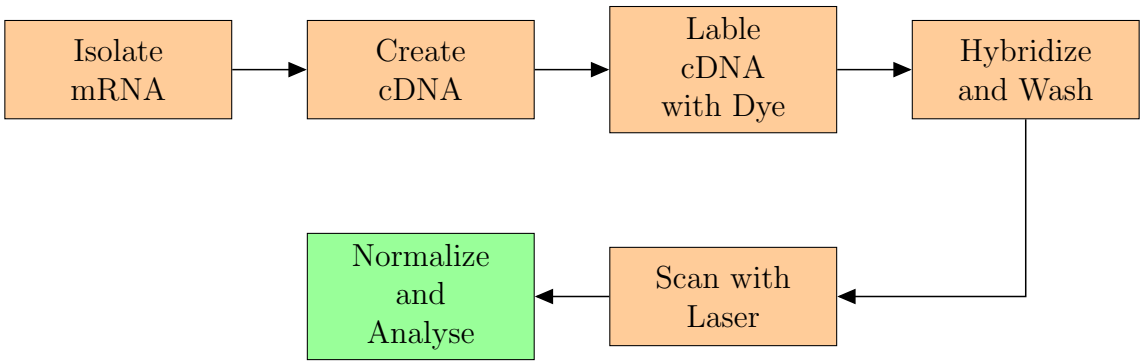


Figure 3: Steps in DNA-Microarray analysis (adapted from [22]).

RNA-seq allows a quantitative measure of gene expressions by resolving the sequence of RNA molecules [23]. The RNA is isolated and transcribed to cDNA and a fragment library is created. A high throughput sequencing method is used to generate $30bp - 300bp$ long reads. After aligning this reads to a reference sequence, an analysis can be done (figure 4) [24, 25].

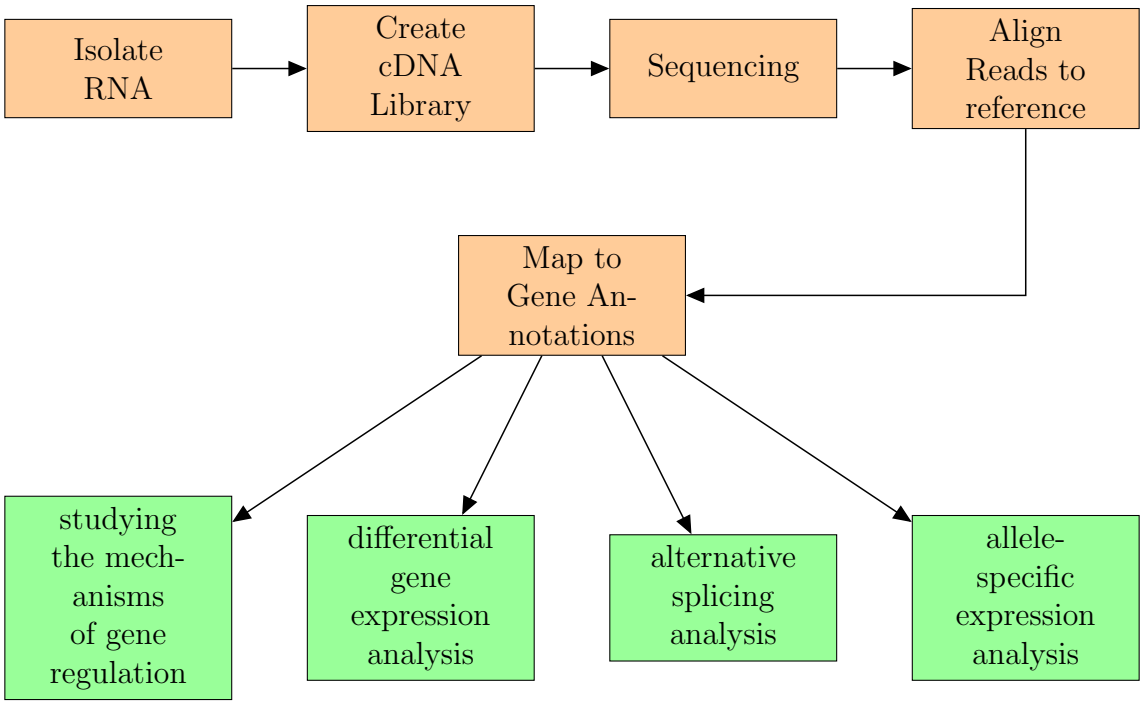


Figure 4: Overall RNA-Seq steps [24–28].

2.2 Integrated Development Environment

The IDE Eclipse *(v.3.8.1-5-1)* [29] with the Build-Management-Tool Maven *(M2E v.4.5.2)* [30] and Subclipse *(v.1.10.13)* were used to implement the extensions of the Genesis software package. Genesis already has an *SVN* repository on the Genome majestix server so it could be easily imported into an Eclipse environment via Subclipse. The IDE is mainly written in Java and commonly used to develop Java applications.

2.3 Genesis

Genesis is a Java tool which supports different visualizing and analyzing options for gene expression data. It was published in 2002 by *Alexander Sturn* [16]. The main function is to cluster large amounts of data. A specific data format called Stanford flat-file is used as input and export file for Genesis. The Graphical User Interface is easy to use and after selecting the preferred clustering method and specifications (Table 1), the clustering can be started. Genesis provides a number of methods which are used to create the similarity matrix for every further clustering. The default setting in Genesis and also the most common method is the Euclidean Distance Measure. The clustering results are shown as heat map and can be easily exported as a *.csv* file. Also preprocessing functions like normalization are implemented.

Table 1: Different menu points and their options in Genesis.

Menu	**Options**
Analysis	Hierarchical Clustering (HCL)
	Self Organizing Maps (SOM)
	k-means Clustering (KMC)
	Support Vector Machines (SVM)
Distance	Pearson Correlation
	Pearson Uncentered
	Pearson squared
	Cosine Correlation
	Covariance value
	Euclidean distance
	Average Dot Product
	Manhattan Distance
	Chebychev Distance
Adjust	Log2 Transform
	Log10 to Log2
	Log10 Transform
	Normalize
	Divide by RMS
	Divide by SD
	Divide by Variance
	Mean Center
	Median Center
	Make Digital
Linkage (HCL)	Average Linkage (WGPMA)
	Complete Linkage
	Single Linkage
	Centroid Linkage
	Median Linkage

2.4 Mahalanobis Distance

Compared to the Euclidean distance the Mahalanobis distance accounts for variation in different directions and also the covariance between variables. To get the distance $d_{(\underline{x},\underline{y})}$ between two data points $(\underline{x}, \underline{y})$ it is necessary to know the points $\underline{x}$ and $\underline{y}$ and S^{-1} the inverse covariance matrix of the whole dataset. The two points $\underline{x}$ and $\underline{y}$ are two vectors,

representing coordinates of the points x and y, $d_{(\underline{x},\underline{y})}$ is the distance between these two points. To calculate the inverse of the covariance matrix S it is necessary to calculate the determinant which can get zero. If it is zero, the matrix is non invertible and the calculation of the covariance matrix determines in a zero division.
Equation (1) is used to calculate the starting distances between every data point and create the similarity matrix. In this equation $(\underline{x} - \underline{y})^T$ is the transposed of the difference between $\underline{x}$ and $\underline{y}$ and S^{-1} is the inverse of S.

$$d_{(\underline{x},\underline{y})} = \sqrt{(\underline{x} - \underline{y})^T S^{-1} (\underline{x} - \underline{y})} \tag{1}$$

2.5 Ward's Linkage

Ward's linkage, also called Ward's minimum variance, tries to minimizes the total cluster variance. Ward's linkage can be calculated according to Equation (2). The variables i and j are the denote indices of the cluster which are used for calculation, $\overline{i}$ and $\overline{j}$ are the center points of the clusters and d represents the Euclidean distance between the center points $\overline{i}$ and $\overline{j}$. The n_i and n_j refers to the number of points in the clusters.

$$d_{Ward(i,j)} = \frac{d^2_{(\overline{i},\overline{j})}}{\frac{1}{n_i} + \frac{1}{n_j}} \tag{2}$$

2.6 General Linkage Equation

All commonly used linkage methods can be described by a general linkage Equation (3).

$$d_{k(i,j)} = \alpha_i d_{ki} + \alpha_j d_{kj} + \beta d_{ij} + \gamma \left| d_{ki} - d_{kj} \right| \tag{3}$$

The implementation of these methods is done by setting the variables α_i, α_j, β and γ to the values given in Table 2. Equation (3) consists of the calculated distance which is $d_{k(i,j)}$ and four other values. The factors i and j are the clusters and that means that there are different $\alpha's$ for the two clusters which are used for the distance calculation. k is defining the new cluster and so $d_{k(i,j)}$ is the new distance of k which consists of the two clusters i and j. Due to the structure of this recursive definition, it was very easy to include additional algorithms like Median-, Group Average-, Flexible- and Centroid-linkage.
Cluster creation is done by merging the two clusters with the smallest distance and re-calculating the similarity matrix for the new cluster [31, 32].

Table 2: Parameters used for different distance calculations [31]. α, β and γ are parameters from equation (3), n_i, n_j and n_k are the numbers of data points in different clusters. (Group Average = UPGMA, Weighted Average = WPGMA)

Linkage	$\boldsymbol{\alpha_i}$	$\boldsymbol{\alpha_j}$	$\boldsymbol{\beta}$	$\boldsymbol{\gamma}$
Group Average	$\frac{n_i}{n_i+n_j}$	$\frac{n_j}{n_i+n_j}$	0	0
Weighted Average	$\frac{1}{2}$	$\frac{1}{2}$	0	0
Median	$\frac{1}{2}$	$\frac{1}{2}$	$-\frac{1}{4}$	0
Centroid	$\frac{n_i}{n_i+n_j}$	$\frac{n_j}{n_i+n_j}$	$-\frac{n_i n_j}{(n_i+n_j)^2}$	0
Ward's	$\frac{n_i+n_k}{n_i+n_j+n_k}$	$\frac{n_j+n_k}{n_i+n_j+n_k}$	$-\frac{n_k}{n_i+n_j+n_k}$	0
Flexible	$\frac{1}{2}(1-\beta)$	$\frac{1}{2}(1-\beta)$	$]-\infty,1[$	0
Single	$\frac{1}{2}$	$\frac{1}{2}$	0	$-\frac{1}{2}$
Complete	$\frac{1}{2}$	$\frac{1}{2}$	0	$\frac{1}{2}$

2.7 Hierarchical Clustering

In Hierarchical Clustering (HCL) each point represents a cluster. This method does not create a specific number of clusters but it creates a hierarchical tree structure which combines always the two closest clusters to a new cluster. In the first step the distances for all points are calculated based on a certain distance measure. After this part, a linkage type which is used for calculating the new cluster distance is chosen. Then the two closest clusters are merged and a new distance is calculated based on the linkage function. The new cluster is added to the distance matrix and the next closest points are merged. These steps are repeated until all points are merged and just a single cluster is left. The created tree can now be used to cut the branches at a certain distance to form distinct clusters.

Data: the $n * n$ distance matrix $S_{x,y}$
Result: Dendrogram C
$U \leftarrow \{1, \ldots, n\}$; /* elements not in a cluster */
while $U \neq Empty$; /* not assigned elements exist */
do
  $U_{S_{min}} = min\{S_{x,y}\}$; /* find minimum distance */
  $U_{S_{next}} = min\{S_{x,y} \cap U_{S_{min}}\}$; /* find nearest point */
  $C \leftarrow U_{S_{min}} \cup U_{S_{next}}$; /* create dendrogram */
  $U \leftarrow U \setminus \{U_{S_{min}} \cup U_{S_{next}}\}$; /* remove C from U */
  $S_{x,y} \leftarrow S_{x,y} \setminus \{S_{min} \cup S_{next}\}$; /* adapt distance matrix */
end

Algorithm 1: Pseudocode for the HCL algorithm

2.8 Partitioning Around Medoids

Partitioning Around Medoids (PAM) [17] is very similar to k-means [17] clustering. Both algorithms require the expected number of clusters as input information. The main difference is that PAM does not use the mean value as center but it uses the closest data point to the mean value as cluster center. At the beginning of the clustering, k number of cluster centers are randomly initialized (Figure 5b). All points are assigned to the closest center (Figure 5c). For each point of the cluster a cluster weight is calculated. The point with the smallest weight is now chosen as new center point (Figure 5d). This is done for all clusters and if each cluster has his center point set, all data points are again assigned to the nearest center (Figure 5e). This process is repeated till no center changes occur (Figure 5f). A maximum number of iterations has to be defined to guarantee termination in case of a non converging calculation.

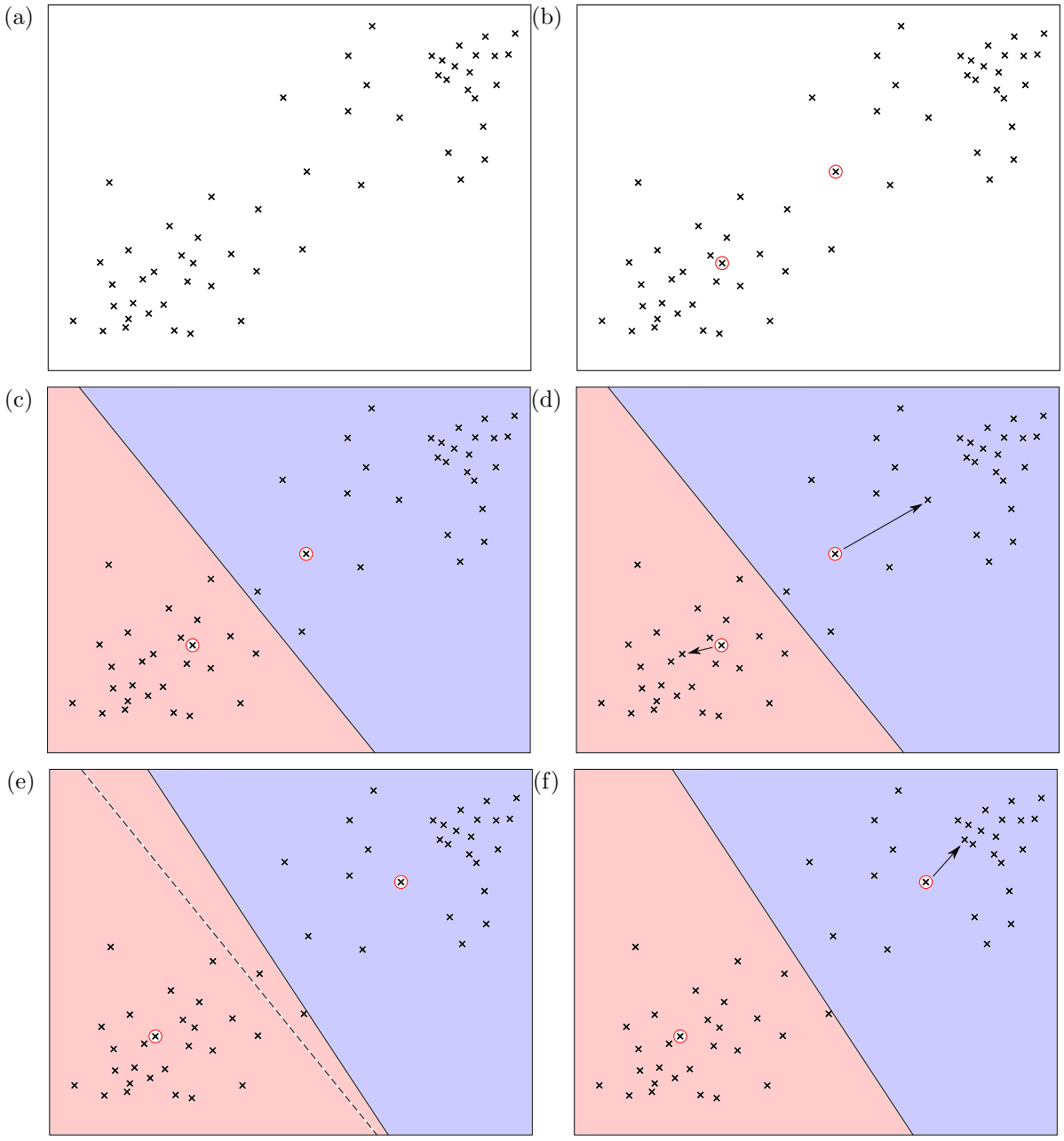


Figure 5: Shows the different steps for PAM clustering. (a) 2D Dataset, (b) random selection of two cluster centers, (c) assign datapoints into two groups according to minimal distance to center, (d) recalculating new center, (e) reassigned data points, (f) calculating new center

Data: the number of clusters k and the $n * n$ distances S
Result: Clusters C_k
$C \leftarrow k$; /* create k cluster centers */
$C_k \leftarrow U \mid min\{S_{x,k}\}$; /* assign all points to centers */
/* with min distance */
while $C_k \neq C_{k_{new}}$; /* cluster center change */
do
  $C_k = C_{k_{new}}$; /* set clusters equal */
  $C_{k_{new}} = min\left\{\sum_{U \in C_k} S_{x,k}\right\}$; /* calculate new center */
  /* for each cluster C_k */
  $C_{k_{new}} \leftarrow U \mid min\{S_{x,k}\}$; /* assign all points to new centers */
  /* $C_{k_{new}}$ with min distance */
end

Algorithm 2: Pseudocode for PAM algorithm.

2.9 Transitive Clustering

Transitive Clustering is a graph based method which tries to create clusters depending on a threshold $[0, 1]$ that is given as input parameter. The weights are calculated from the similarity matrix which are normalized to $[0, 1]$ and then inverted so that a low similarity represents a high weight. Now the similarity matrix can be used to create edges between all points in the dataset. Each edge between two points is weighted with the similarity calculated from the distance.

Edges whose weight is lower than the threshold can be deleted (Figure 6a). This leads to a non transitive graph with missing edges (Figure 6b). The following step tries to build transitive clusters via adding and removing edges with the minimum change in the sum of removed or added weights (Figure 6c). The finished clusters are formed by transitive graphs (Figure 6d). Transitive means that each point is connected with each point of its cluster.

After removing edges from each point which are below the threshold there is a number of possibilities to make the clusters transitive again. This is a very time consuming procedure and so the CAST algorithm [18] was used for implementation. This Cluster Affinity Search Technique (CAST) algorithm is a deterministic procedure which tries to find all clusters one by one. The pseudo code in algorithm 3 explains the basic function. The affinity a_x of an element x is defined in equation 4. The affinity between two elements is high if the weight is higher than the threshold t, otherwise it is low. The CAST algorithm tries to open just one cluster at a time, adds or removes elements until a stable state is achieved, that means no element can be added or has to be removed. The current cluster is closed and a new one is opened and all elements without a cluster are processed

again. With this approach it is not possible to say how many clusters will be created. The weights in the following equations are referred as S

$$a_x = \sum_{y \in C_{open}} S_{x,y} \tag{4}$$

After adding or removing elements, the weights of all points have to be recalculated.

$$a_x = \left(\sum_{y \in C_{open}} S_{x,y} \right) \pm S_{x,y} \tag{5}$$

$$a_x = a_x \pm S_{x,y} \tag{6}$$

Note that this scheme is just a simplified example and that the real data processing has to be done in n dimensions and is far more complex.

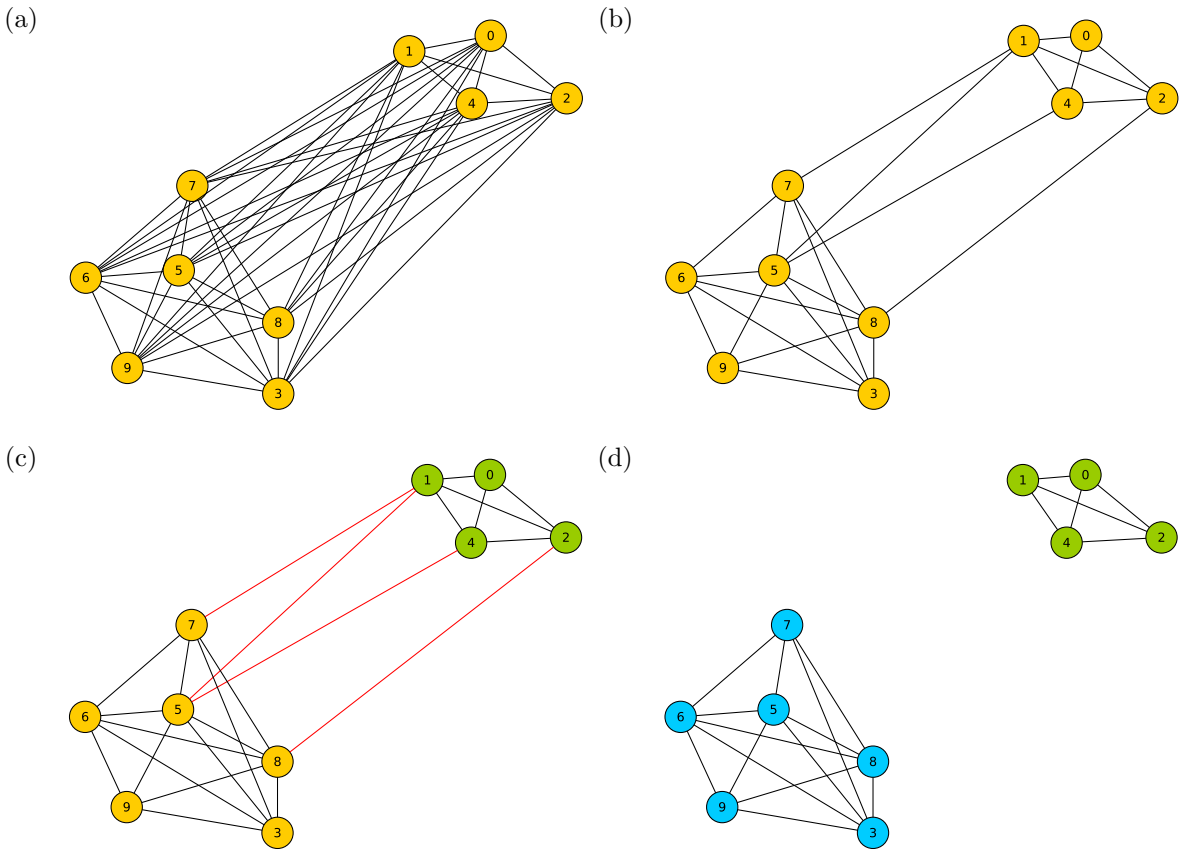


Figure 6: (a) The similarity matrix interpreted as a bidirectional graph, edge length is inverse proportional to similarity. (b) All edges for each point under a certain threshold, specified before execution, are removed. (c) The green nodes already are transitive, removing the red edges is the cheapest option to create a transitive cluster. (d) After removing the red edges, two transitive clusters remain.

Data: An $n * n$ similarity matrix S and a threshold t.
Result: Clusters
$C \leftarrow 0$; /* all clusters closed */
$U \leftarrow \{1, \ldots, n\}$; /* elements not in a cluster */
while $U \neq 0$ **do**
  $C_{open} \leftarrow 0$; /* create a new cluster */
  $a(.) \leftarrow 0$; /* reset affinity */
  Repeat add and remove till no changes occur.
  while $max\{a_u \mid u \in U\} t \geq \mid C_{open} \mid$; /* adding step */
  do
    take the element $u \in U$ *with maximum affinity* t
    $C_{open} \leftarrow C_{open} U\{u\}$; /* add u into C_{open} */
    $U \leftarrow U \setminus \{u\}$; /* remove u from U */
    for all $x \in U \cup C_{open}$ *set* $a_x = a_x + S_{x,u}$; /* update affinity */
  end
  while $min\{a_u \mid u \in C_{open}\} t < \mid C_{open} \mid$; /* removing step */
  do
    take the element $u \in U$ *with minimum affinity* t
    $C_{open} \leftarrow C_{open} \setminus \{u\}$; /* remove u from C_{open} */
    $U \leftarrow U \cup \{u\}$; /* add u into U */
    for all $x \in U \cup C_{open}$ *set* $a_x = a_x - S_{x,u}$; /* update affinity */
    $C \leftarrow C \cup \{C_{open}\}$; /* close the cluster */
  end
end

Algorithm 3: Pseudocode for the CAST algorithm by Ben-dor [18]

2.10 F_1 - score

The F_1-score is one of the most common measures to test accuracy of clustered data. To calculate the F_1-score a true classification called Gold Standard must be known. It considers precision and recall to calculate the score and can be interpreted as weighted average of these two measures [33]. The best F_1-score is 1 and the worst is 0, equation 7 shows that the score is a harmonic mean of precision and recall.

$$F_1 = 2 \cdot \frac{precision \cdot recall}{precision + recall} \tag{7}$$

The precision [34] is the amount of true positive values (TP) divided by the number of all positive results (equation 8). Recall [34] is defined as the amount of true positive values divided by the number of positive results that should have occur. Table 3 shows the used values for the precision and recall calculations.

$$precision = \frac{TP}{TP + FP} \tag{8}$$

$$recall = \frac{TP}{TP + FN} \tag{9}$$

Table 3: Stating the predicted conditions versus the real conditions [33].

		predicted condition	
		positive	negative
real	condition positive	TP	FN
	condition negative	FP	TN

TP... True Positive, FN... False Negative
FP... False Positive, TN... True Negative

2.11 Matthews Correlation Coefficient

The Matthews Correlation Coefficient (MCC) is calculated taking all true and all false states into account and is regarded as a balanced measure. This means that it gives good results also if the clusters are not equally sized [35]. The maximum is 1 which represents a perfect clustering result, 0 means that the result is not better than a random classification and -1 represents an result which is the exact opposite of the gold standard.

$$MCC = \frac{TP \cdot TN - FP \cdot FN}{\sqrt{(TP + FP) \cdot (TP + FN) \cdot (TN + FP) \cdot (TN + FN)}} \tag{10}$$

2.12 Silhouette Value

The Silhouette value [36] is a good measure for the quality of a clustering result if no gold standard is available for comparison. It is based on comparing "tightness" and "separation" of clusters and represents an estimate of the cluster quality. Equation 11 is used to calculate the Silhouette values as proposed by Pablo Jaskowiak et al. [37].

$$S = \frac{1}{n} \sum_{i=1}^{n} \frac{b_{x_i} - a_{x_i}}{max\,(a_{x_i}, b_{x_i})} \tag{11}$$

In this equation a_{x_i} represents the average distance of a cluster center x and all the points from its cluster. b_{x_i} is the mean distance from the previous cluster center x and all distances to the points of the closest cluster. The range is in the interval $[-1, 1]$ and like the MCC, 1 means very high dependencies, 0 means very low dependency and a negative value gives a opposite dependency.

2.13 ClustEval

ClustEval is a free platform for comparing clustering performance developed by Christian Wiwie et al. [14]. It supports a large variety of clustering methods and also a good library of test sets and their gold standards. This platform is used to get test sets and reference F_1-*scores*. The gold standard files were used to evaluate the results from Genesis. A virtual machine image file was provided by the *ClustEval* project which can be easily used by any Virtual Machine. In this thesis $VirtualBox$ [38] version 4.3.36_$Ubuntur$105129 and the image file *version* 1.5 (07.08.2016) [14] was used.

2.14 Tools R & PAST

For the different clustering calculations, *R (v.3.3.2)* [12] and PAST *(v.3.14)* [4] were used. The *R* Packages are listed in Table 4. This packages were needed for the different distance calculations and clustering methods but also for the heatmap drawing. The R-script which calculates the heat maps for comparison is available in the Appendix. The heatmaps in *R* were created with the function *heatmap*.2 and the color settings were changed so that the output resembles the look of the Genesis heatmap.

Table 4: Packages, used version including a short description and their reference.

Package	Version	Description	Ref.
gplots	3.0.1	In this package the function *heatmap.2* was used to create the heatmap, the dendrogram and the whole description.	[39]
cluster	2.0.5	This package was needed because it provided the function *agnes* which is necessary to calculate average, single, ward, weighted, complete and flexible linkage.	[13]
stats	3.3.2	In this package, the function *hclust* provided clustering with median and centroid linkage. It also provided the possibility to use self calculated distance measures and so Mahalanobis Distance was implemented with the help of *cor*, *dist*, *t* and two other functions from the *base* package.	[40]
base	3.3.2	This is the main R package which provided most of the standard functionalities. The functions $forwardsolve$ and *chol* were used for calculation of the Mahalanobis Distance.	[41]
ape	5.1	This package is used for the PH85 tree validation.	[42]

To create Figure 2 a *csv* file from PubMed was downloaded. If a search for specific keywords is done, this *csv* can be downloaded easily on the top right corner of the web page. This *csv* was used by the LaTeX Function *tikzpicture* to read the data and create the diagram.

2.15 Datasets for Validation

Six Datasets were used for validation: *Fibroblast_red*, *Fibroblast_red_ns*, *Fibroblast_ns*, *Dataset 1*, *Dataset 2* and *Dataset 3*. The fibroblast datasets were used to verify the MAhalaanobis distance and the linkage methods with *PAST* [4] and *R* [12]. It was based on an example file available in the standard Genesis workspace named *Fibroblast* [15, 43]. This dataset shows data from a gene expression experiment and in the beginning of each line there is a unique ID for each gene. The other entries are showing a weight an order and the expression rates. In microarray analysis fluorescent intensities represent the value for the amount of transcripts and for RNA-seq it is the number of mapped reads. This data are created by a cDNA microarray and show how strong specific genes are expressed. To be able to create heatmaps with less data points, this dataset was shortened to 28 data rows and is called *Fibroblast_red* (*reduced*), the number of data

points (columns) was not changed and is 22. It has to be mentioned, that the first two columns contain a unique ID and the name of the transcript. The third column is a weight and the fourth column gives an order. A second test set was created because the covariance matrix of the original data matrix was singular. By deleting the zero time column, a new dataset named *Fibroblast_red_ns* (*not singular*) was created. Also a bigger set of data was used to test the implementations. For this purpose, the original *Fibroblast* data set without the zero time column called *Fibroblast_ns* was created.

Table 5: Used datasets and their sizes.

Name	Notice	Rows	Cols.	Clust.	Ref.
Fibroblast_red	reduced Genesis standard example file	30	22	3	[15, 43]
Fibroblast_red_ns	reduced Genesis standard example file, not singular TP 0 excluded	30	22	3	[15, 43]
Fibroblast_ns	original sized Genesis standard example file, not singular TP 0 excluded	519	22	3	[15, 43]
Dataset 1	NCBI GEO database, shortened in rows for easier comparison, no classification given	58	18	-	[44, 45]
Dataset 2	ClustEval dataset with given correct classification	34	999	3	[14, 46]
Dataset 3	completely synthetic created, true classification is available	1024	32	16	[47]

The used dataset was made available for public by the Broad Institute [46] and consists of 34 samples of acute leukemia with three different subtypes. Each sample contains microarray gene expression levels of 999 genes. Also a gold standard is provided to this dataset so that a quantification of results can be done easily [48]. A part of the *Fibroblast.txt* dataset is represented in Table 18 in the Appendix. The datasets were used to verify the implementation of PAM and TC with ClustEval. *Dataset 1* is taken from NCBI GEO Database [44] and has no gold standard file. This set is reduced to 58 data points and is adapted to the *Genesis* input format. Table 19 shows a part of the NCBI GEO [44] data set which is further used for validation. *Dataset 2* is taken from ClustEval [14] and consists of a header entry which specifies format and type, the names and all expression data in lines, separated by tabs. To use this data in *Genesis*, slight

modifications have been made so that this format complies with the *Genesis* input format type. *Dataset 3* is a fully synthetic dataset with 32 columns and 1024 rows consisting of 16 clusters and a correct classification file is available [47]. The Mahalanobis distance and all linkage methods were verified with *PAST* [4] and *R* [12].

2.16 Current Clustering Design in Genesis

Genesis source code is available in an *SVN* on the genome server. The project is reachable over *https://majestix/svn/genome/Genesis/trunk/src/at/tugraz/genome/genesis/*. In this *SVN* folder there is a separate directory for each functionality. All implementations of clustering algorithms are present in *∼/genome/cluster* and have a separate directory with the clustering name, for example *∼/genome/cluster/HCL* for hierarchical clustering or *∼/genome/cluster/KMC* for k-means clustering. There is always a file named like the directory but with a *.java* extension for example *HCL.java* or *KMC.java*. These class files contain the main code for the specific clustering algorithm. Different other *.java* files my be present which are needed to display output or process the data, they do not follow any naming convention. Table 6 gives a short overview of the folders and their main tasks.

Table 6: Source code structure in the *src* directory with the prefix *at.tugraz.genome.*"*Name*"

Names	**Short Description**
genesis	Contains all files which are needed for the clustering algorithms and also for saving and processing the data. All graphical output for the processed data and images are also created and stored in this folders.
go	This folder contains all functionalities of the *GO* function. It is used for the go mapping utility and has a own folder with images.
test	In this folder there is a test function for testing the functionality of the clustering methods. It is a static example which creates always the same result.
util	This folder contains all important utilities which are necessary for other files.

3 Results

All datasets were clustererd with different methods and all results are listed in this section and also general Informations about speed and memory are given. The Mahalanobis Distance [49] method was added, it can be used for calculating the starting distances for data points of all different clustering methods. The implementation of PAM [50] and TC [34] extends the possibilities of clustering. The results were compared with ClustEval [44] and some internal cluster measures were calculated to proof the functionality. The results from the implementations were compared with the outcome of PAST [4] and R [12] to verify that the added code works properly.

3.1 Mahalanobis Distance

The implementation of Mahalanobis Distance measure [49] was done in the overall distance measure menu bar to provide access for all clustering algorithms (figure 7). The Mahalanobis distance measure implementation was added to the $ExpressionMatrix.Java$ which already contained all other distance measure algorithms. The output from $PAST$ and R was compared with the dendrogram created by Genesis (Figure 8). As PAST had an unsatisfying graphical quality and the results were the same as in R only the dendrograms created with R were used. Because the Mahalanobis Distance measurement is not defined for all data sets it was important to create at least two different test sets. If the covariance matrix is singular the determinant will be zero and so the inverse can not be calculated. For this reason the exception was caught and the whole calculation was aborted and a message explaining the problem was displayed. The dataset which was used for comparing the results was $Fibroblast\_red\_ns$. Table 7 shows calculation times for the distance matrix calculation for different distance functions. To compare the created dendrograms the PH85 method proposed by Robinson et al. [51] from the R package ape was used. Both trees were entered manually and if the weight of the edges are not taken into account, the result is 0 which means the the trees are equal.

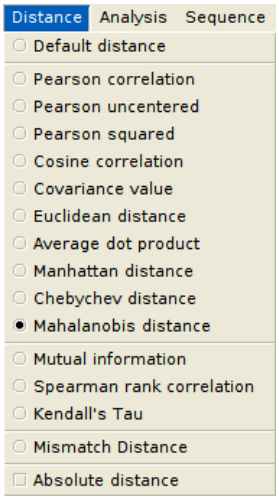


Figure 7: Hierarchical Clusterng menu with the different linkage types.

Table 7: Times for calculating the distance for different measures, average of 10 measurements on the *Fibroblast_ns* dataset.

Distance	Time [ms]	SD [ms]
Euclidean	10.50	0.27
Covariance	11.13	0.29
Cosine	16.67	0.31
Mahalanobis	1407.06	16.82

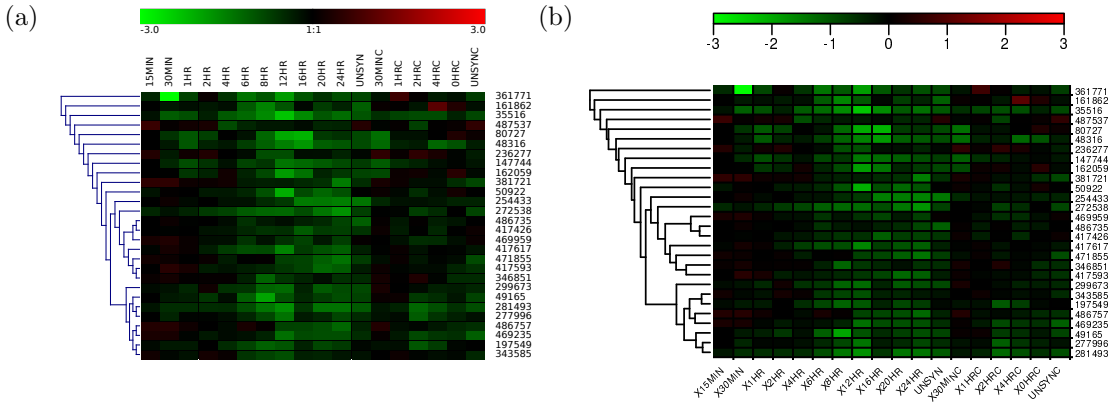


Figure 8: Heatmap and the resulting dendrogram for the *Fibroblast_red_ns* file, with Mahalanobis distance and Single Linkage clustering, generated by Genesis (a) and generated with *R* (b).

3.2 Ward's Linkage

The Ward's method for calculating the similarity matrix was implemented using the recursive formula of Lance and Williams [52] and was added to the distance calculating menu for hierarchical clustering (Figure 9). Because the recursive formula consists of different variables which can be used to calculate other linkage values also Average-, Complete-, Single-, Average Group-, Centroid-, Median-, and Flexible- linkage were implemented or changed. When all factors (Table 2) are set, the recursive formula is used to calculate the distances. This functions were

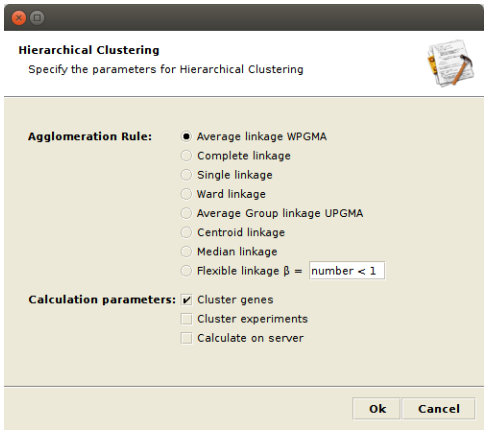


Figure 9: Hierarchical Clusterng menu with the different linkage types.

added as selectable option in the hierarchical clustering menu and for flexible linkage, β was added as an input field in the range $]-\infty, 1[$. The resulting dendrograms are shown in Figure 9. The heatmaps with the corresponding dendrogram calculated by Genesis are depicted on the left side. The right side shows the reference figures calculated with R for the same clustering algorithm and the same starting distance. The PH85 value was calculated for the pairs and yielded always 0. These extensions were added to the existing program classes so that no new classes or structures were required.

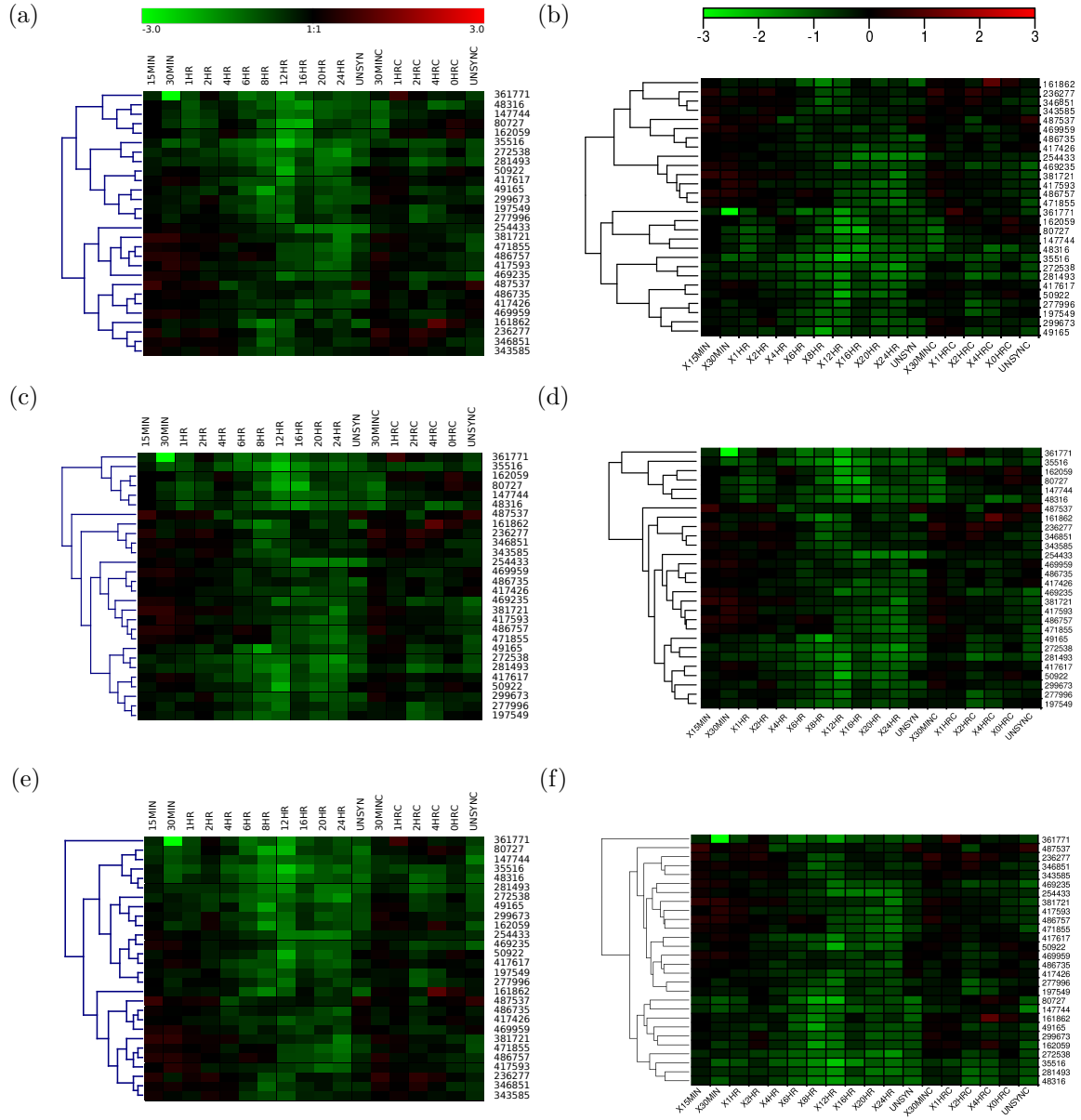


Figure 9: Heatmaps from clustering of the *Fibroblast_red_ns* file with different linkage methods by Genesis (a,c,e) and R (b,d,f). (a) Ward's Linkage Genesis, (b) Ward's Linkage R,(c) Flexible Linkage $\beta = 0$ Genesis, (d) Flexible Linkage $\beta = 0$ R, (e) Group Average Linkage Genesis, (f) Group Average Linkage R

3.3 Integration of PAM and TC

These two algorithms were implemented due to good clustering results described in the publication by Wiwie et al. [14]. For the implementation of partitioning around medoids (PAM) [17] and transitive clustering (TC) [18], new data structures and classes were necessary. Two new folders named *PAM* and *TC* were added in *at.tugraz.genome.genesis.cluster* containing the main files *PAM.java* and *TC.java*, respectively since *TC* clustering is graph based, the file structure looks similar to the Hierarchical Clustering files but some classes were modified. The *PAM* algorithm is a variation of *k-means* clustering and so the already existing structure of *KMC* was used.

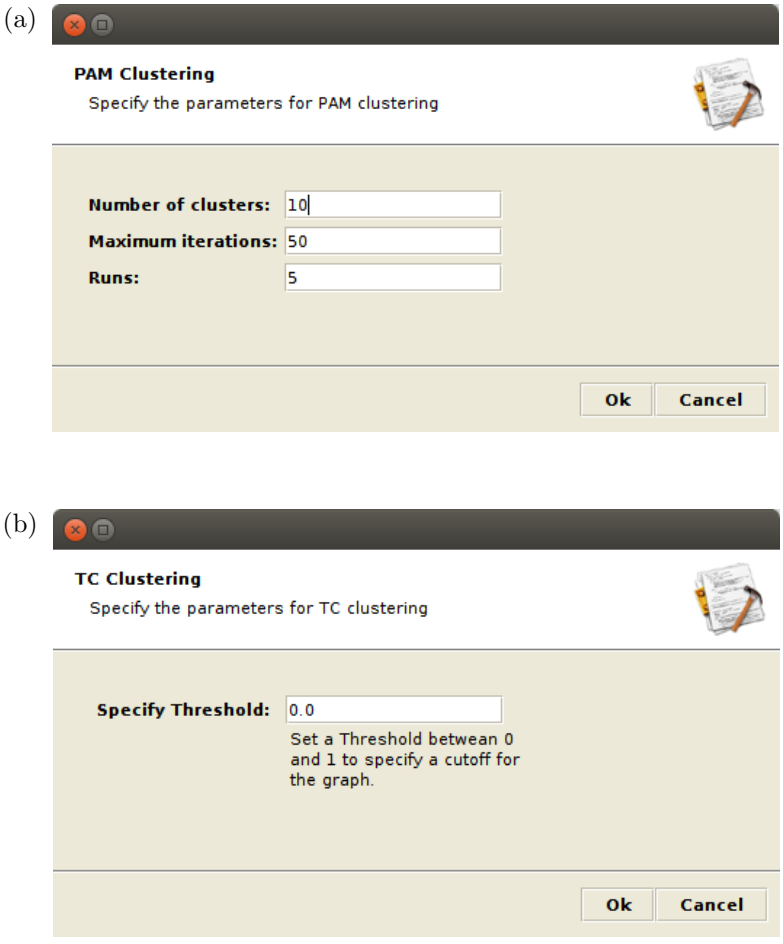


Figure 10: The PAM Clustering menu (a) and the TC Clustering menu (b).

3.4 Results for *Dataset 1*

Dataset 1 was downloaded from the NCBI GEO database [45] and reduced to 58 rows, no classification is given. All clustering results with different parameters and spearman's rank correlation distance measure are displayed in Table 8, this distance measure is used due to good F_1-score [37].

Table 8: Clustering results for *Dataset 1* using TC and PAM with different clustering parameters. The first two settings give two clusters and the second two settings give three clusters.

Name	**TC T=0.4**	**PAM k=2**	**TC T=0.5**	**PAM K=3**
A_16_P00403029	2	2	2	2
A_16_P21621957	1	1	1	3
A_16_P00072385	1	2	2	2
A_16_P59800141	1	1	1	1
A_18_P15785134	1	1	1	1
A_16_P19296953	1	1	1	1
A_16_P59922478	2	2	2	2
A_16_P03462470	1	1	1	1
A_16_P57228325	1	1	1	3
A_16_P03411893	2	2	2	2
A_16_P03096255	1	1	1	3
A_16_P41087272	1	1	1	3
A_16_P01065718	2	2	2	2
A_16_P18388086	1	1	1	1
A_16_P03385907	2	2	2	2
A_16_P16672026	1	1	1	1
A_16_P21424156	1	1	1	1
A_16_P19791432	2	2	2	2
A_16_P20293969	1	1	1	1
A_16_P18653359	1	2	2	3
A_16_P18101626	1	1	1	1
A_16_P18783515	1	2	2	2
A_16_P01721093	1	1	1	3
A_16_P37193919	2	2	2	2
A_18_P15161873	1	2	2	3

Table 8: Continued

Name	**TC T=0.4**	**PAM k=2**	**TC T=0.5**	**PAM K=3**
A_16_P60434836	2	2	2	2
A_16_P38381210	2	2	2	2
A_18_P13634897	1	1	1	1
A_16_P38634204	1	1	1	3
A_16_P18711285	2	2	2	2
A_16_P15059815	1	2	1	2
A_16_P19447663	1	1	1	1
A_16_P56078638	1	1	1	3
A_16_P31322059	1	2	3	3
A_18_P11116317	1	1	1	3
A_16_P36566016	2	2	2	2
A_16_P20072978	2	2	2	2
A_18_P12605410	1	1	1	1
A_16_P15496691	1	1	1	3
A_16_P21098513	1	2	2	2
A_16_P16192764	1	1	1	1
A_16_P19493848	1	1	3	1
A_18_P15756033	2	2	2	2
A_16_P58316227	2	2	2	2
A_16_P02101244	1	1	1	3
A_18_P14664614	1	1	1	1
A_16_P59590588	1	2	2	2
A_18_P14977561	1	2	3	3
A_16_P17103965	1	1	1	1
A_16_P20973757	1	1	1	1
A_16_P18937510	1	1	1	1
A_16_P17423087	2	2	2	2
A_16_P18159690	1	1	1	1
A_16_P00242008	2	2	2	2
A_16_P03790905	1	2	2	2
A_16_P17381660	1	2	3	1
A_16_P40212811	1	2	2	2
A_16_P21320988	1	1	1	1

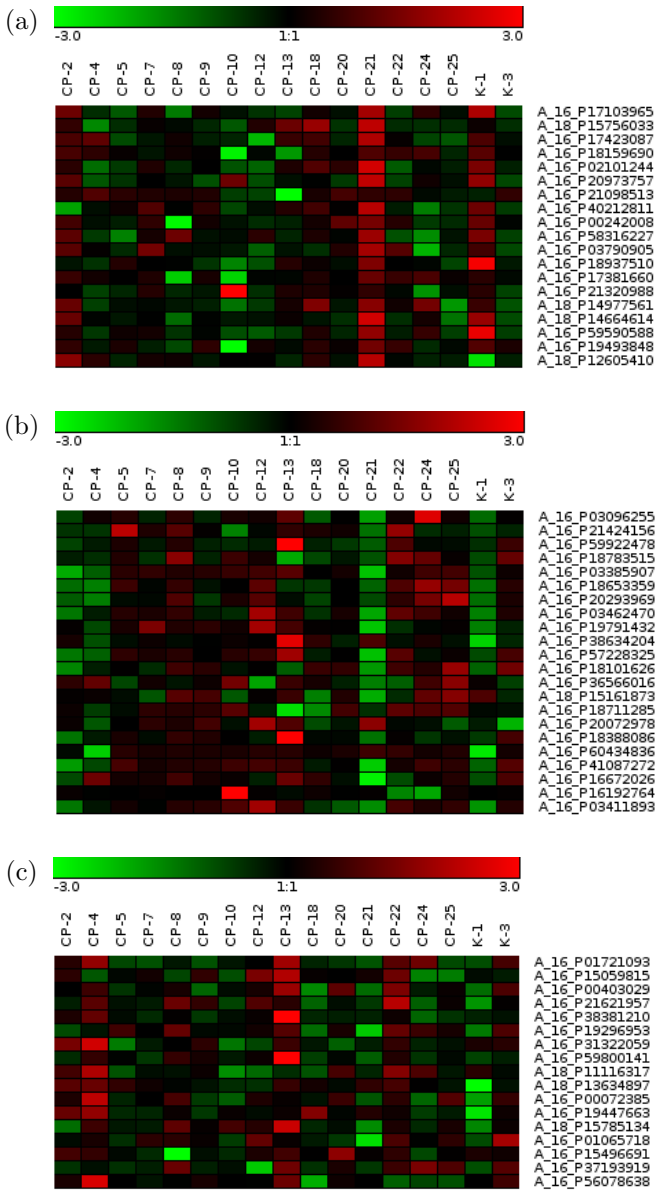


Figure 11: This three heatmaps show the graphical result for *Dataset 1*, PAM Clustering and three clusters which is created by Genesis.

3.4.1 Silhouette Value

The Silhouette values of the clustering results were calculated using equation 11. Figure 11 shows the graphical result. Table 9 summarizes Silhouette values for different input parameters for *Dataset 1*.

Table 9: Silhouette values for *Dataset 1* with different cluster parameters.

Clustering	**Silhouette value Genesis**
TC ($T = 0.4$)	0.5828
TC ($T = 0.5$)	0.4524
PAM ($k = 2$)	0.5788
PAM ($k = 3$)	0.4760

Tables 20 to Table 25 in the Appendix section are showing the Silhouette values for different clustering methods, amount of clusters and test sets calculated from the results above.

3.5 Results for *Dataset 2*

This dataset is from the ClustEval project [14, 46] and has a annotation which can be used for further classification. The same parameter as in ClustEval were used to get comparable clustering results. Table 10 shows the clustering results for this dataset. Moreover on this parameters were used to calculate the values in Table 11 with equations 8 and 9. These values were used to calculate the F_1-score and MCC (Table 13).

Table 10: Cluster assignment of samples from *Dataset 2* and two clustering methods with the parameters which gave the best approach in *ClustEval* and Spearmans Rank Correlation and also for the already implemented methods HCL and k-means.

	Names	**Gold - Standard**	**TC T = 0,5050505**	**PAM k = 3**	**HCL**	**k-means k = 3**
Group 1	ALL_9692_B.cell	1	1	1	1	1
	ALL_9692_B.cell_2	1	3	1	3	1
	ALL_17281_B.cell	1	1	1	1	2
	ALL_17281_B.cell_2	1	1	1	1	2
	ALL_20414_B.cell	1	1	1	1	2
	ALL_20414_B.cell_2	1	1	1	2	2
	ALL_549_B.cell	1	1	1	1	1
	ALL_549_B.cell_2	1	1	1	1	1
	ALL_20185_B.cell	1	1	1	1	2
	ALL_20185_B.cell_2	1	1	1	1	2
	ALL_18239_B.cell	1	1	1	1	1
	ALL_18239_B.cell_2	1	1	1	1	2
	ALL_7092_B.cell	1	3	2	3	1
	ALL_7092_B.cell_2	1	1	1	1	1
	ALL_R23_B.cell	1	1	1	1	1
	ALL_R23_B.cell_2	1	3	2	2	1
Group 2	ALL_19881_T.cell	2	2	2	2	2
	ALL_19881_T.cell_2	2	2	2	2	2
	ALL_9723_T.cell	2	2	2	2	2
	ALL_9723_T.cell_2	2	2	2	2	2
	ALL_14402_T.cell	2	2	2	2	2
	ALL_14402_T.cell_2	2	2	2	2	2
	ALL_22474_T.cell	2	2	2	2	2
	ALL_22474_T.cell_2	2	3	3	3	3
Group 3	AML_13	3	3	3	3	3
	AML_13_2	3	3	3	3	3
	AML_16	3	3	3	3	3
	AML_16_2	3	3	3	3	3
	AML_1	3	3	3	3	3
	AML_1_2	3	3	3	3	3
	AML_3	3	3	3	3	3
	AML_3_2	3	3	3	3	3
	AML_6	3	3	3	3	1
	AML_6_2	3	3	3	3	3

Table 11: TP, FN, FP and TN values for all results.

	TC T = 0.5050505				**PAM k = 3**			
	TP	FN	FP	TN	TP	FN	FP	TN
Group 1	13	3	0	18	14	2	0	18
Group 2	7	1	0	26	7	1	2	24
Group 3	10	0	4	20	10	0	1	23
Sum	**30**	**4**	**4**	**64**	**31**	**3**	**3**	**65**

Table 12: TP, FN, FP and TN values for all results.

	HCL				**k-means k = 3**			
	TP	FN	FP	TN	TP	FN	FP	TN
Group 1	12	4	0	18	14	2	9	17
Group 2	7	1	2	24	1	7	4	24
Group 3	10	0	3	21	115	9	3	21
Sum	**29**	**5**	**5**	**63**	**30**	**18**	**26**	**62**

Table 13: F_1-score and MMC for the two clustering methods, plus HCL and k-means, and different input parameters for *Dataset 2* with *Genesis* [15] and ClustEval [14].

Name	**F_1-score Genesis**	**F_1-score ClustEval**	**MCC Genesis**	**MCC ClustEval**
TC ($T = 0.5050505$)	0.8824	0.933	0.820	0.880
PAM ($k = 3$)	0.9118	0.921	0.866	0.783
HCL	0.8529	–	0.779	–
k-means ($k = 3$)	0.7225	–	0.320	–

Table 14: Silhouette values for two clustering methods and different input parameters with Genesis [15] and ClustEval [14] for *Dataset 2*.

Clustering	**Silhouette value Genesis**	**Silhouette value ClustEval**
TC ($T = 0.5050505$)	0.3549	0.3600
PAM ($k = 3$)	0.3425	0.3520
HCL	0.3395	–
k-means ($k = 3$)	0.2980	–

3.6 Results for *Dataset 3*

This dataset is a synthetic one and the largest with a known annotation. Since this dataset is very large, no graphical results are omitted. Table 15 lists values relevant to calculate the F_1-score and MCC which are depicted in Table 16. Table 15 also states the values for k-means to have a comparable result.

Table 15: TP, FN, FP and TN values for all results.

	TC T = 0.5				**PAM k = 16**				**k-means k = 16**			
	TP	FN	FP	TN	TP	FN	FP	TN	TP	FN	FP	TN
Group 1	64	0	0	960	64	0	0	960	64	0	0	960
Group 2	64	0	0	960	64	0	0	960	64	0	0	960
Group 3	64	0	0	960	0	64	29	931	64	0	64	761
Group 4	64	0	0	960	56	8	0	960	64	0	0	960
Group 5	64	0	0	960	64	0	0	960	64	0	0	960
Group 6	64	0	0	960	64	0	64	896	64	0	0	960
Group 7	64	0	0	960	0	64	8	952	0	64	2	958
Group 8	64	0	0	960	64	0	0	960	64	0	0	960
Group 9	64	0	0	960	35	29	0	960	64	0	0	960
Group 10	64	0	0	960	54	10	0	960	64	0	0	960
Group 11	64	0	0	960	64	0	128	832	64	0	64	761
Group 12	64	0	0	960	64	0	0	960	64	0	0	960
Group 13	64	0	0	960	0	64	10	950	64	0	0	960
Group 14	64	0	0	960	64	0	0	960	60	4	0	960
Group 15	64	0	0	960	64	0	0	960	64	0	0	960
Group 16	64	0	0	960	64	0	0	960	0	64	2	958
Sum	**1024**	**0**	**0**	**15360**	**785**	**239**	**239**	**15121**	**892**	**132**	**132**	**14958**

Table 16: F_1-score and MMC for the thee clustering methods for *Dataset 3* with *Genesis*.

Name	**F_1-score Genesis**	**MCC Genesis**
TC ($T = 0.5$)	1.000	1.000
PAM ($k = 16$)	0.766	0.751
k-means ($k = 16$)	0.871	0.862

3.7 Assessment of Clustering Quality

PAM and TC clustering methods were compared with the *ClustEval* package to get comparable results. The parameters were chosen according to the *ClustEval* parameters and so spearman's rank correlation measure was used [37]. All clustering results are depicted in Tables 10, 15 and 15. As mentioned in the methods section above, the F_1-score and

the MMC is calculated and shown in tables 13 and 16.

3.8 Runtimes and Memory

The distance matrix was calculated once for each run and the single distance calculation was done for each comparison step. For *Dataset 1* (1024 rows), the similarity matrix requires approximately $100\mu s$ for *Dataset 2* $767\mu s$ and for *Dataset 3* it takes about $2866\mu s$. The calculation time for TC in general is longer because it needs more time to calculate the edge weight. For *Dataset 1* the similarity matrix is $1024 \cdot 1024$ divided by 2 minus 1042 due to redundancy and the fact that the main diagonal is 1 for each entry. Equation 12 shows the overall calculation for an n rows long dataset. All time and memory results are given in Table 17

$$Size = \frac{n \cdot n}{2} - n \tag{12}$$

Table 17: Time and Memory measures for a single distance calculation and a distance matrix calculation.

		PAM			**TC**		
Dataset	**Name**	**Time**[μs]	**SD**[μs]	**Memory**	**Time**[μs]	**SD**[μs]	**Memory**
Dataset 1	Distance Calc.	100.46	24.50	1997776 *byte* ($\sim 1.9MB$)	998.27	59.51	1997776 *byte* ($\sim 1.9MB$)
	Single Dist.	0.08		4 *byte*	0.08		4 *byte*
Dataset 2	Distance Calc.	767.23	103.91	2663528 *byte* ($\sim 2.5MB$)	1399.52	154.5	2663528 *byte* ($\sim 2.5MB$)
	Single Dist	0.08		4 *byte*	0.08		4 *byte*
Dataset 3	Distance Calc	2866.39	1751.57	2667488 *byte* ($\sim 2.5MB$)	3196.52	269.90	2667488 *byte* ($\sim 2.5MB$)
	Single Dist	0.08		4 *byte*	0.08		4 *byte*

4 Discussion

The amount of gene expression data generated with high-throughput microarray or sequencing technologies is constantly increasing. Therefore analysis methods like clustering and classification is getting more and more important. Genesis [15] is a Java software package which already supports differed clustering algorithms like hierarchical or k-means clustering. To support a wider range of clustering approaches, some extensions were implemented in the course of this thesis.
The Mahalanobis distance measure algorithm and several linkage variants for hierarchical clustering were added and the results were compared with PAST [4] and R [12]. The Partitioning Around Medoids [50] and Transitive Clustering TC[34] methods were added and the results were crosschecked with ClustEval [14]. Clustering quality was assessed with the Silhouette value, the Matthews Correlation Coefficient and the F_1-score

4.1 Implementation

The Genesis project has an SVN Repository which can be loaded with Subclipse into the Eclipse IDE [29] to perform changes. The extensions were implemented following the existing structure. After becoming familiar with the program structure, the implementation was straight forward and except a few small mistakes, no major problems occurred. New classes and folders were added for each clustering method. Existing code could not be reused (e.g. by extending classes), because the algorithms are considerably different from the already implemented ones. As PAM is quite similar to k-means the structural layout was used for implementation.

4.2 Distance Measure

The distance measure is used to calculate a similarity matrix which contains pairwise distances between all data points. This matrix has a redundant half, because it is symmetric along the main diagonal. Additionally the main diagonal elements are zero because the distances between the same points are zero. This allows to considerably reduce the memory, because only the half of this matrix needs to be stored.
Genesis implements a class containing all distance measures, where it calculates the distances but does not store them. This approach has a memory advantage but also a major disadvantage because of increased calculation time. When distances are stored the size of the memory needed is $\frac{n^2}{2} - n$ and n is the number of data points. The k-means method is able to calculate larger datasets but it is also very slow because the time to calculate every distance at its needed time is extremely time consuming. As computational time and memory in general is getting faster and cheaper, the similarity matrix is used for

cluster calculation. Table 17 shows the time and space which is needed for the whole matrix construction and for calculating one distance. We experience speed-ups of factor 10 depending on the size of the dataset and the number of iterations.
In Genesis multiple methods can be selected for the distance calculation. This functions depend on the type of the data which are going to be clustered. In the paper Wiwie et al. [14] Spearman's Rank correlation was used for gene expression data because it yielded the best F_1-score. Also Susmita et al. [53] is using Spearman's Rank correlation on gene expression data due to good results. For this thesis, also the Spearman's Rank correlation was used.

4.3 Mahalanobis Distance

The Mahalanobis Distance is often used to measure distances between points in multidimensional space. It assumes that each dimension has a different variance and it accounts for the covariances between two data points. Equation (1) describes this distance and is used to calculate the starting similarity matrix for every clustering method. The implementation was straight forward, although access to the whole dataset is required because the covariance matrix has to be calculated. It is important to mention that his calculation is computationally expensive for large datasets. The time for calculating the distance matrix with the Mahalanobis function is about two orders of magnitude longer than the other methods (Table 7).
It can happen that the determinant of the covariance matrix becomes zero and so the inverse of the matrix cannot be calculated. This case must be addressed in the program code to properly inform the user. A *java.lang.ArithmeticException: / by zero* is thrown and to handle this case a `try` block which catches the divide by zero exception is used and a warning message explaining the error is displayed and the calculation is canceled. This is faster than calculating the determinant and checking whether it is zero because it does not have to be calculated and checked, it just calculates and if there is a zero division, it stops. When comparing the clustering results of Genesis and R slight differences were encountered in certain tests. Further investigations revealed, that in the case of two identical distances, it is not defined which one should be chosen for merging. In fact Genesis is using the *last* smallest distance by default to create a new cluster and R is using the $first$ smallest distance. It is not possible to say which way is the correct one. As both approaches are valid, Genesis was modified to use the $first$ smallest distance from the similarity matrix to get comparable results with R.
The result created by Genesis (figure 8a) is compared with the clustering software $PAST$ and the heatmap created in R (figure 8b). The results for these algorithms are identical except for the order of the rows. A first look might be irritating because the order of the

elements in this dendrograms is different. Comparing the structure of the tree without taking the edge lengths into account using the PH85 method yields 0 in all cases. This means the trees are complete equal (Appendix Figure 10).

4.4 Ward's Linkage

HCL uses different methods to determine which clusters should be merged. Therefore all combinations of linkages are calculated and the best one is chosen to create a new cluster. The Ward's linkage method is also called the minimum variance criterion because it tries to minimize the total cluster variance. The specific equation (2) shows the Ward Linkage calculation which can also be calculated by the recursive defined function (3). This recursive definition allows simple implementation of different linkage methods. The additional algorithms were Average Group- and Flexible- linkage. Most of these algorithms were already implemented in Genesis but due to structural reasons, all of these calculations were changed to just use the recursive equation. Table 2 displays the factors needed for different calculations and equation (3) shows the overall formula for the distance calculations. With an `if` request the method is chosen and the values are calculated as mentioned in table 2. All of these methods were crosschecked for correct results with $PAST$ or R to ensure no errors were made during the implementation. No significant change in calculation time was observed after changing the implementation to the general formula.

4.5 Integration of PAM

Partitioning Around Medoids Clustering (PAM) [14] is highly similar to k-means but is robust against outliers and also faster. To integrate a whole new clustering algorithm, a new package was created in the current Genesis source code structure. The structure within the package is similar to the already existing k-means clustering algorithm but could not be derived due to major changes in the similarity matrix calculation.

4.6 Integration of TC

As Transitivity Clustering (TC) yielded good results in clustering gene expression data [14], it was added to the Genesis clustering functions. TC yields clusters as the result and is similar to hierarchical clustering [1] although it works differently. The structure of the TC package is similar to the HCL package but the main parts in this classes were changed.

4.7 Partitioning Around Medoids

This method is highly similar to k-means clustering. The advantage of PAM compared to k-means is that the selection of an existing point as center instead of the true mean leads to a robustness against outliers and is computationally slightly cheaper [1]. Figure 5 shows an example of the clustering process on a 2D example dataset. Asyali et al. [54] describes PAM as an easy to implement and highly transparent clustering method which has a low computational complexity of $O(tkN)$ just like k-means. N is the number of elements to cluster, t is the maximum number of iterations and k is the number of clusters. Compared to k-means, PAM is robust to outliers and it is important to mention that both methods, PAM and k-means, can't form arbitrary clusters because the element affiliation to a cluster center is given by its distance and not by any density dependencies [1].

4.8 Transitive Clustering

This method is robust against noise and it needs only a threshold as the only input parameter which determines the number of clusters created. This means that it is impossible to tell in advance how many clusters are going to be created. This algorithm was implemented according to the pseudo code first mentioned by Ben-Dor et al. [18] (Algorithm 3). Because the Cluster Affinity Search Technique (CAST) is very efficient computationally and so it was chosen for implementation. The computational complexity is $O(N^2 \cdot log(N))$ which is much larger than for the PAM method [18] because it has an quadratic increase over data points N. It is important to mention that despite several e-mail attempts it was not possible to get exact information about the implementation method of the TC clustering method in ClustEval [14] and therefore CAST was chosen.

4.9 Datasets and Validation

Six Datasets were used for validation (Table 5). The first three datasets, *Fibroblast_red*, *Fibroblast_red_ns* and *Fibroblast_ns* were taken from Genesis. This test sets are used for validating the new distance measures. *Dataset 1* was downloaded from the NCBI GEO [44] database and is a randomly chosen gene expression dataset which is clustered and evaluated. *Dataset 2* comes with the ClustEval [14] platform and is a bone marrow gene expression analysis for leukemia [46]. This dataset has a so called Gold Standard which is used for classification validation because the correct clustering of the samples is known and so TP, TF, FN and FP can be calculated. *Dataset 3* is a synthetic one and has a given true classification for comparison [47].
Validation of clustering results in general is not easy because no reference results are available. *Dataset* 2 and 3 have a Gold Standard which represents the correct classification,

which allows to compare clustering qualities. The F_1-*score* which is a common accuracy measure and the MCC which takes also the TN into account can be calculated. If the cluster sizes have a strong variation in size (data points) the MCC gives better information about the quality. If no Gold Standard is available, inner cluster measurement values can be used to calculate cluster quality. The most common validation measure is the Silhouette value [36]. It uses the distances between points in a cluster and compares this distances to the distances of the same points to all points of an other cluster. This leads to a good measure of cluster "tightness" and "separation". A large Silhouette value means that there is a high similarity between the elements of one cluster. A small value means there is no similarity according to the element distances.

To crosscheck the results, *Dataset 1* was adapted to be used with Genesis. The data is clustered in two clusters and the Silhouette values are calculated. Then three clusters are created and the values are again calculated. A slight decrease in the value from two to three clusters was recognized and so no further clustering was done. For this dataset the Silhouette values are larger than for the *Dataset 2* clusters. This has to do with the different dataset and the distribution and cluster "separation" which is different for each dataset. For the validation, different parameters were used and the Silhouette values were calculated. Tables 20-25 in the Appendix show Silhouette values from 0.3425 to 0.5828 for each cluster against each cluster. The main diagonals in this tables should always be smaller than the other values to guarantee a high Silhouette value which in fact is the case for all tables. The achieved Silhouette values are rather low, suggesting clusters are not well separated.

Comparing the Silhouette values for *Dataset 2* calculated on the Genesis results 0.3549 for TC and 0.3425 for PAM with the results from ClustEval [14] which are 0.360 for TC and 0.352 for PAM it is easy to see that the values are quiet low, but at least very similar. If clusters are "tight" and close to each other, low Silhouette values can be observed, although the clusters give good F_1-score. This is the case with *Dataset 2* which has a very good F_1-score of 0.8824 for TC and 0.9118 for PAM but a Silhouette value of 0.3549 for TC and 0.3425 for PAM. This compares well with the results of ClustEval [14]. There is a slight difference in this values which result from the fact that it was not possible to reconstruct the exact implementation in ClustEval [14]. *Dataset 3* yielded an F_1-score of 0.871 for k-means a bit better than PAM with an F_1-*score* of 0.766. This results are also comparable with the MCC which is 0.816 for k-means and 0.746 for PAM. However TC reached an F_1-*score* of 1 which means that all samples were correctly classified.

4.10 Conclusion

One distance measurement, three new linkage methods and two new clustering methods were added to the Genesis project. The distance and similarity measures were cross-checked with R and PAST to validate their correct implementation. The validation of PAM and TC with ClustEval gave good F_1-score and similar Silhouette values. In contrast to the existing clustering algorithms, where distances are calculated on the fly, for the new methods, the distance matrix was precalculated, because nowadays memory is assumed to be "cheaper" than computational time. The test sets were clustered with different parameters and a good Silhouette values could be achieved. Overall it is hard to say if an result is good or bad because it always depends on the type of data as mentioned by Jaskowiak et al. [37]. The proper implementation of the algorithms was validated and the results were comparable to the results from ClustEval and the F_1-score and MCC for the datasets were quite good for PAM and very good for TC. The implementation provides new methods which can be useful in analyzing and validating biomedical data.

5 Literature

References

[1] Andreopoulos B, An A *et al.*: **A roadmap of clustering algorithms: finding a match for a biomedical application**. *Briefings in Bioinformatics* 2009. **10**(3): 297–314.

[2] Fodeh SJ, Brandt C *et al.*: **Complementary ensemble clustering of biomedical data.** *Journal of Biomedical Informatics* 2013. **46**(3): 436–443.

[3] PubMed [Internet]. Bethesda (MD): National Library of Medicine (US). https://www.ncbi.nlm.nih.gov/pubmed/, Date accessed: 2017-12-05.

[4] Hammer Ø, Harper DA *et al.*: **PAST: Paleontological Statistics Software Package for Education and Data Analysis**. *Palaeontologia Electronica* 2001. **4**(1): 1–9.

[5] Eisen MB, Spellman PT *et al.*: **Cluster analysis and display of genome-wide expression patterns.** *Proceedings of the National Academy of Sciences of the United States of America* 1998. **95**(25): 14863–14868.

[6] Howe EA, Sinha R *et al.*: **RNA-Seq analysis in MeV**. *Bioinformatics* 2011. **27**(22): 3209–3210.

[7] Saeed AI, Sharov V *et al.*: **TM4: A free, open-source system for microarray data management and analysis**. *BioTechniques* 2003. **34**(2): 374–378.

[8] Rossum GV: Python. https://www.python.org/, Date accessed: 2016-11-28.

[9] Hoon MD: Pycluster: Python Package Index. https://pypi.python.org/pypi/Pycluster, Date accessed: 2016-11-28.

[10] MathWorks - MATLAB. http://de.mathworks.com/, Date accessed: 2016-11-28.

[11] Abonyi J: *Clustering Toolbox: For Use with MATLAB*. MathWorks, ????

[12] R Development Core Team: *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2008. http://www.r-project.org.

[13] Maechler M and Rousseeuw P: CRAN - Package cluster. https://cran.r-project.org/web/packages/cluster/index.html, Date accessed: 2016-12-05.

[14] Wiwie C, Baumbach J *et al.*: **Comparing the performance of biomedical clustering methods.** *Nature Methods* 2015. **12**(11): 1033–1038.

[15] Sturn A: **Cluster Analysis for Large Scale Gene Expression Studies**. *Master Thesis Institute for Biomedical Engineering, Graz University of Technology and The Institute for Genomic Research* 2000.

[16] Sturn A, Quackenbush J *et al.*: **Genesis: cluster analysis of microarray data.** *Bioinformatics* 2002. **18**(1): 207–208.

[17] Kumar P and Wasan SK: **Comparative Study of K-Means, Pam and Rough K-Means Algorithms Using Cancer Datasets**. In *In Proceedings of CSIT: 2009 International Symposium on Computing, Communication, and Control (ISCCC 2009)* . 2011: 136–140.

[18] Ben-Dor A, Shamir R *et al.*: **Clustering gene expression patterns.** *Journal of Computational Biology* 1999. **6**(3-4): 281–97.

[19] Taub FE, DeLeo JM *et al.*: **Sequential Comparative Hybridizations Analyzed by Computerized Image Processing Can Identify and Quantitate Regulated RNAs**. *DNA* 1983. **2**(4): 309–327.

[20] Lister R, O'Malley RC *et al.*: **Highly Integrated Single-Base Resolution Maps of the Epigenome in Arabidopsis**. *Cell* 2008. **133**(3): 523–536.

[21] Nazar RN, Chen P *et al.*: **DNA chip analysis in diverse organisms with unsequenced genomes**. *Molecular Biotechnology* 2010. **44**(1): 8–13.

[22] Pollack JR, Perou CM *et al.*: **Genome-wide analysis of DNA copy-number changes using cDNA microarrays**. *Nature Genetics* 1999. **23**(1): 41–46.

[23] Chu Y and Corey DR: **RNA sequencing: platform selection, experimental design, and data interpretation.** *Nucleic Acid Therapeutics* 2012. **22**(4): 271–4.

[24] Griffith M, Walker JR *et al.*: **Informatics for RNA Sequencing: A Web Resource for Analysis on the Cloud**. *PLoS Computational Biology* 2015. **11**(8): 1–20.

[25] Tarazona S and García-Alcalde F: **Differential expression in RNA-seq: A matter of depth**. *Genome Research* 2011. **21**(12): 2213–2223.

[26] Trapnell C, Hendrickson DG *et al.*: **Differential analysis of gene regulation at transcript resolution with RNA-seq.** *Nature Biotechnology* 2013. **31**(1): 46–53.

[27] Griffith M, Griffith OLO *et al.*: **Alternative expression analysis by RNA sequencing**. *Nature Methods* 2010. **7**(10): 843–7.

[28] Rozowsky J, Abyzov A *et al.*: **AlleleSeq: analysis of allele-specific expression and binding in a network framework**. *Molecular Systems Biology* 2011. **7**(522): 1–15.

[29] The Eclipse Foundation: Eclipse IDE for Java Developers. https://www.eclipse.org/downloads/packages/eclipse-ide-java-developers/neon1a, Date accessed: 2016-06-14.

[30] CCCB: MeV. http://www.tm4.org/, Date accessed: 2016-11-28.

[31] Xu R and Wunsch DC: **Clustering Algorithms in Biomedical Research: A Review**. *IEEE Reviews in Biomedical Engineering* 2010. **3**: 120–154.

[32] Cormack R: **A Review of Classification**. *Journal of the Royal Statistical Society. Series A (General)* 1971. **134**(3): 321–367.

[33] Powers DMW: **Evaluation: From Precision, Recall and F-Measure To ROC, Informedness, Markedness & Correlation**. *Journal of Machine Learning Technologies* 2011. **2**(1): 37–63.

[34] Wittkop T, Emig D *et al.*: **Partitioning biological data with transitivity clustering.** *Nature Methods* 2010. **7**(6): 419–420.

[35] Boughorbel S, Jarray F *et al.*: **Optimal classifier for imbalanced data using Matthews Correlation Coefficient metric**. *PLoS ONE* 2017. **12**(6): e0177678.

[36] Rousseeuw PJ: **Silhouettes: A graphical aid to the interpretation and validation of cluster analysis**. *Journal of Computational and Applied Mathematics* 1987. **20**(1): 53–65.

[37] Jaskowiak PA, Campello RJ *et al.*: **On the selection of appropriate distances for gene expression data clustering**. *BMC Bioinformatics* 2014. **15**(2): S2.

[38] Watson J: **VirtualBox: Bits and Bytes Masquerading As Machines**. *Linux J.* 2008. **2008**(166).

[39] Gregory R. Warnes BB: CRAN - Package gplots. https://cran.r-project.org/web/packages/gplots/index.html, Date accessed: 2016-12-06.

[40] R Core Team and contributors worldwide: R: The R Stats Package. https://stat.ethz.ch/R-manual/R-devel/library/stats/html/stats-package.html, Date accessed: 2016-12-05.

[41] R Core Team and contributors worldwide: R: The R Base Package. https://stat.ethz.ch/R-manual/R-devel/library/base/html/00Index.html, Date accessed: 2017-01-18.

[42] Paradis E, Blomberg S *et al.*: CRAN - Package ape. https://cran.r-project.org/web/packages/ape/index.html, Date accessed: 2018-04-10.

[43] Ge M, Sattler K *et al.*: **The Transcriptional Program in the Response of Human Fibroblasts to Serum**. *Nature* 1999. **283**(1): 83–87.

[44] Edgar R: **Gene Expression Omnibus: NCBI gene expression and hybridization array data repository**. *Nucleic Acids Research* 2002. **30**(1): 207–210.

[45] NCBI GEO. ftp://ftp.ncbi.nlm.nih.gov/geo/series/GSE70nnn/GSE70145/, Date accessed: 2017-09-03.

[46] Tamayo P and Slonim D: **Molecular classification of cancer: class discovery and class prediction by gene expression with self-organizing maps: methods and applications to hematopoietic differentiation**. *Proceedings of the National Academy of Sciences of the United States of America* 1999. **96**(October): 2907–2912.

[47] Fränti P, Virmajoki O *et al.*: **Fast agglomerative clustering using a k-nearest neighbor graph**. *EEE Transactions on Pattern Analysis and Machine Intelligence 2006* 2006. **28**(11): 1875–1881.

[48] Monti S, Tamayo P *et al.*: **Consensus clustering: A resampling-based method for class discovery and visualization of gene expression microarray data**. *Machine Learning* 2003. **52**(1-2): 91–118.

[49] D'haeseleer P: **How does gene expression clustering work?** *Nature Biotechnology* 2005. **23**(12): 1499–1501.

[50] Van der Laan MJ, Pollard KS *et al.*: **A new partitioning around medoids algorithm**. *Journal of Statistical Computation and Simulation* 2003. **73**(8): 575–584.

[51] Robinson DF and Foulds LR: **Comparison of phylogenetic trees.** *Mathematical Biosciences* 1981. **53**(1): 31–147.

[52] Lance GN and Williams WT: **A General Theory of Classificatory Sorting Strategies: 1. Hierarchical Systems**. *The Computer Journal* 1967. **9**(4): 373–380.

[53] Datta S and Datta S: **Methods for evaluating clustering algorithms for gene expression data using a reference set of functional classes.** *BMC Bioinformatics* 2006. **7**(1): 397.

[54] Asyali MH, Colak D *et al.*: **Gene Expression Profile Classification: A Review**. *Current Bioinformatics* 2006. **1**(1): 55–73.

6 Appendix

6.1 Datasets for Validation

Table 18: Example data set from Genesis [15] and described by Eisen M [5]. This format contains a unique ID, a short Name, a gene weight n gene order and an experiment weight. The remaining entries show the gene expression at different time points.

UNIQID	NAME	GWEIGHT	GORDER	0HR	15MIN	30MIN	1HR	2HR	4HR	6HR	8HR	12HR	16HR	20HR
EWEIGHT				1	1	1	1	1	1	1	1	1	1	1
361771	EST W95909	1	517	0	-0.47	-3.32	-0.81	0.11	-0.60	-1.36	-1.03	-1.84	-1.00	-0.60
487537	SID487537 H.sapiens mRNA for selenoprotein P	1	516	0	0.66	0.07	0.20	0.29	-0.89	-0.45	-0.29	-0.29	-0.15	-0.45
486735	SID486735 Human peptidyl-prolyl isomerase and essential mitotic regulator (PIN1) mRNA, complete cds	1	515	0	0.14	-0.04	0.00	-0.15	-0.58	-0.30	-0.18	-0.38	-0.49	-0.81
417426	Homo sapiens protein 4.1-G mRNA, complete cds	1	514	0	-0.04	0.00	-0.23	-0.25	-0.47	-0.60	-0.56	-1.09	-0.71	-0.76
469959	SID469959 EST AA029909	1	513	0	0.28	0.37	0.11	-0.17	-0.18	-0.60	-0.23	-0.58	-0.79	-0.29
381721	SID381721 EST AA059077	1	512	0	0.54	0.53	0.16	0.14	0.20	-0.34	-0.38	-0.36	-0.49	-0.58
471855	SID471855 Lumican	1	511	0	0.20	0.14	0.00	0.11	-0.34	-0.03	0.04	-0.76	-0.81	-1.12
486757	EST AA180272	1	510	0	0.40	0.43	0.18	0.00	-0.14	0.29	0.07	-0.79	-0.81	-0.92
417593	Carnitine palmitoyltransferase I (CPTI)	1	509	0	0.01	0.46	0.28	-0.34	-0.23	-0.36	-0.45	-0.64	-0.79	-1.22
50922	EST H19324	1	508	0	-0.23	0.04	0.00	-0.30	-0.29	-0.45	-0.97	-2.06	-0.89	-1.22

Table 19: Example data from NCBI GEO [44] (*Dataset 2*) has a unique ID in the first column. The weights are one but there is no Order and also no names. The rest of the entries shows log2 ratio of probe set signal intensity in the sample to the mean in the reference set amount at different times.

ID_REF	NAME	GWEIGHT	CP-2	CP-4	CP-5	CP-7	CP-8	CP-9	CP-10	CP-12	CP-13	CP-18	CP-20
EWEIGHT			1	1	1	1	1	1	1	1	1	1	1
A_16_P17103965		1	0.06060	-0.087	-0.12900	-0.01120	-0.16200	-0.02500	-0.06000	-0.09260	-0.10800	-0.02510	-0.05870
A_16_P01721093		1	0.06760	0.129	0.00465	-0.00117	0.03170	-0.00178	0.03260	0.04410	0.13700	0.02480	0.01450
A_16_P15059815		1	0.04590	-0.00557	0.02620	0.03640	0.00118	0.04840	-0.00221	0.07730	0.09960	0.03160	0.02930
A_16_P03096255		1	0.02700	0.0996	0.11800	0.05430	0.12400	0.05350	0.10700	0.10100	0.16700	0.00684	0.08650
A_16_P21424156		1	0.01140	0.0108	0.16700	0.06460	0.10100	0.04790	-0.04080	0.04010	0.07560	0.06310	0.05570
A_16_P59922478		1	-0.00674	0.0233	0.06630	0.02500	0.07190	0.01860	0.01110	0.02260	0.25900	0.01910	0.00090
A_18_P15756033		1	0.00582	-0.125	-0.06040	-0.02360	-0.03630	-0.05650	-0.09070	-0.00490	0.04580	0.07830	-0.06500
A_16_P18783515		1	-0.00656	-0.0194	0.04470	-0.02150	0.12700	-0.01060	0.06420	0.03420	-0.11900	-0.03990	-0.00135
A_16_P00403029		1	0.04670	0.0848	0.01520	0.04430	0.06130	-0.01450	0.03990	0.06050	0.14200	-0.03820	0.10400
A_16_P17423087		1	-0.02260	-0.00998	-0.11100	-0.08010	-0.05600	-0.08480	-0.08820	-0.17600	-0.03320	-0.02720	-0.08800
A_16_P21621957		1	0.01520	0.0942	0.02400	0.01710	0.10600	0.06880	-0.01030	0.08930	0.06270	-0.03170	0.02570
A_16_P38381210		1	0.00112	0.0656	-0.03570	-0.06430	-0.02200	-0.07720	-0.03950	-0.01720	0.23800	-0.06100	-0.05400
A_16_P18159690		1	0.00823	-0.0277	-0.07870	-0.11900	-0.06760	-0.08370	-0.42600	-0.09970	-0.30000	-0.03790	-0.08950
A_16_P03385907		1	-0.16000	-0.0683	0.09710	0.10500	0.08710	0.09280	0.11300	0.12500	0.15200	0.04700	0.07160
A_16_P02101244		1	-0.03460	-0.116	-0.08820	-0.03970	-0.07850	-0.05770	-0.10200	-0.11100	-0.04080	-0.01320	-0.03370
A_16_P19296953		1	0.02020	0.0641	0.13300	0.08190	0.16300	0.07360	0.07390	0.09560	0.15000	-0.00698	0.10100
A_16_P18653359		1	-0.01870	-0.0344	0.11200	0.06050	0.10600	0.05160	0.06660	0.13700	0.02310	0.02130	0.06590
A_16_P20973757		1	0.04380	-0.0653	-0.04300	-0.01140	-0.02410	-0.06120	0.05610	-0.05930	-0.02100	-0.01300	-0.04690
A_16_P20293969		1	-0.02580	-0.0546	0.04070	0.04330	0.11600	0.00393	0.07290	0.12100	0.03930	-0.00632	0.04910
A_16_P31322059		1	0.05060	0.0974	-0.07580	-0.02530	-0.01190	0.01210	-0.07240	-0.04740	0.01960	-0.02300	-0.06110
A_16_P03462470		1	-0.05310	0.0641	0.15200	0.13700	0.16800	0.06390	0.04750	0.31700	0.18400	0.03720	0.13300
A_16_P21098513		1	0.02070	0.057	0.02530	0.01750	0.01460	0.01810	-0.05890	-0.06350	-0.23900	0.01100	0.04710
A_16_P59800141		1	0.00449	0.109	0.04730	-0.00924	0.10900	0.07730	0.02330	0.06460	0.36500	-0.01810	0.04350
A_16_P40212811		1	-0.18400	-0.0899	-0.07160	-0.03010	-0.08290	-0.05210	-0.12600	-0.08950	-0.05780	-0.03930	-0.07850
A_18_P11116317		1	0.05670	0.134	-0.05200	-0.00164	-0.00340	0.01730	-0.09520	-0.06780	-0.02350	-0.05180	0.05940
A_16_P00242008		1	0.03630	-0.0642	-0.08200	0.01180	-0.35600	-0.02170	-0.06800	-0.10000	-0.08560	-0.02110	0.07110
A_16_P58316227		1	-0.00194	-0.0911	-0.13400	-0.04370	0.00869	-0.06690	-0.06350	-0.03070	-0.07790	-0.03900	-0.05970

6.2 Silhouette Value Calculations

Table 20: Silhouette value and in between cluster distances a_i, b_i and for all Clusters against all Clusters. *Dataset* 1 with Spearman's rank correlation and PAM clustering with $k = 2$.

	Cluster 1	**Cluster 2**
Cluster 1	0.3909	0.7687
Cluster 2	0.8049	0.2743
	S_2	S_1
	0.5144	0.6431
	Silhouette value	
	0.5788	

Table 21: Silhouette value and in between cluster distances a_i, b_i and for all Clusters against all Clusters. *Dataset* 1 with Spearman's rank correlation and TC clustering with $T = 0.4$.

	Cluster 1	**Cluster 2**
Cluster 1	0.2798	0.8430
Cluster 2	0.7263	0.3786
	S_2	S_1
	0.6147	0.5509
	Silhouette value	
	0.5828	

Table 22: Silhouette value and in between cluster distances a_i, b_i and c_i for all Clusters against all Clusters. *Dataset* 1 with Spearman's rank correlation and PAM clustering with $k = 3$.

	Cluster 1	**Cluster 2**	**Cluster 3**
Cluster 1	0.3461	0.7945	0.5350
Cluster 2	0.7726	0.2567	0.5980
Cluster 3	0.7455	0.4336	0.3718
	S_2 and S_3	S_1 and S_3	S_1 and S_2
	0.5521	0.6769	0.3051
	0.5358	0.4080	0.3783
	Silhouette value		
	0.4760		

Table 23: Silhouette value and in between cluster distances a_i, b_i and c_i for all Clusters against all Clusters. *Dataset* 1 with Spearman's rank correlation and TC clustering with $T = 0.5$.

	Cluster 1	**Cluster 2**	**Cluster 3**
Cluster 1	0.3466	0.7958	0.6122
Cluster 2	0.8124	0.2741	0.6160
Cluster 3	0.5893	0.6079	0.4535
	S_2 and S_3	S_1 and S_3	S_1 and S_2
	0.5740	0.6556	0.2592
	0.4128	0.5492	0.2637
	Silhouette value		
	0.4524		

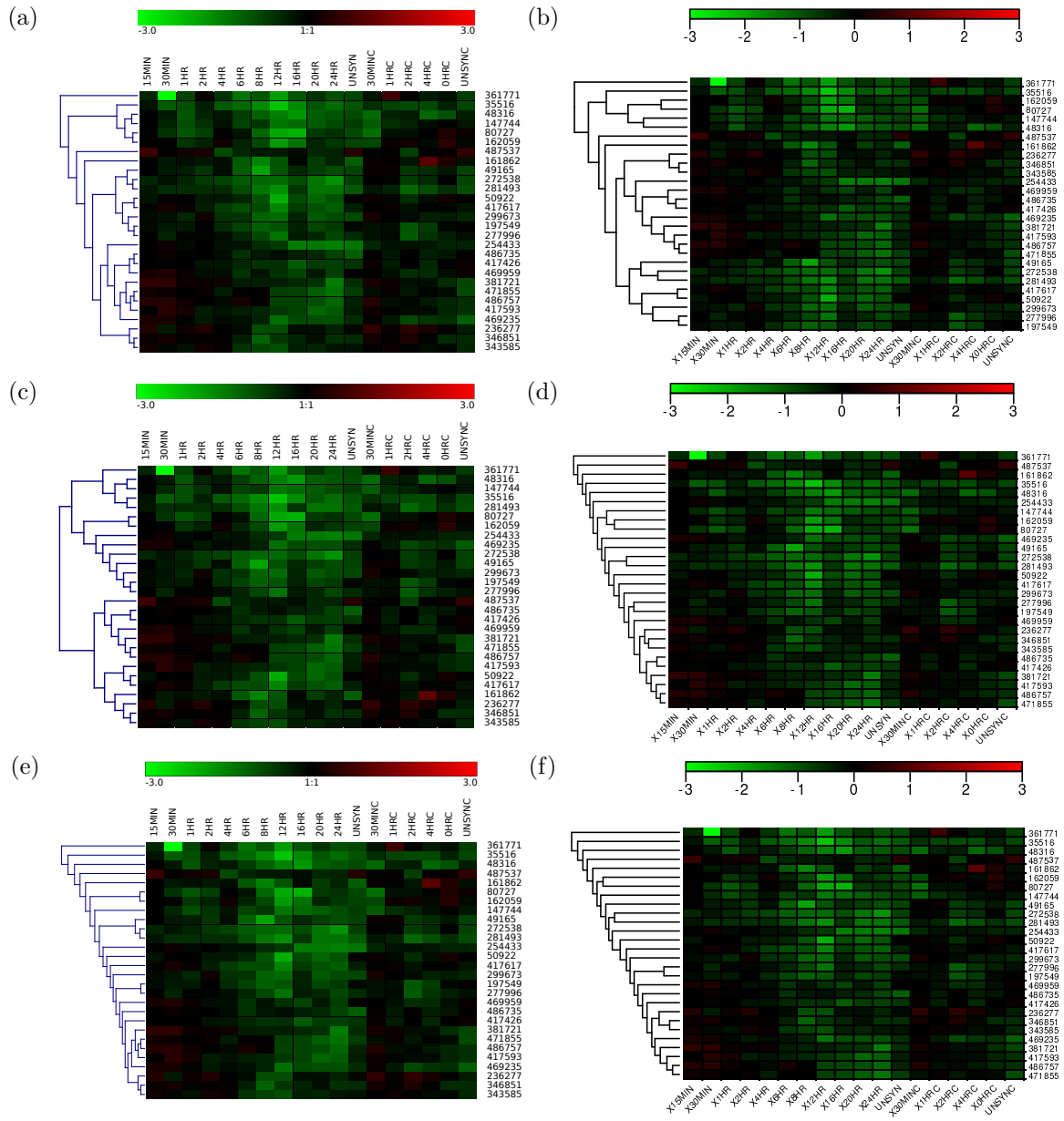
Table 24: Silhouette value and in between cluster distances a_i, b_i and c_i for all Clusters against all Clusters. *Dataset* 2 with Spearman's rank correlation and PAM clustering with $k = 3$.

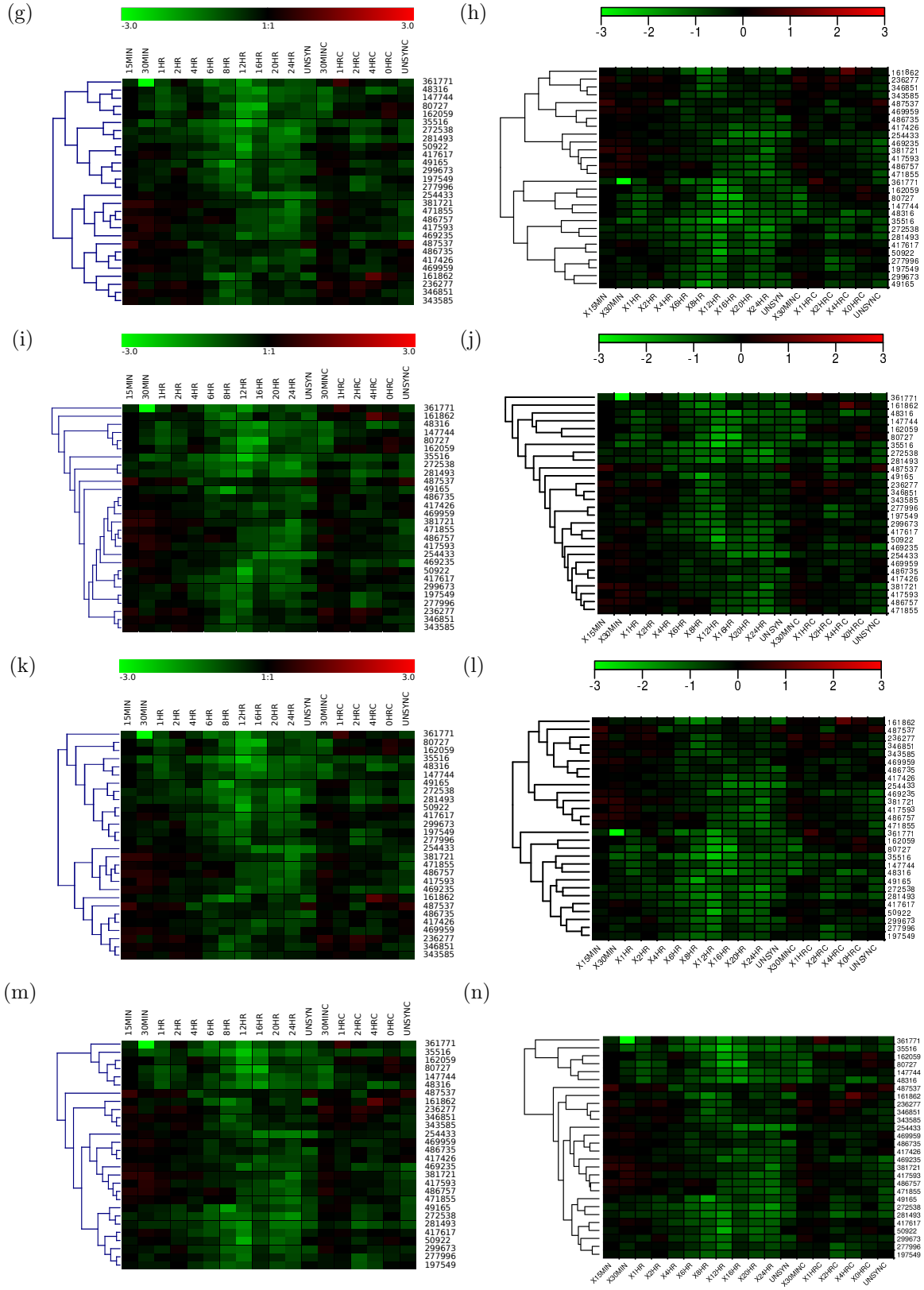
	Cluster 1	**Cluster 2**	**Cluster 3**
Cluster 1	0.3870	0.6575	0.5409
Cluster 2	0.6067	0.3436	0.4591
Cluster 3	0.5783	0.5669	0.3748
	S_2 and S_3	S_1 and S_3	S_1 and S_2
	0.3622	0.4775	0.1835
	0.3309	0.3940	0.3071
	Silhouette value		
	0.3425		

Table 25: Silhouette value and in between cluster distances a_i, b_i and c_i for all Clusters against all Clusters. *Dataset* 2 with Spearman's rank correlation and TC clustering with $T = 0.5050505$.

	Cluster 1	**Cluster 2**	**Cluster 3**
Cluster 1	0.3755	0.6455	0.4712
Cluster 2	0.6067	0.3436	0.5409
Cluster 3	0.5745	0.5729	0.3692
	S_2 and S_3	S_1 and S_3	S_1 and S_2
	0.3812	0.4677	0.2163
	0.3465	0.4003	0.3174
	Silhouette value		
	0.3549		

6.3 Linkage Results Genesis and R




(g)
-3.0
1:1
3.0
15MIN
30MIN
1HR
2HR
4HR
6HR
8HR
12HR
16HR
20HR
24HR
UNSYN
30MINC
1HRC
2HRC
4HRC
0HRC
UNSYNC
(h)
-3
-2
-1
0
1
2
3
(i)
(j)
(k)
(l)
(m)
(n)

(o) (p)

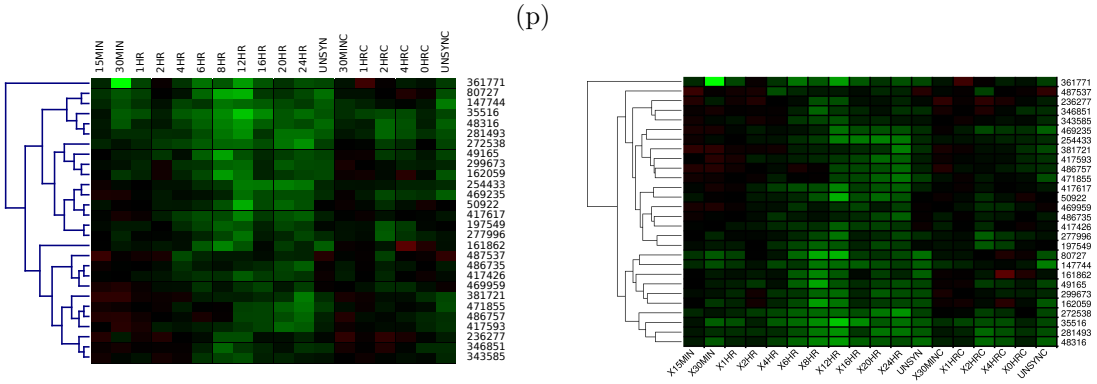


Figure 10: The figures on the left side are showing the heat map result generated by Genesis and the right side shows a heat map for the same clustering algorithm generated by *R*. The heat maps in *R* are generated with the *heatmap*.2 function from the *gplots* package. All starting distances are euclidean distances and the used clustering algorithms for the left and the right image are always the same.
(a) Weighted Average WPGMA Genesis, (b)Weighted Average WPGMA *R*, (c) Median Linkage Genesis, (d) Median Linkage *R*, (e) Centroid Linkage Genesis, (f) Centroid Linkage *R*, (g) Ward's Linkage Genesis, (h) Ward's Linkage *R*, (i) Single Linkage Genesis, (j) Single Linkage *R*, (k) Complete Linkage Genesis, (l) Complete Linkage *R*, (m) Flexible Linkage $\beta = 0$ Genesis, (n) Flexible Linkage $\beta = 0$ *R*, (o) Group Average Linkage Genesis, (p) Group Average Linkage *R*

6.4 Script

```
#load Packages
  install.packages("gplots")
  library(gplots)
  install.packages("cluster")
  library(cluster)

# Mahalanobis Function
################################################################################

cholMaha <- function(X) {
  dec <- chol( cov(X) )
  tmp <- forwardsolve(t(dec), t(X) )
  dist(t(tmp))
}
distMahal = cholMaha(Fibroblasts);

# Data Processing
################################################################################

#set directory or put file in same folder
#setwd(dir = "Documents/Master/Master Projekt/files/")

```

```
#read Data
Fibroblasts <- read.table("Fibroblasts.txt",
                          sep="\t",
                          comment.char="#",
                          header = 1)
#save original data
data <- Fibroblasts

# assign labels in column 1 to "rnames"
rnames <- data[,1]

# transform column 2-5 into a matrix
matData <- data.matrix(data[,2:ncol(data)])

# assign row names
rownames(matData) <- rnames

# Color Definition
################################################################################
# create color pattern
myPalette <- colorRampPalette(c("green", "black", "red"))(n = 299)

# define color breaks
colBreaks = c(seq(-3,-1,length=100),
              seq(-0.99,0.99,length=100),
              seq(1,3,length=100))


# Createing the Heat Map
################################################################################

#set b ]-inf,1[ for flexible linkage and create heat map
b = 0;
heatmap.2(matData,
          #use distMahal if mahalanobis needed! use median or centroid
          #hclustfun = function(x) hclust(distMahal,method = 'single'),

          #use if average, single, complete, ward, weighted, or flexible
          hclustfun = function(x) agnes(matData,
                                        diss = FALSE,
                                        metric="euclidean",
                                        method="flexible",
                                        par.method =
                                         c((1/2)*(1-b),(1/2)*(1-b),b,0)),
          cellnote = matData,
          main = "",
```

```
        notecol="transparent",
        density.info="none",
        trace="none",
        margins =c(10,4),
        col=myPalette,
        breaks=colBreaks,
        dendrogram="row",
        Colv="NA",
        sepwidth=c(0.01,0.01),
        sepcolor="black",
        colsep=1:ncol(matData),
        rowsep=1:nrow(matData),
        cexRow=0.5,
        cexCol=0.5,
        srtCol=45,
        keysize=1,
        lmat = rbind(3:4,2:1),
        lwid = c(1,1),
        lhei = c(1,1),
        key.par=list(mar=c(4,0,19,4.7)))
```

Listing 1: R-Scrip used to generate the reference heatmaps