Czech Technical University in Prague
Faculty of Electrical Engineering

Department of Cybernetics

BACHELOR PROJECT ASSIGNMENT

Student:	Tomáš S i x t a
Study programme:	Electrical Engineering and Information Technology
Specialisation:	Cybernetics and Measurement
Title of Bachelor Project:	Knowledge-Oriented Gene Expression Data Processing

Guidelines:

1. Get familiar with the gene expression data MOTOL.
2. Learn to work in R environment, focus on the set of Bioconductor packages.
3. Study and review the available resources of background knowledge for microarray data.
4. Use the resources from the previous step to find significant gene sets.
5. Use the same resources to cluster genes and construct metagenes as cluster represe.
6. Classify MOTOL data with the aid genes and metagenes.
7. Give a summary of the reached results.

Bibliography/Sources:

[1] Baxevanis, Ouellette: Bioinformatics: A Practical Guide to the Analysis of Genes and Proteins. 3rd Edition, Wiley, 2005.
[2] Bioconductor Online Documentation available at: http://www.bioconductor.org/docs/pslides

Bachelor Project Supervisor: Ing. Jiří Kléma, Ph.D.

Valid until: the end of the winter semester of academic year 2009/2010

České vysoké učení technické v Praze
Fakulta elektrotechnická
L.S.

[illegible]
prof. Ing. Vladimír Mařík, DrSc.
Head of Department

[illegible]
doc. Ing. Boris Šimák, CSc.
Head

Prague, Februar 23, 2009

České vysoké učení technické v Praze
Fakulta elektrotechnická

Katedra kybernetiky

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Student:	Tomáš Sixta
Studijní program:	Elektrotechnika a informatika (bakalářský), strukturovaný
Obor:	Kybernetika a měření
Název tématu:	Znalostní zpracování dat genové exprese

Pokyny pro vypracování:

1. Seznamte se s daty genové exprese MOTOL.
2. Seznamte se s Bioconductorem a R package.
3. Prostudujte zdroje anotačních informací pro microarray data, vytvořte jejich rešerši.
4. Použijte anotace k vyhledání signifikantních tříd genů.
5. Využijte anotační data pro shlukování a tvorbu metagenů.
6. Klasifikujte MOTOL data s apriorní znalostí (geny, metageny).
7. Zhodnoťte dosažené výsledky.

Seznam odborné literatury:

[1] Baxevanis, Ouellette: Bioinformatics: A Practical Guide to the Analysis of Genes and Proteins. 3rd Edition, Wiley, 2005.
[2] Bioconductor Online Documentation available at: http://www.bioconductor.org/docs/pslides

Vedoucí bakalářské práce: Ing. Jiří Kléma, Ph.D.

Platnost zadání: do konce zimního semestru 2009/2010

L.S.
ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE
Fakulta elektrotechnická

Mařík

prof. Ing. Vladimír Mařík, DrSc.
vedoucí katedry

[illegible]

doc. Ing. Boris Šimák, CSc.
děkan

V Praze dne 23. 2. 2009

Czech Technical University in Prague

Faculty of Electrical Engineering

Bachelor project

Knowledge-Oriented Gene Expression Data Processing

Tomáš Sixta

Supervisor: Ing. Jiří Kléma, Ph.D.

Study programme: Electrical Engineering and Information Technology

Specialisation: Cybernetics and Measurement

June, 2009

Poděkování

Rád bych na tomto místě poděkoval vedoucímu práce Ing. Jiřímu Klémovi, Ph.D. za odborné rady a časté a přínosné konzultace. Děkuji Barboře Nechanské za pomoc při řešení lékařských otázek souvisejících s touto prací. Rovněž děkuji rodičům Marii a Josefovi Sixtovým za podporu v mém dosavadním studiu.

Prohlášení

Prohlašuji, že jsem svou bakalářskou práci vypracoval samostatně a použil jsem pouze podklady (literaturu, projekty, SW atd.) uvedené v přiloženém seznamu.

V Praze dne 12.6.2009 .... [illegible]

Podpis

Abstract

Microarray technology is a very helpful tool for geneticists as it allows them to measure expression levels of thousands of genes simultaneously. However the resulted data are usually noised and their classification is hard, because they contain too many attributes (probes) and usually too low number of samples. Due to these reasons, it is convenient to reduce dimension of the data. It can be done by various ways, e.g. by clustering. This work describes fuzzy clustering algorithm currently implemented in web-accessible program DAVID of National Institute of Allergy and Infectious Diseases, which clusters gene lists by their associated annotation terms rather than the expression levels. Using this approach the clusters are then well biologically interpretable. Our target was to rewrite the algorithm to language R in order to surpass the restrictions of the web interface (a limited length of a gene list and non-adjustable annotation data). Using this code we have studied the influence of various annotation data on clustering results. We have clustered genes from Motol and ALL/AML data and created metagenes based on their gene expression matrices. Metagenes have been then analysed by two different classifiers.

Abstrakt

Microarray technologie představuje velmi užitečný nástroj pro genetiky, protože umožňuje měřit tisíce genových expresí najednou. Výsledná data jsou však zatížena šumem a jejich klasifikace je obtížná, protože obsahují příliš mnoho příznaků (sond) a většinou příliš malý počet vzorků. Z těchto důvodů je tedy vhodné snížit jejich dimenzi. Toho je možné dosáhnout různými způsoby, např. pomocí shlukování. Tato práce popisuje fuzzy shlukovací algoritmus, který je v současné době zabudován do programu DAVID Národního institutu pro alergie a infekční onemocnění, je přístupný přes webové rozhraní a geny neshlukuje podle jejich expresí, ale přidružených anotačních termů. Díky tomuto přístupu jsou pak skupiny genů dobře biologicky interpretovatelné. Naším cílem bylo přepsat tento algoritmus do jazyka R a odstranit omezení webového rozhraní (maximální velikost množiny genů a nemožnost měnit anotační data). Využívaje tohoto kódu, studovali jsme vliv různých anotačních dat na výsledné shluky. Provedli jsme shlukování genů přítomných v datech Motol a ALL/AML a z jejich matic genové exprese vytvořili metageny a ty posléze analyzovali dvěma různými klasifikátory.

Contents

Figures . . . xi

Tables . . . xii

1 Introduction . . . 1
1.1 Biological minimum . . . 1
1.2 Microarray technology . . . 2
1.3 Motol data . . . 3

2 Annotation sources . . . 4
2.1 KEGG pathways . . . 4
2.2 Gene Ontology . . . 4
2.3 Uniprot . . . 4
2.4 Another annotation sources . . . 5
2.5 The NCBI institute . . . 6

3 R & Bioconductor . . . 7
3.1 R . . . 7
3.1.1 Basic description . . . 7
3.1.2 Calling an external C code . . . 7
3.1.3 Working with databases . . . 8
3.2 Bioconductor . . . 10
3.2.1 AnnotationDbi . . . 10
3.2.2 Safe . . . 12
3.2.3 TopGO . . . 14
3.2.4 Macat . . . 17
3.3 Information sources . . . 20

4 Clustering . . . 21
4.1 DAVID . . . 21
4.2 Fuzzy clustering algorithm . . . 22
4.2.1 Kappa Statistics . . . 22
4.2.2 From original data to final clusters . . . 23
4.2.3 Algorithm's features . . . 26
4.2.4 Implementation . . . 27

5 Results . . . 28
5.1 On parameters identification . . . 28
5.1.1 Determining $C_{0,0}$ . . . 28
5.1.2 Collecting annotation data . . . 29

5.1.3 Parameters for internal seed relatedness 30
5.2 Clustering 31
5.2.1 Clustering with manually created distance matrix 33
5.2.2 Motol data 34
5.2.3 ALL/AML 36
5.3 Classification 38
5.3.1 Discussion 40

6 Conclusions 42

7 References 44

8 Apendices 48
8.1 Appended CD 48
8.2 Used software 49
8.3 Download of DAVID knowledgebase 49
8.4 Create arff file with metagenes from gene clusters 50

Figures

3.1 Macat results for chromosome 6 20
4.1 The genes before clustering 24
4.2 Qualified and unqualified initial seeds 25
4.3 Groups in the middle of iterative merging 25
4.4 Final clusters 26
5.1 Dependence of kappa on $C_{0,0}$ 28
5.2 Clustering workflow diagram 31
5.3 Classification workflow diagram 39
8.1 DAVID knowledgebase download page 49

Tables

3.1 topGO results 17
4.1 Sample gene-annotation matrix 22
4.2 Sample contingency table 23
5.1 Probe abbreviations 33
5.2 Comparison table of DAVID's and our clusters 34
5.3 Genes in sample cluster (D_{goall}, Motol data) 35
5.4 Genes in sample cluster (D_{go}, Motol data) 35
5.5 Genes in sample cluster (N_{go}, Motol data) 36
5.6 Dependence of properties of clusters on similarity threshold (ALL/AML) 37
5.7 Dependence of properties of clusters on similarity threshold (Motol) . . 38
5.8 Accuracy of classifiers on original and clustered data 40

1 Introduction

Using traditional methods to assay gene expression, researchers were able to survey a relatively small number of genes at a time. During the last twenty years several new methods which allow to measure thousands of genes simlutaneously were developed, e.g. the microarrays (for example the microarray used in [42] measured more than 7000 probes while in [11] about 32000 probes). They convert expressions of genes in a given tissue to a vector of the real numbers. If several different samples is measured, the expression matrix can be created. Usually also the different classes of tissues are investigated, e.g. tumor and control tissue. The vector of expressions of one gene for all samples is then simply called "the gene". The expression matrix can be analysed in various ways - we can e.g. try to find the differentially expressed genes or to create a classifier in order to predict class of a new, unknown sample. However because the microarrays typically consist of a small number of samples and lots of attributes (genes), it may be hard to find a classifier with high accuracy, because some classifiers have high accuracy on training data, but fail on testing data due to overfitting. Microarray data are also usually noised (see [10]), which may also negatively influence the accuracy of the classifiers. Due to these reasons it is convenient to reduce dimension of the data. It can be done by various ways, e.g. by clustering. There are lots of clustering algorithms like hierarchal clustering, K-means, correlation clustering etc., but all of them have one common disadvantage: as they do not care about a biological background of the microarrays, the resulted clusters are usually not biologically interpretable. This work describes fuzzy clustering algorithm proposed in [41] and currently implemented in web-accessible program DAVID of National Institute of Allergy and Infectious Diseases, which clusters gene lists by their associated annotation terms rather than expression levels. Our target was to rewrite the algorithm to language R in order to surpass the restrictions of the web interface: the maximum length of a clustered gene list is limited to 3000 genes and it also does not allow to modify the used annotation data. This work describes the behaviour of the algorithm on two different datasets (Motol and ALL/AML) for its various parameters and various annotation data. If we want to use the clusters in order to create the classifiers, the genes involved in clusters should be merged together into an entity called metagene. It is done in a very simple way in this work: metagene's expression for given sample is median of expressions over all genes in the cluster. The clustering and classification results are then described in detail in chapter 5.

1.1 Biological minimum

Every organism living on planet Earth has its own genetics information in which is encoded its anatomy, physiology and some another features of the organism. This information is written into the DNA - Deoxyribonucleic acid. DNA consists of two long polymers of simple units called nucleotides which run in opposite directions to each other and make a typical double helix structure. Genetic information is carried by mollecules called bases. Natural DNA has four types[1]: adenine (A), cytosine (C), guanine (G) and thymine (T). The basic genetic organization unit is a triplet codon - a set of three bases (three letters), which define one amino acid or has control meaning[2]. Several codons form a gene - the basic unit of heredity. Genes provide information needed to make proteins[3]. The process of making protein is called gene expression. It has several steps (transcription, RNA processing, translation and folding and post-translation modifications), but the complete description would rapidly enlarge the size of this work. There are lots of methods how to measure gene activity (we can also simply say "measure gene expression"). One of them is descibed in the following paragraph.

1.2 Microarray technology

The microarrays allow us to measure expression of multiple genes at once. One microarray chip typically consists of thousands of discrete spots called probes - DNA molecules with known genetic code.
As described in [8] and [9], the experiment begins with isolation of the mRNA from the investigated cells. The mRNA (messenger RNA) is a product of gene expression, it is created during the transcription step and it is a working copy of DNA. The RNA molecules are then labeled by attaching a fluorescent dye and added to the probes on the microarray. After that the hybridization process takes place: DNA has four different bases and each base is linked or is complement for the another. Adenine will always pair with thymine, and guanine with cytosine. For example the complementary sequence to TTCACG is AAGTGC. When two complementary sequences find each other they will lock together and we say, they hybridize. In this case the first sequence is located in a probe and the second is a coloured mRNA from investigated cells. After the hybridization is complete, the microarray is placed in a scanner that consists of some lasers, a special microscope, and a camera. The fluorescent tags are excited by the laser, and the microscope and camera work together to create a digital image of the array. Gene expressions are then computed from intensity and hue of the probes.
If the microarray is hybridized with DNA prepared from one biological sample, we talk about one-colour or single channel experiment. These arrays give estimations of

[1] According to [7] it is possible to artificialy create DNA with different number of bases and the new DNA is even able to be a base of a life.

[2] For example UAG encodes termination of translation.

[3] Note that some genes also encode functional RNA.

the absolute levels of gene expression. Otherwise, if two different biological samples (e.g. diseased tissue versus healthy tissue, each sample is coloured by a different fluorescent dye) are used, we talk about two-colour or two-channel microarrays and they measure a ratio between gene expressions of the samples.

1.3 Motol data

Motol data is a collection of 22 gene expression profiles from a bladder tissue. 12 patients (samples) suffered from recurrent blader cancer and 10 were control samples from patients which were successfully healed. The chip, used for obtaining the data, was the Applied Biosystems Human Genome Survey microarray, v2 (AB1700). It is an one-colour system with 32,878 probes for the interrogation of 29,098 genes but only for 18,279 the Entrez IDs[4] are known. The data is distributed as a matrix with 22 columns and 32,878 rows.
Note that Applied Biosystems has already canceled manufacture and sale of the 1700 Expression Array System instrument. For more information on the chip see [11].

[4] One of the widely accepted gene IDs. Annotations can be found only for probes with known Entrez ID.

2 Annotation sources

2.1 KEGG pathways

KEGG (Kyoto Encyclopedia of Genes and Genomes, see [12, 13, 14, 15]) is a collection of five online databases. For our purposes the most interesting is the KEGG pathway database. It contains manually drawn pathway maps and associated text information (KEGG pathway entries) for metabolism, various other cellular processes, and human diseases. In biochemistry, the metabolic pathways are series of chemical reactions occurring within the cells. They form the metabolism and are important to the maintenance of the homeostasis, the organism's ability to regulate its inner environment.
The KEGG data are freely available for academic and non-commercial users and can be searched via the web interface. For more information see http://www.genome.jp/kegg/pathway.html.

2.2 Gene Ontology

Gene ontology is a controlled vocabulary, that describes gene products in species-independent manner. Physically, the ontologies are directed acyclic graphs. If two terms (nodes) have any relation with one another, the parent term is less specialized then the child node is: for example a parent node for the nuclear chromosome (GO:0000228) is the chromosome (GO:0005694). In this example (adopted from [17]) the structure of terms is also suggested. Each term has an unique numerical identifier of the form GO:nnnnnnn (GO:0000228) and a term name (nuclear chromosome). It is also important that many GO terms have synonyms: it may be a related phrase, alternative wording, spelling or use a different system of nomenclature or it just may be a true synonym. This feature allows scientists to investigate given ontology efficiently, because their effort is not hampered by variations in terminology. The GO project has developed three ontologies describing biological processes, cellular components and molecular functions. These three structures allow to investigate the vocabulary from different biological views.
The GO vocabularies, association tools and documentation are freely available and was placed in the public domain. For more information see http://www.geneontology.org/.

2.3 Uniprot

UniProt (Universal Protein Resource) is a collaboration project of the European Bioinformatics Institute (EBI), the Swiss Institute of Bioinformatics (SIB) and the

Protein Information Resource (PIR) (see [18]). It maintains three databases: The UniProt Knowledgebase (UniProtKB), the UniProt Reference Clusters (UniRef) and the UniProt Archive (UniParc). The UniProtKB is the central hub for the collection of functional information on the proteins. It consists of the two sections: UniProtKB/Swiss-Prot (reviewed, manually annotated) and UniProtKB/TrEMBL (unreviewed, automatically annotated). The database contains all protein sequences publicly available except the synthetic sequences, pseudogenes and some other small groups. The UniRef databases provides clustered sets of sequences from UniProtKB and selected UniParc records. And finally, the UniParc contains the protein sequences (and only them, with no additional annotation) retrieved from the different source databases. In order to avoid redundancy an unique identifier is assigned to each protein.
All copyrightable parts of Uniprot database are published under Creative Commons Attribution-NoDerivs License, which means, they may be freely copied and distributed even for commercial purposes, but modified version of the databases may be distributed only with permittion of the Uniprot Consortium. For more information see http://www.uniprot.org/.

2.4 Another annotation sources

In this paragraph, we just shortly name the another annotation sources available via internet:

- Enzyme nomenclature (Enzyme Commission numbers):
  http://www.chem.qmul.ac.uk/iubmb/enzyme/.
- Online Mendelian Inheritance in Man (OMIM), database of mendelian traits and disorders: http://www.ncbi.nlm.nih.gov/sites/entrez?db=omim.
- PubMed, a database of medical bibliographic information, also contains links to full-text articles at participating publishers:
  http://www.ncbi.nlm.nih.gov/sites/entrez?db=pubmed
- PROSITE, database of protein domains, families and functional sites as well as associated patterns and profiles to identify them: http://www.expasy.ch/prosite/
- Interpro, another database of protein families, domains, regions, repeats and sites in which identifiable features found in known proteins can be applied to new protein sequences:
  http://www.ebi.ac.uk/interpro/
- Biocarta pathways: http://www.biocarta.com/genes/index.asp

There are tens of another annotation sources, so the main challenge is not "how to collect enough annotation data", but "how find relevant and non-redundant" ones.

2.5 The NCBI institute

As seen in the previous paragraph, many annotation sources is maintained by National Center for Biotechnology Information (NCBI, see [24]). The NCBI was established on November 4, 1988, as a division of the National Library of Medicine (NLM) at the National Institutes of Health (NIH). It has multi-disciplinary research group composed of computer scientists, molecular biologists, mathematicians, biochemists, research physicians, and structural biologists, but for us, more important are services and databases provided by the institute. One of them is GenBank: NIH genetic sequence database, an annotated collection of all publicly available DNA sequences. It currently contains about 85 bilion bases in 82 milion sequence records and these numbers are growing rapidly. Another important database is PubMed. It is a large collection of medical bibliographic information and allows scientists to search them via web interface. Sometimes links to full-text articles are also presented.

3 R & Bioconductor

3.1 R

3.1.1 Basic description

R is an interpreted language and environment for statistical computing and graphics. It can be considered as an open source implementation of S, the language and environment developed at Bell Laboratories by John Chambers and colleagues. There are some important differences, but much code written for S runs unaltered under R. The philosophy of R has also been influenced by Scheme, well known functional programming language. R was initially written by Ross Ihaka and Robert Gentleman at the Department of Statistics of the University of Auckland in Auckland, New Zealand, but thanks to its openness the bug reports and some source code are also sent by comunity.
The main purpose of R is statistical computing and data analysis, but it can be also easily extended via packages. There are about eight packages supplied with the R distribution and many more are available through the CRAN family of Internet sites covering a wide range of the modern statistics. Another way, how to extend R capabilities, is to call an external code written in C, C++ or FORTRAN.
The R can be run on various operating systems (*nix, Mac OS and Windows) on about 16 different platforms including i386 and x86_64. For more information about R and about its installation and maintaining see [25] and [26].

3.1.2 Calling an external C code

Because some algorithms are very computationally-intensive (e.g. clustering), it could be convenient to rewrite them into a lower level language like C. In our case a C code computes the kappa statistics (see paragraph 4.2) for all gene pairs and for about 55,000 pairs is 100 times faster than its R equivalent. However a C code must be compiled before it is first used. This is simple on *nix operating systems but could be tricky on Windows. Probably the simplest way, how to compile a dynamic-link library (dll) on Windows is to use an IDE (e.g. Dev-C++, for more information see [27].
To compile a C code to be called from R just type in OS shell

```
tsix@penguin:~$ R CMD SHLIB src/kappa.c
```

Note, that the kappa.c contains a function named `kappa()` with the following prototype:

```
void kappa(double *C11,double *C10,double *C01,double *cols,int *numOfPairs,
           double *results);
```

The function has five parameters and the sixth serves as a return variable. If no error occures, the kappa.so should appear in src directory. To call the function it is necessary to load the dynamic library into the session and it is also convenient to write a simple wrapper:

```
> dyn.load("src/kappa.so");
> kappa2<-function(C11,C10,C01,cols)
+ {
+   return(.C("kappa",as.double(C11),as.double(C10),as.double(C01),
+   as.double(cols),length(C11),as.double(array(0.1,dim=length(C11))))[[6]]);
+ }
```

It is a similar approach as can be found in [28] and the function is now called in the same way as the other R functions are:

```
> kappa2(c(1,3,2),c(2,2,0),c(0,1,0),10)
[1] 0.4117647 0.4000000 1.0000000
```

For more and advanced examples see [27].

3.1.3 Working with databases

There are two packages which serve as an abstraction layer between R and a back-end database: the Rdbi and the DBI. Both of them has similar functionality but there is only one database driver for the Rdbi (RdbiPgSQL) and all remaining db drivers are currently written for the DBI. This is the reason this paragraph describes only the DBI's capabilities.
If we want to work with a database, a device driver must be loaded first. According to an example found in [29] (the package vignette), it is done for MySQL by

```
> library(DBI)
> library(RMySQL)
> drv <- dbDriver("MySQL")
```

and for SQLite by

```
> library(DBI)
> library(RSQLite)
> drv <- dbDriver("SQLite")
```

and similarly for another DBMSs. After the driver is loaded, we simply connect to the database via `dbConnect()`. For MySQL type (with appropriate arguments)

```
> dbconn <- dbConnect(drv, "information_schema", "root", "")
```

The second argument is the database name, the third is the login name and the last one is a password. For SQLite the second argument is the file name the database is stored in:

```
> dbconn <- dbConnect(drv, "/home/tsix/R/x86_64-pc-linux-gnu-library/2.8/motolan
n.db/extdata/motolann.sqlite")
```

Being connected to the database we can send some SQL queries (this example is for information_schema database):

```
> dbListTables(dbconn) #List all tables in database
 [1] "CHARACTER_SETS"
 [2] "COLLATIONS"
 ...
> dbListFields(dbconn,"CHARACTER_SETS") #List fields in CHARACTER_SETS
table
[1] "CHARACTER_SET_NAME"   "DEFAULT_COLLATE_NAME" "DESCRIPTION"
[4] "MAXLEN"
> xx<-dbGetQuery(dbconn,"SELECT * FROM CHARACTER_SETS LIMIT 2")
> is.data.frame(xx)
[1] TRUE
> xx
  CHARACTER_SET_NAME DEFAULT_COLLATE_NAME              DESCRIPTION MAXLEN
1               big5      big5_chinese_ci Big5 Traditional Chinese
 2
2               dec8      dec8_swedish_ci        DEC West European
 1
```

When we are done it is convenient to free up the allocated resources:

```
> dbDisconnect(dbconn)
[1] TRUE
> dbUnloadDriver(drv)
[1] TRUE
```

These examples obviously do not cover all facilities of the DBI package. For more information see the DBI documentation file [DBI location]/doc/DBI.pdf. A path to this file can be also discovered by

```
> file.path(system.file(package = "DBI"),"doc","DBI.pdf")
```

3.2 Bioconductor

Bioconductor is an open source and open development software project to provide tools for the analysis and comprehension of genomic data. It is a collection of R packages with various purpose - there are infrastructure packages, which provides a technical background for another packages (e.g. networking, working with GUI etc.), experiment data, which provide miscellaneous microarray and another data, annotation packages and packages focused on DNA microarray and another data analysis. As described in [30], each Bioconductor package contains at least one vignette - a document that provides a textual, task-oriented description of the package's functionality.
Some packages with a relation to this work are shortly described in the following paragraphs.

3.2.1 AnnotationDbi

From Bioconductor 2.3, the recommended package for collecting annotations is the AnnotationDbi. It is much easier to use this package when comparing with the AnnBuilder[5]. It uses the SQLite instead of R environments for storing data[6], so it has better performance and more possibilities and the data can be even used independently on R.
Let's assume we want to build an annotation package for a human microarray. After installation of the AnnotationDbi package the only thing we acctually need is a file with two columns separated by a tab where the first contains manufacturer's IDs and the second some sort of widely accepted gene accession. For example, this file could look like that:

```
38642_AT       214
1244_AT        6773
1461_AT        4792
35687_AT       4515
31558_AT       27251
```

In this case the second column contains Entrez IDs. File must not have a header and the missing values are indicated by `NA`s. To successfully build a human annotation

[5] Note that the AnnBuilder is deprecated and it is not distributed with the Bioconductor 2.4 and newer.

[6] Fortunatelly these environment variables are built as well, so old scripts need not be rewritten.

package we need to have the human.db0 package installed and then just call function `makeHUMANCHIP_DB()` with apropriate arguments. On the other hand this document comes with another R function with the same purpose, so we will shortly describe its usage.

3.2.1.1 Function `buildAnnPackage()`

This function is located in the src/buildAnnPackage.R and has similar arguments like the `makeHUMANCHIP_DB()`:
prefix - A main part of the new package name (new package is named {*prefix*}.db)
fileName - A file with a map between manufacturer's IDs and some sort of widely accepted gene accession.
outputDir - A directory the new package is saved to.
baseMapType - Type of the second column in the map file. Possible values are "gb" (genebank accesssions), "ug" (unigene), "eg" (Entrez genes) and "refseq" (refseq accessions).
version - Version of the new package.
manufacturer - Microarray manufacturer.
chipName - Name of the used chip.
manufacturerUrl - Manufacturer's URL.
affy - If we have an annotation file for an Affymetrix chip, we can set this parameter to TRUE and function automatically parse such a file and produce a mapping described in this chapter.
updateSource - Determines whether update the human.db0 package first. If true, about 150MB file will be downloaded.
verboseLevel - Verbose (0=quiet, 1=basic).

3.2.1.1.1 Working example

To build an annotation package for the data/motolann.data datalist, start `R` console, set appropriate working directory (by the `setwd()` function) and then type

```
> buildAnnPackage("motolann","data/motolann.data","output/motolann");
```

Note that the output/motolann is a directory and must already exist. After a while a new package is created and can be installed. In the operating system shell type

```
tsix@penguin:~$ R CMD INSTALL output/motolann/motolann.db
```

and the package is installed. To verify it type in R session

```
> library(motolann.db)
> dbconn<-motolann_dbconn()
> dbGetQuery(dbconn,"SELECT * FROM probes LIMIT 5")
  probe_id accession   _id
1 "100002"      <NA>    NA
2 "100003"      <NA>    NA
3 "100027"      <NA>  6644
4 "100036"      <NA> 13405
5 "100037"      <NA>  6346
```

Note that column `_id` contains database specific IDs and has no relation with Entrez IDs. To obtain structure of all tables type

```
> motolann_dbschema()
--
-- HUMANCHIP_DB schema
-- ===================
--

-- The "genes" table is the central table.
CREATE TABLE genes (
  _id INTEGER PRIMARY KEY,
  gene_id VARCHAR(10) NOT NULL UNIQUE           -- Entrez Gene ID
);

-- Data linked to the "genes" table.
CREATE TABLE probes (
  probe_id VARCHAR(80) PRIMARY KEY,             -- manufacturer ID
  accession VARCHAR(20) NULL,                   -- GenBank accession number
  _id INTEGER NULL,                             -- REFERENCES genes
  FOREIGN KEY (_id) REFERENCES genes (_id)
);
...
```

3.2.2 Safe

The Safe (Significance Analysis of Function and Expression, the method proposed by [33]) is an example of package, that implements some algorithm focused on the microarray data analysis. This one tries to identify genes having significant association with a clinical outcome[7]. Hovewer it does not work in gene-by-gene

[7] In our case genes having significant association with bladder cancer reccurency.

manner (like simple standard parametric tests), but it investigates significance of gene categories, which are sets of genes with similar annotations or function.

3.2.2.1 Working example

In this experiment we analyse Motol data by metabolic pathways. Before we can run any experiment the data should be preprocessed first. The first thing we need is an expression matrix of all genes:

```
> library(safe)
> motol1 <- read.csv("data/motol.csv")
```

The motol.csv is expected to contain a matrix $(n \times m + 1)$, where n is number of genes and m number of samples and the first column contains the gene names.
Next the row names are set to the gene names and the column with gene names is ommited from the original matrix:

```
> motol<-motol1[,2:dim(motol1)[2]]
> probes.vector<-(motol1[, 1])
> rownames(motol) <- probes.vector
```

Doing these steps we are now able to correctly interpret the results. In the following step we create mapping between genes and KEGG pathways. This can be done easily thanks to the annotation package built in paragraph 3.2.1:

```
> library(motolann.db)
> probes.pathways<-as.list(motolannPATH)
```

The last thing we do before the experiment is the creation of incidence matrix. Its row names are genes and column names are pathways names. If a gene belongs to a pathway, the correspondent matrix item has value 1 and 0 instead. Note that in Safe version 2.0 the incidence matrix can be generated automatically by function `safe()`, but while running multiple examples its convenient to build it externally due to efficiency reasons.

```
> C.motolmatrix <- getCmatrix(probes.pathways, as.matrix = TRUE,
+ present.genes = as.character(probes.vector),min.size=20)
```

Note that although this command worked well with older versions of Safe, it fails when running Safe 2.0 from unknown reason. The argument `min.size=20` means only the pathways with at least 20 members will be considered.
At this moment we are ready to call function `safe()`, which do the analysis. It has three main arguments: expression matrix, incidence matrix and classification

vector. We also set the random generator seed in order to ensure reproducibility of the experiment.

```
> set.seed(496)
> motol.cl<-c(0,0,0,0,0,0,0,0,0,0,1,1,1,1,1,1,1,1,1,1,1,1)
> results <- safe(motol, motol.cl, C.motolmatrix)
> results
SAFE results:
  Local: t.Student
  Global: Wilcoxon

      Size    Stat Emp.p
05213   59 1189939 0.003
04520   97 1876139 0.004
00960   24  509463 0.009
04670  145 2655475 0.012
...
```

Note that hsa:05213 is the Endometrial cancer, hsa:04520 the Adherens junction, hsa:04520 the Alkaloid biosynthesis II and hsa:04670 the Leukocyte transendothelial migration and the remaining categories were not included due to typographical reasons. The p-values seems to be pretty low, but because we were considering many categories simultaneously, they must be adjusted to reflect it. Hovewer this is not done in this work, because it primarily deals with a different issue.
The `safe` function works in two stages. Local statistics assess the association between expression and the response of interest in a gene-by-gene manner, and a global statistic measures the extent of association in genes assigned to a category relative to their complement. The default local statistics is the Student's t-statistic and the default global one is Wilcoxon rank sum. Function `safe()` also supports another statistics, sources of functional categories and different experimental designs. For further information see the vignette and some additional tutorials and examples are available at http://www.duke.edu/~dinbarry/SAFE/.

3.2.3 TopGO

Another package containing a category analysis algorithm is the topGO. Its primary purpose is to provide an enrichment analysis (described in [36]) for gene ontologies. The package is able to work with different test statistics and with multiple GO graph algorithms.

3.2.3.1 Working example

A typical topGO session, as it is also described in [35], requires first to build an object of class topGOdata which encapsulates gene scores, used microarray and the annotation data. Analogous to the previous paragraph we load the data from csv file:

```
> library(siggenes)
> library(topGO)
> library(motolann.db)
> motol<-read.csv("data/motol.csv",header=T)
> motol.cl<-c(0,0,0,0,0,0,0,0,0,0,1,1,1,1,1,1,1,1,1,1,1,1)
> rownames(motol)<-motol[,1]
> motol<-motol[2:dim(motol)[2]]
> myGeneNames<-rownames(motol)
```

The only other thing needed to build a topGOdata object is to compute the gene scores. In most cases it means to compute their p-values for the differential expression[8].

```
> geneList <- getPvalues(motol, classlabel = motol.cl, alternative = "greater")
```

Unfortunately all genes in Motol data has exactly the same score so in this way it is impossible to correctly select a set of interesting genes.

```
> geneList[1:10]
282357 224689 226296 220448 226590 225638 219335 227184 218879 219301
     1      1      1      1      1      1      1      1      1      1
> unique(geneList)
[1] 1
```

If we want to continue in our example, the scores must be computed in a different way. Hovewer, when doing that is difficult to estimate whether the topGO results would have a biological sense so the remaining part of the example should be considered just as a technical example of the topGO's abilities.
To create an interesting gene set we use the siggenes package (see [37]) and its methods implementing significance analysis of microarrays:

```
> library(siggenes)
> sam.out <- sam(motol, motol.cl, rand = 123, gene.names = myGeneNames)
> sum.sam.out <- summary(sam.out, 0.12, ll = FALSE)
```

[8] The p-values are actually also adjusted using false discovery rate.

```
> myInterestedGenes<-rownames(sum.sam.out@mat.sig)
> geneList <- factor(as.integer(myGeneNames %in% myInterestedGenes))
> names(geneList) <- myGeneNames
```

The last thing we need in order to build a topGOdata object is to select one of the three ontologies. Posible values are BP (biological process), MF (molecular function) a CC (cellular component) and int his example we select the second one:

```
> GOdata <- new("topGOdata", ontology = "MF", allGenes = geneList,
+ annot = annFUN.hgu, affyLib = "motolann.db")

Building most specific GOs .....          ( 2536 GO terms found. )

Build GO DAG topology ..........          ( 2965 GO terms and 3532 relations.
)

Annotating nodes ...............          ( 16731 genes annotated to the
GO terms.
)
```

The TopGO implements three different graph algorithms (denoted as classic, elim and weight) and two different types of test statistics: Fisher's exact test which is based on gene counts, and a Kolmogorov-Smirnov like test which needs gene scores. Because scores of the genes has not been properly discovered, we use just the first of them.
The interface for the algorithms is function `getSigGroups` which has two arguments: an object of class topGOdata and object of class groupStats that contains information about used algorithm.

```
> test.stat <- new("classicCount", testStatistic = GOFisherTest, name =
+ "Fisher test")
> resultFis <- getSigGroups(GOdata, test.stat)
> test.stat <- new("elimCount", testStatistic = GOFisherTest, name = "Fisher
test",
+ cutOff = 0.01)
> resultElim <- getSigGroups(GOdata, test.stat)
> test.stat <- new("weightCount", testStatistic = GOFisherTest,name = "Fisher
test",
+ sigRatio = "ratio")
> resultWeight <- getSigGroups(GOdata, test.stat)
```

The results can be summarized by function `genTable`:

```
> l <- list(classic = score(resultFis), elim = score(resultElim), weight
=
+ score(resultWeight))
> allRes <- genTable(GOdata, l, orderBy = "weight", ranksOf = "classic",
top = 10)
```

Variable `allRes` now holds the first ten GO terms ordered by results of algorithm weight.

	GO.ID	Term	Annotated	Expected	Rank in classic	classic	elim	weight
1	GO:0004618	phosphoglycerate kinase activity	2	0.00	1	0.0012	0.0012	0.0012
2	GO:000499	vasoactive intestinal polypeptide recept...	5	0.00	3	0.0030	0.0030	0.0030
3	GO:0004726	non-membrane spanning protein tyrosine p...	12	0.01	5	0.0072	0.0072	0.0072
4	GO:0005509	calcium ion binding	1081	0.65	9	0.0229	0.0229	0.0229
5	GO:0004722	protein serine/threonine phosphatase act...	50	0.03	10	0.0295	0.0295	0.0295
6	GO:0008307	structural constituent of muscle	52	0.03	11	0.0307	0.0307	0.0307
7	GO:0008201	heparin binding	91	0.05	12	0.0531	0.0531	0.0531
8	GO:0030145	manganese ion binding	157	0.09	18	0.0900	0.0900	0.0900
9	GO:0005506	iron ion binding	307	0.18	21	0.1691	0.1691	0.1691
10	GO:0000287	magnesium ion binding	449	0.27	22	0.2382	0.2382	0.2382

Table 3.1 topGO results

3.2.4 Macat

As the time went on it appeared the biologically valid functional category the microarray data can be analysed by is also a gene position on a chromosome (see [38]). The idea is simple - if exists a place on chromosome with "many" differentially expressed genes (they even need not to share any biological function), this place is considered as interesting one and is worth further investigating.

3.2.4.1 Basic theory

For each gene a parameter $d(i)$ which measures its significance is computed:

$$d(i) = \frac{\overline{x}_A(i) - \overline{x}_B(i)}{s(i) + s_0}, \tag{3.1}$$

where $\overline{x}_A(i)$ and $\overline{x}_B(i)$ are the mean expression levels of gene i in group A and B respectively, $s(i)$ is the pooled standard deviation of the expression values of the gene i and s_0 is constant for all genes. To prevent a high statistic for genes with a very low standard deviation the s_0 is set to the median over all gene standard deviations $s(i)$. Note that the macat also allows to compute Student's classical t-statistic instead of $d(i)$.
Microarrays usually measure the expression of a limited number of genes and their distances on the chromosome differ greatly. Since we want to compute differential expression statistics for the larger chromosomal regions, we need a method to interpolate scores between the measured values. Formally it is a regression problem.

Let $y = f(x)$ be t-score of chromosomal coordinate x. Values of $f(x)$ are known just for few observations x_i and we want to smoothly estimate the function. This requirement can be satisfied by kernel methods. The macat is able to work with the three different kernel approaches:

1. kNN: For every chromosomal coordinate compute the average of the k nearest genes.
2. Radial basis function: For every chromosomal coordinate compute the average over all genes weighted by distance as explained in detail below.
3. Base-pair distance kernel: Similar to the k-Nearest-Neighbors, but using this kernel the average is taken over all genes within a certain radius of the position, whose value has to be determined.

It is very important to properly choose the kernel parameters. For example for kNN it is recomended to set k to cover approximately 10% of the genes. Another biologically motivated heuristics can be found in the package vignette.

3.2.4.2 Working example

As well as in the previous paragraphs we begin by loading and preprocessing the Motol data:

```
> library(macat)
> library(motolann.db)
> motol.cl<-c(0,0,0,0,0,0,0,0,0,0,1,1,1,1,1,1,1,1,1,1,1,1)
> motol<-read.csv("data/motol.csv",header=T)
> rownames(motol)<-motol[,1]
> motol<-motol[2:dim(motol)[2]]
```

Then we create an object encapsulating microarray data and classification vector and which will be used as an input for used algorithms:

```
> motolmacat<-preprocessedLoader(motol,"motolann.db",motol.cl, rdafile=FALSE,
+             tabfile=FALSE)
```

If we decide to use the kNN kernel function, its optimal length k may be computed by function `evaluateParameters()`:

```
> evalkNN6 = evaluateParameters(motolmacat, class = "1", chromosome = 6,
+            kernel = kNN, paramMultipliers = c(0.01, seq(0.1, 2, 0.1),2.5))
```

The argument `class=1` means, the optimal k is computed considering the samples with cancer reccurency. The argument `paramMultipliers` determines which k are tested. Is valid, that

$$k = \frac{n_j.\overrightarrow{p_i}}{10} \tag{3.2}$$

where $\overrightarrow{p_i}$ is the i-th item of `paramMultipliers` and n_j is the number of genes on j-th chromosome (in this case on the sixth one).

```
> evalkNN6$best
$k
[1] 37
```

The last step is to run the algorithm and summarize results:

```
> e1 = evalScoring(motolmacat, class = "1", chromosome = 6, nperms = 1000,
+      kernel = kNN, kernelparams = evalkNN6$best, cross.validate= FALSE)
Investigating 12 samples of class 1 ...
Compute observed test statistics...
Building permutation matrix...
Compute 1000 permutation test statistics...
250 ...500 ...750 ...1000 ...
Compute empirical p-values...
Compute quantiles of empirical distributions...Done.
Computing sliding values for scores...
Compute sliding values for permutations...
All done.
```

Unfortunately the results show, that chromosome 6 is probalby not related with the bladder cancer reccurency:

```
> getResults(e1)
NULL
> plot(e1)
```

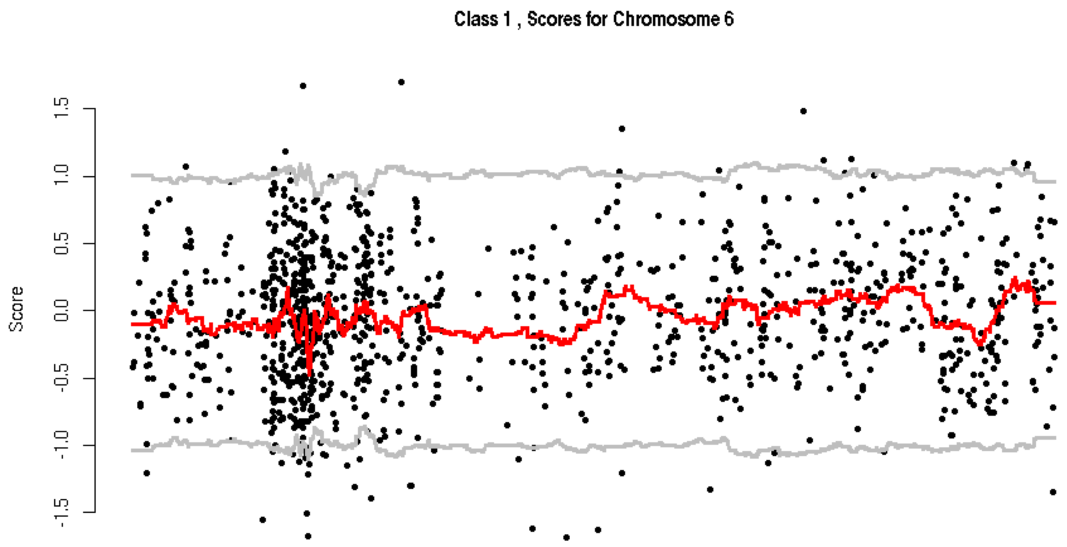


Figure 3.1 Macat results for chromosome 6

There are no significant regions[9] on chromosome 6, which can be associated to recurrency of bladder cancer and this result does not change at all 22 remaining chromosomes.

3.3 Information sources

The R language and environment can be downloaded from http://www.r-project.org/. There are also links to mailing lists, book annotations (usually in English, but some books are also written in German and even in Czech), bug tracking and other useful links. The reference manual of R base package can be found in various mirrors, e.g. http://stat.ethz.ch/R-manual/R-patched/library/base/html/00Index.html.
Bioconductor community maintanis web site http://www.bioconductor.org/, from which Bioconductor and its packages can be downloaded.

[9] In the opposite case the red line would leave the belt between the two grey horizontal lines.

4 Clustering

A typical microarray experiment produces a huge amount of data which may very complicate the following analysis. The first trouble is purely the amount - some algorithms consume a lot of memory by design and may be very computationally-intensive. These algorithms may run for a long time or even never finish: the memory runs out or the scientist dies before the algorithm terminates. Classification of such data is also hard, because there are thousands of attributes (genes) in one microarray and just a few samples and the classifiers may be overfitted. Another trouble is connected with the microarray technology. On the one hand, it is able to measure the expression levels of thousands of genes simultaneously, but on the other hand, the results are noised (see [10]) and contain a lot of unreliable probes which may negatively affect the accuracy of classifiers. These are the reasons, why we try to reduce dimension of the data. One approach to do that is clustering.
There are many well known clustering algorithms like hierarchal clustering, K-means, correlation clustering etc. All of these algorithms have many implementations and work well, but this work describes a different approach, which is, according to [41], more convinient for biological data than its "classic" coleagues.

4.1 DAVID

The Database for Annotation, Visualization and Integrated Discovery (DAVID, see [40]) is a web-accessible program, that provides comprehensive set of functional annotation tools for investigators to understand biological meaning behind large list of genes. The first version of DAVID/EASE was released in 2003 and since then, series of novel algorithms and services were integrated. Nowadays it allows scientists to annotate gene lists, discover enriched functional-related gene groups, has rich visualization abilities, allows to cluster redundant annotation terms by gene similarity and genes by functional similarity, etc. The main target of this work is to implement DAVID's fuzzy clustering algorithm to cluster genes by functional similarity. The Output of the implementation should be at least comparable with the original DAVID's results. One may ask a question, why to duplicate the parts of DAVID functionality, when they are accesible via web interface for free? There are at least two reasons why to do that:
The clustering is a computationally-intensive task and a performance of DAVID's servers is not unlimited, so maximum length of gene lists is limited to 3000 genes. Unfortunately e.g. Motol data contains more than 18000 annotated genes so this gene list cannot be processed via the web interface.
Although DAVID is highly integrated and powerful tool, it is not able to do all imaginable work. For example imagine, we want to create a classifier, which separates samples with the cancer reccurency from the control ones and we want to

use DAVID's clusters. They would have to be collected manually from DAVID web pages, which may be an activity vulnerable to errors and if the length of the gene list was greater than 3000, the analysis would not be possible.

4.2 Fuzzy clustering algorithm

The algorithm described in this paragraph was proposed in [41]. Because there is no short name of the algorithm in relevant papers, for purposes of this and only this work we denote it as the DAVID[10].
Classical algorithms like K-means cluster microarray data using gene expression levels. A gene can be imagined as a vector in m-dimensional space (m is number of samples) and the distance among the genes can be given for example by Euclidean measure. Although these methods work well on technical data, they completely neglect a biological background of microarray data, which may not be acceptable in some cases. The DAVID proposes a different approach - it does not use the gene expressions, but it works with the annotation terms. The more common terms have two genes the more similar they are. The DAVID also tries to reflect, that one gene may participate in more than one functional groups which means, one gene may be presented in more than one final clusters. This is the reason we are calling this algorithm fuzzy. The DAVID has also ability to automatically recognize outliers and discard them. Thanks to these features the genes grouped together into one cluster has similar biological function, so DAVID produces better clustering results than the classical methods because the clusters are well biologically interpretable (this statement is promoted by three studies and another examples described in [41]).

4.2.1 Kappa Statistics

In the DAVID algorithm, a gene distance is measured by the kappa statistics also known as Cohen's kappa coefficient. Let's assume we measure a similarity of two genes using ten annotation terms as shown in table 4.1:

	$\mathbf{t}_1$	$\mathbf{t}_2$	$\mathbf{t}_3$	$\mathbf{t}_4$	$\mathbf{t}_5$	$\mathbf{t}_6$	$\mathbf{t}_7$	$\mathbf{t}_8$	$\mathbf{t}_9$	$\mathbf{t}_{10}$
gene A	1	1	0	1	0	1	1	1	0	1
gene B	1	1	1	1	0	0	1	0	0	1

Table 4.1 Sample gene-annotation matrix

[10] Technically this is wrong because DAVID is integrated web tool, that allows annotate, analyse and cluster gene lists and the discussed clustering algorithm is just part of one of its provided services (Gene Functional Classification).

The gene A is annotated by terms t_1, t_2, t_4, t_6, t_7, t_8 and t_{10} and the gene B by t_1, t_2, t_3, t_4, t_7 and t_{10}, so they share five annotation terms (t_1, t_2, t_4, t_7 and t_{10}) at all. These numbers can be arranged into a table:

		Gene A		
		1	**0**	**Row total**
Gene B	**1**	5 ($C_{1,1}$)	1 ($C_{0,1}$)	6 ($C_{*,1}$)
	0	2 ($C_{1,0}$)	2 ($C_{0,0}$)	4 ($C_{*,0}$)
Column total		7 ($C_{1,*}$)	3 ($C_{0,*}$)	10 (T_{ab})

Table 4.2 Sample contingency table

The kappa statistics is then computed using following formulas:

$$O_{ab} = \frac{C_{1,1} + C_{0,0}}{T_{ab}} = \frac{5+2}{10} = 0.7 \tag{4.1}$$

$$A_{ab} = \frac{C_{*,1} * C_{1,*} + C_{*,0} * C_{0,*}}{T_{ab} * T_{ab}} = \frac{6*7+4*3}{10*10} = 0.54 \tag{4.2}$$

$$K_{ab} = \frac{O_{ab} - A_{ab}}{1 - A_{ab}} = \frac{0.7 - 0.54}{1 - 0.54} = 0.348 \tag{4.3}$$

Kappa greater than zero indicates, the genes A and B are in agreement, more so than by random chance. If the K_{ab} was equal to 0, genes would share just random amount of annotation terms and $K_{ab} = 1$ indicates, the genes are annotated by exactly the same terms. Some another features of kappa statistics are further discussed in paragraph 5.1.1.

4.2.2 From original data to final clusters

Having defined a suitable metrics we can come up to the algorithm now. Let's assume, we have positioned each gene in a two dimensional space, so the situation may look as in figure 4.1:

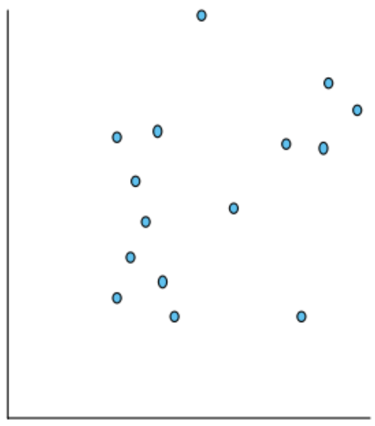


Figure 4.1 The genes before clustering

Each point represents one gene. Also note that the absolute coordinates of the points are not important, the only important information contained in the figure are distances among particular points.
Each gene has a chance as a medoid to form an initial seeding group. It covers all genes, which are enough closely related with medoid (default threshold value is kappa>0.35, it is an algorithm's parameter). In order to be declared as the qualified one an initial seed must meet two necessary conditions: it must contain at least 4 members (also an algorithm's parameter) and at least 50% of its members must have close relationships each other (e.g. kappa>0.35). There are two descriptions of the algorithm in available bibliography - one is straightly the article in Genome Biology and the second is in its additional file 13. The first example is strictly graphical (in fact, the figures 4.1, 4.2, 4.3 and 4.4 have been adopted from it), but the second, although it is not explicitely said, does not use the medoid while investigating the second condition. It means the relationships between medoid and the other members are not counted so they cannot help to meet the 50% threshold. However there is not explicitely said whether neglect or not to neglect the medoid so this is another slightly unwanted algorithm's parameter. The state after the second step is visualized in the figure 4.2:

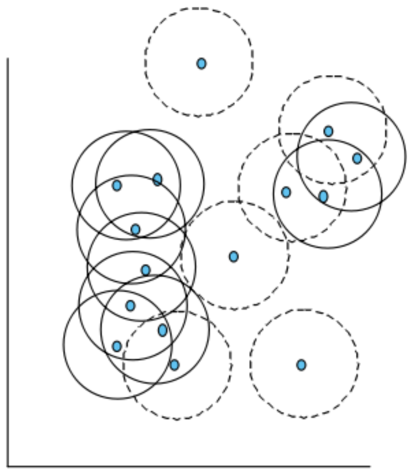


Figure 4.2 Qualified and unqualified initial seeds

The qualified seeds are indicated by solid-line circles while the unqualified ones by dashed-line circles.
Every qualified initial seeding group is iteratively merged with each other to form a larger group: two seeds are merged if and only if they share majority (e.g.>50%) of its members.

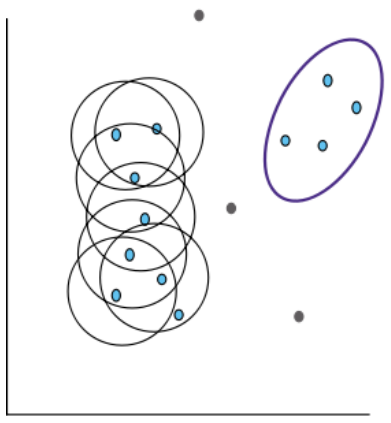


Figure 4.3 Groups in the middle of iterative merging

Figure 4.3 shows the state in the midle of merging. The cluster in thicker oval cannot be furthermore merged with another seeds as it is already stable. Medoids, which have not been able to form an initial seed are greyed out and are considered as outliers. The process of merging continues until all seeds are merged into final clusters:

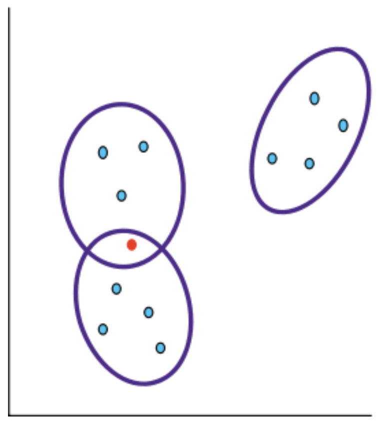


Figure 4.4 Final clusters

After the merging is done, three final clusters are formed. One gene (in red) belonging to two clusters represents the fuzziness capability of the algorithm. Outliers has been simply removed and they are excluded from further analysis.

4.2.3 Algorithm's features

The algorithm described in the previous paragraph has following features:

- The fuzzines: as shown in previous paragraph, one gene may belong to more than one final clusters.
- It automatically discards the outliers. This feature has also been shown in previous paragraph.
- The number of final clusters is determined automatically. This may be both advantage and disadvantage. It is probably the advantage for biologists, because they need not to estimate a parameter, which is more technical than biological, but it is at least complication for computer scientists, because it makes harder the comparison with another clustering algorithms.
- Results of clustering are very dependent on quality of available annotation terms. It means that different annotation terms used for clustering imply different clustering results. However this cannot be felt as an disadvantage, because DAVID clustering algorithm is designed to work with annotation terms so this dependence is not a bug, but a feature.
- Clustering results do not depend on the gene expression levels. It means, the algorithm can be run just once for given microarray chip and all other assays using this chip can simply exploit the results. This is a huge advantage especially for larger chips.

4.2.4 Implementation

The fuzzy clustering algorithm described in previous paragraphs was written in language R. Kappa statistics is computed by a C function called from R and the annotation data are stored in the SQLite database. The script can be found on appended CD in src/cluster.R. Interface for the algorithm is the function `davidCluster()` and it works in several steps:
Preprocessing of the annotation data. The data from given annotation package are inserted into one database table, duplicated rows are deleted and aggregate information like lengths of gene anotations are obtained. SQL is used plentifully in this step.
Building pre-initial seeds[11]. These are groups of genes, which are enough closely related with the medoid but we do not care about their number or if they have close relationships each other.
Building the initial seeds. All pre-initial seeds with less than a members ($a = 4$ by default) or with members not having close relationships each other are excluded from the next steps.
Merging initial seeds into final clusters.
After the function is done a list of clusters is returned. Each cluster is defined by a vector of database gene IDs, which should be converted to Entrez or probe IDs. For more information see files in directory "examples" on appended CD.

[11] Note that the “pre-initial seed” does not belong to the official nomenclature as it is defined in [41].

5 Results

5.1 On parameters identification

As mentioned in the previous paragraphs, the algorithm has several parameters. If we want to make our results comparable with the DAVID ones, the parameters that DAVID uses in its implementation and the annotation data should by identified and collected first. Although we solved mainly technical problems, the estimation of the $C_{0,0}$ and some other parameters, that control deciding whether an ititial seeding group is qualified or not, was nontrivial as it couldn't be determined from [41]. The annotation data are very important for the algorithm so their collecting should be also described here.

5.1.1 Determining $C_{0,0}$

In fact, the $C_{0,0}$ is not a parameter of the algorithm, however there is very strong reason to set it to fixed value for all gene lists. Otherwise a gene pair would be more related in larger gene lists (and with greater kappa) which is not a wanted feature. We would expect the relationship of two genes would be independent on the list the genes are part of. This is the reason, why DAVID does not compute $C_{0,0}$s from length of given gene list, but from number of all available annotation terms. Currently it is about 103590 terms and this value is also used in our implementation. Note that it is not important, if the value is 103590, 90000 or 110000. The reason is obvious from figure 5.1:

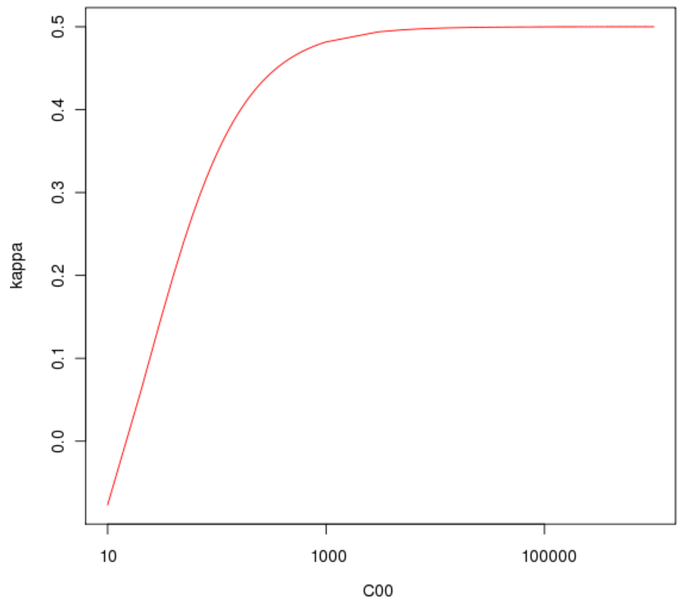


Figure 5.1 Dependence of kappa on $C_{0,0}$

In this case $C_{1,1} = 20$, $C_{0,1} = 10$ and $C_{1,0} = 30$ but the shape of the curve does not change for different values. The similar issue is also discussed in [41] in additional file 11.

5.1.2 Collecting annotation data

Annotation data are essential for proper work of the algorithm. There are many ways how to get them. This implementation rely on R annotation packages built by AnnotationDbi, but they don't contain all data used by DAVID. BioCarta pathways, Swiss-Prot keywords, BBID pathways, SMART domains, NIH genetic association DB, COG/KOG ontology, InterPro domains, and PIR superfamily names are not compiled into package and must be downloaded separately. Biocarta pathways cannot be downloaded for free due to its proprietary licence, but the others are offered by DAVID to download from its knowledgebase[12]. However for a long time it was impossible to gain data directly from DAVID due to unspecified troubles with licence. In the end of May 2009 the data were released[13] and could be succesfully downloaded. A majority of the annotation terms was last updated in the beginning of 2008[14], so they slightly differ from the data obtained by the AnnotationDbi package, which were collected in April 2009.
DAVID allows to investigate in detail which annotation terms were used for the kappa computation. For example consider a list with two probes: 37542_AT and 31558_AT. Load them to DAVID via the web interface and use the Gene Functional Classification tool in order to cluster them. Of course, no cluster appears but if we click on the button "2 genes from your list are not in the output" and then to the link "RG" (related genes) in row with affy_id 37542_AT, we can see the kappas between the given genes. By clicking on some kappa value (for example on 0.57) the browser is redirected to a page, where all annotation terms used for computation are listed and there is also a contingency table. Currently 2 terms are common and three different ($C_{0,1} = 0$ and $C_{1,0} = 3$). But now, return to the beginning and use the same list by Functional Annotation tool. If we click on the button "Functional Annotation Table", the unexpected thing appears: displayed annotation terms are not the same as the terms used for computation of kappa! There are more common terms ($C_{1,1} = 6$) and also more different ones ($C_{0,1} = 0$ and $C_{1,0} = 4$) so for these values the kappa is not 0.571 but 0.750[15]. In addition, it seems the data used by the

[12] Note that these data must be then added to given annotation package which is very simple task, because it stores data in SQLite database and additional terms can be added by `CREATE TABLE` and `INSERT INTO` queries.

[13] Unfortunatelly the licence forbids to make any copies, except for its internal use, so they couldn't be distribued on appended CD.

[14] See http://david.abcc.ncifcrf.gov/content.jsp?file=update.html.

[15] The $C_{0,0}$ is 103589 in both cases.

Gene Functional Classification are not available to download, which means, we will not be able to make our results the same or very similar to the DAVID ones. It only remains to tune the algorithm parameters in order to generate reasonable results.

5.1.3 Parameters for internal seed relatedness

While building pre-initial seeds and merging final clusters are simple and clear steps, it turned out that a decision whether a seed is qualified or not is much more complicated. We manually created the distance matrix for random 20 genes[16] in order to explore, how this step is done by DAVID. During the experiments, parameter Multiple Linkage Threshold was set to 1.0 in order to disable final merging. Choosen gene lists were also tested manually. Despite this effort there are two testcases we are still unable to fully explain and make DAVID's behaviour clear.
Consider a short list containing five probes: 1602_AT, 1007_S_AT, 1267_AT, 1432_S_AT and 136_AT. Upload it to DAVID, cluster by the Gene Functional Classification and one cluster with four members appears and 136_AT is considered to be an outlier. But now add the 38642_AT into the list and rerun the clustering. Two clusters appear but only the first is interesting for us. It contains the same members as the cluster created from shorter list, but in addition, the 136_AT is included. Just remember the results are not influenced by final merging so these clusters should also be the initial qualified seeds and the initial seeds are either qualified or discarded and there is no way how to exclude only several genes from it. Although 38642_AT is not a member of these clusters, in the first case the 602_AT, 1007_S_AT, 1267_AT, 1432_S_AT and 136_AT is not qualified but in the second case it is! There is one possible explanation, but is neither simple nor elegant. It assumes the program runs exactly in these steps:

1. Build pre-initial seeds.
2. Discard all pre-initial seeds with less then 4 (e.g.) members.
3. Investigate an internal seeds relatedness. In this step no kappas are computed. For example consider a sample seed $S = \{G_m, G_1, G_2, G_3\}$ and assume, we want to investigate the relatedness of the G_2 with remaining genes. Instead of computing the kappas we make an intersection of the S with another initial seed, in which G_2 is the medoid. A number of genes in the result set is then the number of genes the G_2 is related with. Of course, the intersection contains G_m and G_2 and they should not be counted but it is not important for now. If an initial seed with the medoid G_2 was to small and was discarded, the G_2 is automatically related with no genes. This feature can explain DAVID's behaviour and is also used in our implementation.

[16] All of them were members of Demolist 2, which is a pre-built gene list for new users to easily get into and test DAVID functions.

The second testcase also compares two seeds: 1432_S_AT, 1602_AT, 1267_AT, 1007_S_AT, 136_AT and 136_AT, 1007_S_AT, 1432_S_AT, 38642_AT (the first probes are the medoids). The first one is not a qualified seed and the second is. Even if we use the approach proposed in the previous testcase, the first seed (unqualified) seems to have better internal relatedness, than the second (qualified) has. For example let's count relations of the 1007_S_AT with the rest of the first seed. According to the distance matrix, we have created for 20 random genes, the 1007_S_AT is related with the 1432_S_AT, 1602_AT, 1267_AT and 136_AT, totally with 4 genes from 4 possible. If we substract the 1432_S_AT (medoid), it is 3/3. However e.g. 136_AT is related only with the 1432_S_AT and 1007_S_AT (2/4), but if we have already discarded the initial seed with medoid 136_AT, it is related with no gene. After we count close relations for all genes in both seeds, the genes in the first seed have more close relations than the genes in the second seed and the result does not change for any variation of the computation. Despite that, DAVID considers the first seed unqualified and the second one qualified. We are currently unable to explain this behaviour.

5.2 Clustering

A typical user workflow diagram may look as in the figure 5.2:

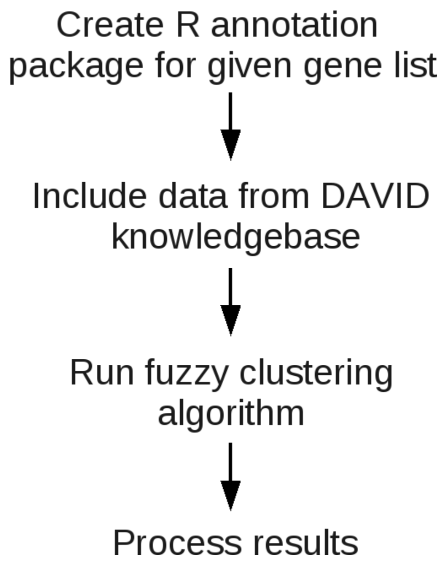

Create R annotation package for given gene list
Include data from DAVID knowledgebase
Run fuzzy clustering algorithm
Process results

Figure 5.2 Clustering workflow diagram

If we want to cluster the gene list more times or cluster only its subsets, the first two steps are needed only before the first run. The annotation package should be built by the AnnotationDbi (see paragraph 3.2.1). For more information on including the DAVID knowledgebase see apendix 8.3. Before we begin to present any results and their comparison with DAVID, the used parameters and nomenclature should be

described first.
The parameters that our implementation shares with DAVID and are adjustable from its web interface) had following values:

Initial Group Membership:	4 (the minimum gene number in a seeding group)
Final Group Membership:	4 (the minimum gene number in one final group)
Multiple Linkage Threshold:	0.50 (controls, how many members must two seeds share in order to be merged)
Similarity Threshold:	0.35 (the minimum kappa value to be considered biologically significant)
Similarity Term Overlap:	4 (the minimum number of annotation terms overlapped between two genes in order to be qualified for kappa calculation, not implemented in our source code)

The parameters that are related only to our implementation and that we estimated in order to fit DAVID's clustering results had following values:

relatednessUseMedoid:	`FALSE` (determines, whether count relations medoid-gene, when investigating the internal seed relatedness)
davidColsNumber:	103590 (total number of annotation terms)

DAVID uses 14 different annotation categories: Gene ontology (GO) biological process, GO molecular function, GO cellular component, KEGG pathways, BioCarta pathways, Swiss-Prot keywords, BBID pathways, SMART domains, NIH genetic association DB, UniProt sequence features, COG/KOG ontology, NCBI OMIM, InterPro domains, and PIR superfamily names. Except Biocarta pathways, all of them can be downloaded from the DAVID knowledgbase. GO biological process, GO molecular function, GO cellular component, KEGG pathways, UniProt sequence features and NCBI OMIM can be also downloaded by the AnnotationDbi directly from National Center for Biotechnology Information (NCBI). In addition, there are two strategies, how to work with gene ontologies: if the gene G is annotated by the term T from an ontology, the first strategy automatically annotates the gene by all ancestor nodes of the T, while the second annotates G just by term T. The data and strategies can be combined and we will denote them in the following text by these abbreviations:

D_{goall}:	Use only annotation data from DAVID knowledgebase, GO with all-ancestors strategy.
D_{go}:	Use annotation data from DAVID knowledgebase, GO with just-given-term strategy, GO data downloaded from NCBI.
N_{go}:	Use only six annotation categories from NCBI, GO with just-given-term strategy.

These are not all possible combinations but the remaining ones are not mentioned in this work.

5.2.1 Clustering with manually created distance matrix

This paragraph compares DAVID's and our clustering results with manually created distance matrix for random 20 genes. These experiments were run only in order to estimate the implementation parameters. Thanks to the distance matrix, different annotation data cannot influence the results, because they are simply not used. Before we begin to present the results, we denote each probe by one letter or digit:

probe ID	letter	probe ID	letter
179_AT	0	1463_AT	A
1167_S_AT	1	1774_AT	B
1124_AT	2	1258_S_AT	C
1602_AT	3	1242_AT	D
1007_S_AT	4	1750_AT	E
131_AT	5	136_AT	F
1276_G_AT	6	38642_AT	G
1267_AT	7	1244_AT	H
1829_AT	8	1461_AT	I
1432_S_AT	9	35687_AT	J

Table 5.1 Probe abbreviations

Using this notation, the gene list 1007_S_AT, 1432_S_AT, 136_AT, 38642_AT can be written as 49FG. Table 5.2 compares DAVID's and our results. Column "false negatives" contains number of genes, which were included by DAVID but excluded by our implementation whereas "false positives" is number of genes excluded by DAVID, but included by our implementation. Also note, the clusters were additionally sorted into alphabetical order.

Gene list	**DAVID's clusters**	**Our clusters**	**False negatives**	**False positives**	**Note**
0123456789 ABCDEFGHIJ	1. 3479FG 2. 5BDH	1. 3479F 2. 5BDH	1	0	All 20 probes
3479F	1. 3479	1. 3479	0	0	Discussed in 5.1.3
3479FG	1. 3479FG	1. 3479F	1	0	Discussed in 5.1.3
49FG	1. 49FG		4	0	Discussed in 5.1.3
124579ABD EGHIJ	1. 5BDH	1. 5BDH	0	0	
04569ABFHIJ			0	0	
04569ABFGHIJ	1. 49FG		4	0	38642_AT, see 5.1.3
0235679BC DFGIJ			0	0	

Table 5.2 Comparison table of DAVID's and our clusters

The table shows that the DAVID's and our results are similar, especially for longer lists. However in some cases (the 4-th and 7-th row) is our implementation fully wrong. It is caused by unrevealed influence of probe 38642_AT, as discussed in the second testcase in the paragraph 5.1.3.

5.2.2 Motol data

The Motol data were clustered using three different combinations of annotation data: D_{goall}, D_{go} and N_{go}. Because we cannot compare the results with DAVID, only the number of clusters and its lengths are mentioned. In addition, we included the gene names from several clusters in order to check out their biological relatedness. Note that the lengths of clusters are not exactly the same as the number of probes in given cluster, because one probe may be mapped to more genes and otherwise.

5.2.2.1 D_{goall}

Number of clusters: 7
Lengths of clusters: 364, 16922, 4, 17, 5, 6, 16
Names of genes from the cluster with 6 genes:

probe ID	Gene name
107070	solute carrier family 38, member 7
126840	solute carrier family 38, member 10
133249	solute carrier family 38, member 10
126979	solute carrier family 38, member 11
216971	solute carrier family 38, member 9
199400	solute carrier family 36 (proton/amino acid symporter), member 2
129845	solute carrier family 36 (proton/amino acid symporter), member 3

Table 5.3 Genes in sample cluster ($\mathbf{D_{goall}}$, Motol data)

5.2.2.2 D_{go}

Number of clusters: 25
Lengths of clusters: 16563, 9, 5, 20, 4, 18, 5, 6, 4, 13, 5, 4, 5, 26, 8, 4, 4, 7, 11, 11, 11, 4, 6, 4, 7
Names of genes from the second cluster (9 genes):

probe ID	Gene name
120590	aldolase A, fructose-bisphosphate
173434	aldolase B, fructose-bisphosphate
195743	aldolase C, fructose-bisphosphate
142235	enolase 2 (gamma, neuronal)
125158	fructose-1,6-bisphosphatase 1
212759	glucose phosphate isomerase
120739	lactate dehydrogenase A
126065	transaldolase 1
229010	transaldolase 1
235487	transaldolase 1
147521	triosephosphate isomerase 1

Table 5.4 Genes in sample cluster ($\mathbf{D_{go}}$, Motol data)

5.2.2.3 N_{go}

Number of clusters: 83
Lengths of clusters: 12906, 5, 18, 4, 10, 4, 14, 9, 4, 4, 29, 308, 5, 9, 26, 35, 5, 6, 6,

5, 8, 30, 5, 21, 6, 9, 8, 13, 5, 6, 12, 14, 5, 13, 5, 11, 5, 6, 8, 5, 4, 6, 6, 18, 4, 5, 12, 6, 4, 4, 5, 4, 331, 30, 25, 6, 14, 4, 8, 4, 74, 107, 45, 5, 4, 6, 5, 4, 4, 5, 5, 4, 6, 8, 5, 5, 7, 5, 4, 4, 4, 7, 9
Names of genes from the second cluster (5 genes):

probe ID	**Gene name**
203610	amiloride binding protein 1 (amine oxidase (copper-containing))
147381	amine oxidase, copper containing 2 (retina-specific)
206763	monoamine oxidase A
214773	monoamine oxidase B
198574	amine oxidase, copper containing 3 (vascular adhesion protein 1)

Table 5.5 Genes in sample cluster (N_{go}, Motol data)

5.2.2.4 Discussion

The algorithm is able to find a group of genes with similar biological function. Although we have presented just one (small) cluster for each result, genes in the other groups are also biologically related. On the other hand, the structure of clusters is not very good: there are several small clusters and one very large, that contains majority, or even almost all genes from list[17]. The "smallest large" cluster has appeared for N_{go} annotation data. They do not contain as many terms as D_{goall} and D_{go} have and it seems, that one way, how to break large clusters is to use smaller ammount of annotation data (this opinion is also supported by results on ALL/AML). However this approach is questionable: smaller amount of annotation data causes that algorithm may not find some important groups of biologically related genes, because it simply does not have any evidence for their relatedness.

5.2.3 ALL/AML

Our implementation of fuzzy clustering algorithm was also tested on ALL/AML data, a famous microarray assay published and analysed in [42]. They originate from 72 patients, 47 suffered from acute lymphoblastic leukemia (ALL) and 25 from acute myeloid leukemia (AML) and expression levels of 7,129 genes were measured for each sample.

[17] Motol data contains 18,279 annotated genes.

5.2.3.1 D_{goall}

Number of clusters: 11
Lengths of clusters: 5563, 20, 6, 6, 4, 10, 8, 5, 8, 5, 11

5.2.3.2 N_{go}

Number of clusters: 139
Lengths of clusters: 17, 17, 288, 15, 4, 9, 4, 24, 15, 276, 12, 5, 5, 43, 7, 4, 572, 4, 97, 5, 8, 74, 4, 8, 118, 95, 105, 11, 8, 5, 14, 12, 98, 34, 5, 12, 17, 34, 4, 4, 7, 8, 73, 6, 5, 53, 5, 14, 10, 18, 38, 4, 82, 73, 5, 26, 5, 11, 5, 21, 6, 8, 4, 18, 45, 7, 4, 10, 18, 22, 5, 5, 4, 6, 6, 9, 13, 5, 31, 5, 9, 10, 30, 4, 4, 17, 10, 9, 7, 5, 6, 20, 6, 5, 9, 4, 39, 7, 8, 23, 7, 4, 4, 4, 20, 4, 4, 4, 4, 5, 4, 4, 4, 4, 4, 5, 4, 14, 6, 9, 4, 5, 11, 4, 10, 4, 5, 5, 6, 4, 5, 6, 21, 8, 5, 4, 5, 6, 4

5.2.3.3 Discussion

The numbers of clusters for given annotation data confirm, the smaller amount of annotation data may cause higher number of clusters. However also should be mentioned many clusters (32) has minimum possible length (four genes).

5.2.3.4 Influence of kappa value on clustering results

The other way, how to decrease size of the largest cluster, is to change the similarity threshold (minimum kappa). Several properties of the clusters for various similarity thresholds and D_{goall} set of annotation data are summarized in tables 5.6 (ALL/AML data) and 5.7 (Motol data):

Kappa	**Number of clusters**	**Largest cluster length**	**Involved genes**	**Outliers**
0.35	11	5563	5618	1151
0.43	40	4991	5243	1886
0.50	85	3745	4514	2615
0.55	128	1681	3892	3237
0.60	138	962	3190	3939
0.65	142	661	2546	4583
0.70	113	497	1850	5279
0.75	85	315	1217	5912
0.80	65	108	754	6375

Table 5.6 Dependence of properties of clusters on similarity threshold (ALL/AML)

Kappa	Number of clusters	Largest cluster length	Involved genes	Outliers
0.35	7	16922	16990	1289
0.55	174	13067	14763	3516
0.60	250	7350	13251	5028
0.65	318	2928	11703	6576
0.70	355	1799	9645	8634

Table 5.7 Dependence of properties of clusters on similarity threshold (Motol)

5.2.3.5 Discussion

Tables 5.6 and 5.7 show that despite algorithm's design, there is a way, how to control number of created clusters. The dependence is nor linear neither smooth, but it has just one parameter and user need not change amount of annotation data. Unfortunatelly we still cannot say without experiment, how many clusters will appear. If we want to attain or at least get close to certain number of clusters, the script must be rerun iteratively with different similarity threshold. The second column of table shows, how length of the largest cluster decreases with increasing kappa. Together with that, number of genes involved in any cluster decreases. For example for ALL/AML data and kappa=0.65 only 2546 are clustered while 4583 are considered to be outliers. User must thus decide between uniformity of clusters lengths and number of clustered genes.

5.3 Classification

A typical classification workflow in this work is shown in figure 5.3:

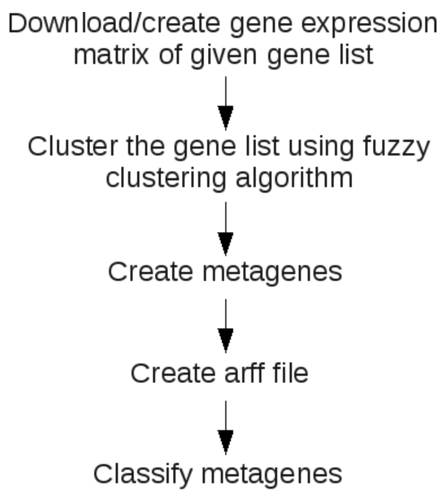

Download/create gene expression matrix of given gene list
Cluster the gene list using fuzzy clustering algorithm
Create metagenes
Create arff file
Classify metagenes

Figure 5.3 Classification work-flow diagram

The first step is dependent on the data, we want to classify. Sometimes they can be downloaded by GEO omnibus[18] and sometimes they must be built from text files. Metagenes are created in very simple way: metagene's expression for given sample is a median of expressions over all genes in the cluster. The fourth step is needed only because we use the Weka[19] for all classifiers.
We compared percent of correctly classified instances on original microarray data and on metagenes for two kinds of classifiers: the C4.5 and random forest. Additionally it is included the accuracy of the ZeroR, which simply predicts the majority class in the training data. Metagenes were created for D_{goall}, D_{go} and N_{go} clusters using kappa=0.35 and also for D_{goall} using different kappas (denoted as the index). The D_{goall} set is preferred, because it is the most consistent (it does not combine annotation terms from different years) and the widest available set. The original data were also classified in order to find out, how the clustering influences a quality of classifiers. All classifiers were validated by cross-validation (10 folds).

[18] http://www.ncbi.nlm.nih.gov/geo/

[19] Weka is a collection of machine learning algorithms for data mining tasks. It is a multiplatform software, because it is written in Java. Weka is being developed in the University of Waikato, New Zealand. The program and its usage is described in detail in [43].

	C4.5	**Random forest**	**ZeroR**
Motol original	**68.18**	**36.36**	**54.55**
Motol D_{go}	45.45	59.09	-//-
Motol N_{go}	77.27	45.45	-//-
Motol $D_{goall_{0.35}}$	31.81	45.45	-//-
Motol $D_{goall_{0.55}}$	50.00	45.45	-//-
Motol $D_{goall_{0.60}}$	40.91	36.36	-//-
Motol $D_{goall_{0.65}}$	36.36	45.45	-//-
Motol $D_{goall_{0.70}}$	22.73	40.91	-//-
ALL/AML original	**81.94**	**86.11**	**65.28**
ALL/AML N_{go}	84.72	80.56	-//-
ALL/AML $D_{goall_{0.35}}$	54.17	68.06	-//-
ALL/AML $D_{goall_{0.43}}$	70.83	75.00	-//-
ALL/AML $D_{goall_{0.50}}$	66.67	77.78	-//-
ALL/AML $D_{goall_{0.55}}$	86.11	88.89	-//-
ALL/AML $D_{goall_{0.60}}$	80.56	79.17	-//-
ALL/AML $D_{goall_{0.65}}$	72.22	84.72	-//-
ALL/AML $D_{goall_{0.70}}$	79.16	76.39	-//-
ALL/AML $D_{goall_{0.75}}$	72.22	76.39	-//-
ALL/AML $D_{goall_{0.80}}$	75.00	77.78	-//-

Table 5.8 Accuracy of classifiers on original and clustered data

5.3.1 Discussion

The first eight rows show the accuracy of the classifiers on the original and clustered Motol data. Some of them are even worse then classifier ZeroR, which classifies all samples into the largest class. However it is not very suprising as Motol data are known to be hard to be classified and it is also difficult to find any patterns in it (see [3]). From this reason the classification result on Motol N_{go} should't be considered as succesfull as they could appear just by chance. The results for ALL/AML are more interesting: they show, it is definitely useful to cluster genes into more groups. The accuracy of the classifiers on $D_{goall_{0.35}}$, $D_{goall_{0.43}}$ and $D_{goall_{0.50}}$ was very low.

The accuracies on remaining clustered data are comparable with results on original ALL/AML. The accuracy on N_{go} is also comparable, so in this case it was not important, the clusters were created without considering some functional categories.

6 Conclusions

In this work, we have described our implementation of DAVID fuzzy clustering algorithm, which is currently accessible via web intreface on the pages of the National Institute of Allergy and Infectious Diseases (http://david.abcc.ncifcrf.gov/). Our implementation has two main targets: allow researches to change the annotation data used for clustering and allow to analyze a gene list with theoretically unlimited length. However in the real world the length is still limited. Clustering of the Motol data for the largest set of annotation data took about 24 hours (CPU was AMD Turion 64 X2 1.6GHz, 4GB RAM) and because the clustering has quadratic complexity ($O(n^2)$), the twice longer list means the four times longer time of computation. The algorithm was rewritten into the R statistical language using its advanced capabilities like calling an external C code and working with databases. Our implementatrion will probably never have the same output as the DAVID has. The reason is simple: although DAVID offers its knowledgebase (set of annotation data) to be downloaded for free, these data are not the same as DAVID uses internally for clustering purposes. In addition, the knowledgebase was last updated more than year ago (beginning of 2008). The newest annotation data can be downloaded by R package called AnnotationDbi, but it allows to download only six categories from the 14, which uses DAVID. Despite that, our implementation is able to find biologically related groups of genes in a given list and it is well adjustable.
Two different microarray assays were used for testing: the Motol data and ALL/AML. Motol data is a collection of 22 gene expression profiles from a bladder tissue. 12 patients (samples) suffered from recurrent blader cancer and 10 were control samples. ALL/AML originate from 72 patients, 47 suffered from acute lymphoblastic leukemia (ALL) and 25 from acute myeloid leukemia (AML). On both datasets it was shown, that too low similarity threshold (minimum value of the kappa statistics, for which are two genes enough related) causes creation of one large cluster, that contains majority or even almost all genes from the list. This cluster can be broken by greater value of the similarity threshold or by using a less amount of annotation data. It was shown, that the greater similarity threshold reduces length of the largest cluster, but on the other hand, it increases the number of genes considered to be outliers. However sometimes it may be wanted feature, because the filtered gene list then contains only genes, which are well biologically related.
One purpose of the clusters is to serve as metagenes and consequently to replace the original dataset in further analysis. A metagene expression level for a given sample was compueted as a the median of the expressions over all genes in the cluster. The dimension of the original dataset is reduced in this way. Of course, the metagenes cannot fully replace the original dataset, but we believe, they may be useful as they

have much lower dimension than the original data and are created only from genes with strong biological relations.
Finally, we classified the original datasets and metagenes by two kinds of classifiers: C4.5 and random forest. As expected (in accordance with [3]), their accuracy on both clustered and original Motol data was low: some classifiers were even worse, than ZeroR, which classifies all samples into the largest class. However on clustered ALL/AML was shown, that for some values of the similarity threshold the classifiers may be at least as accurate as classifiers on the original data. However this work is mainly focused on clustering, so the classification was not explored deeply.
We would like to further test the algorithm on more datasets and try to explore, if there is a set of parameters, for which the clustering results are optimal for given purpose (e.g. the classifiers have in average the best accuracy on the resulting metagenes, as many clusters as possible appeared, etc.). This implementation will be used by IDA (Intelligent Data Analyses) research group (in CTU in Prague) for various research purposes.

7 References

[1] Baxevanis and Ouellette. *Bioinformatics: A Practical Guide to the Analysis of Genes and Proteins.* Wiley, 3rd Edition edition, 2005.

[2] *Bioconductor Online Documentation.* Available at: http://www.bioconductor.org/docs/pslides, 2009.

[3] Miloš Krejník. *Předzpracování a klasifikace genomických dat.* České vysoké učení technické v Praze, 2008.

[4] Wikipedia. *DNA.* Wikipedia, The Free Encyclopedia, available from http://en.wikipedia.org/wiki/DNA, 2009a.

[5] Wikipedia. *Gene.* Wikipedia, The Free Encyclopedia, available from http://en.wikipedia.org/wiki/Gene, 2009b.

[6] Wikipedia. *The genetic code.* Wikipedia, The Free Encyclopedia, available from http://en.wikipedia.org/wiki/Genetic_code, 2009c.

[7] American Chemical Society. *Artificial Genetics: New Type Of DNA Has 12 Chemical Letters Instead Of Usual 4.* ScienceDaily, 2009, March 23.

[8] National Center for Biotechnology Information. *Microarrays: chipping away at the mysteries of science and medicine.* Available from http://www.ncbi.nlm.nih.gov/About/primer/microarrays.html, 2007.

[9] Paul Muhlrad. *DNA microarray technology to identify genes controlling spermatogenesis.* Available from http://www.mcb.arizona.edu/wardlab/microarray.html, 2001.

[10] Virginie M Aris, Michael J Cody, Jeff Cheng, James J Dermody, Patricia Soteropoulos, Michael Recce and Peter P Tolias1. *Noise filtering and nonparametric analysis of microarray data underscores discriminating markers of oral, prostate, lung, ovarian and breast cancer.* BMC Bioinformatics, 5:185, 2004.

[11] Applied Biosystems. *Applied Biosystems Human Genome Survey Microarray V2.0.* Product Bulletin, 2005.

[12] Kanehisa M., Araki M., Goto S., Hattori M., Hirakawa M., Itoh M., Katayama T., Kawashima S., Okuda S., Tokimatsu T., Yamanishi Y.. *KEGG for linking genomes to life and the environment.* Nucleic Acids Res. 36, D480-D484, 2008.

[13] Kanehisa M., Goto S., Hattori M., Aoki-Kinoshita K.F., Itoh M., Kawashima S., Katayama T., Araki M., Hirakawa M.. *From genomics to chemical genomics: new developments in KEGG.* Nucleic Acids Res. 34, D354-357, 2006.

[14] Kanehisa M. and Goto S.. *KEGG: Kyoto Encyclopedia of Genes and Genomes.* Nucleic Acids Res. 28, 27-30, 2000.

[15] Kyoto Encyclopedia of Genes and Genomes. *KEGG PATHWAY Database.* http://www.genome.jp/kegg/pathway.html, 2009.

[16] The Gene Ontology Consortium. *Gene Ontology: tool for the unification of biology.* Nature Genet. 25: 25-29, 2000.

[17] The Gene Ontology Consortium. *The Gene Ontology.* http://www.geneontology.org/, 2009.

[18] Universal Protein Resource. *Uniprot.* http://www.uniprot.org/, 2009.

[19] Bairoch A, Bougueleret L, Altairac S, Amendolia V, Auchincloss A, Puy GA, Axelsen K, Baratin D, Blatter MC, Boeckmann B et al. *The Universal Protein Resource (UniProt).* Nucleic Acids Res, 36:D190–D195, 2008.

[20] Nomenclature Committee of the International Union of Biochemistry and Molecular Biology (NC-IUBMB). *Enzyme Nomenclature.* http://www.chem.qmul.ac.uk/iubmb/enzyme/, 2009.

[21] Prosite. *Database of protein domains, families and functional sites.* http://www.expasy.ch/prosite/, 2009.

[22] EMBL-EBI. *InterPro.* http://www.ebi.ac.uk/interpro/, 2009.

[23] Biocarta. *Biocarta patrhways.* http://www.biocarta.com/genes/index.asp, 2009.

[24] National Center for Biotechnology Information. *National Center for Biotechnology Information.* http://www.ncbi.nlm.nih.gov/, 2009.

[25] Hornik. *The R FAQ.* http://CRAN.r-project.org/doc/FAQ/R-FAQ.html, ISBN 3-900051-08-9, 2009.

[26] R Foundation. *The R Project for Statistical Computing.* http://www.r-project.org/, 2009.

[27] Roger D. Peng and Jan de Leeuw. *An Introduction to the .C Interface to R.* Available from: www.ats.ucla.edu/stat/R/library/interface.pdf, 2002.

[28] Charles Geyer. *Calling C and Fortran from R.* Available from: http://www.stat.umn.edu/~charlie/rc/.

[29] R Special Interest Group on Databases (R-SIG-DB). *DBI: R Database Interface.* R package version 0.2-4, 2009.

[30] Gentleman, Rossini, Dudoit and Hornik. *The Bioconductor FAQ.* Available from: http://www.bioconductor.org/docs/faq/, 2003.

[31] Herve Pages, Marc Carlson, Seth Falcon and Nianhua Li. *AnnotationDbi: Annotation Database Interface.* R package version 1.4.3, 2009.

[32] William T. Barry. *safe: Significance Analysis of Function and Expression.* R package version 1.8.0, 2008.

[33] Barry W.T., Nobel A.B. and Wright F.A.. *Significance Analysis of functional categories in gene expression studies: a structured permutation approach.* Bioinformatics, 2005.

[34] Ting-Yuan Liu, ChenWei Lin, Seth Falcon, Jianhua Zhang and James W. MacDonald. *KEGG: A data package containing annotation data for KEGG.* R package version 1.16.1, 2008.

[35] Adrian Alexa and Jorg Rahnenfuhrer. *topGO: Enrichment analysis for Gene Ontology.* R package version 1.2.1, 2006.

[36] Subramanian A, Tamayo P, Mootha VK, Mukherjee S, Ebert BL, Gillette MA, Paulovich A, Pomeroy SL, Golub TR, Lander ES, Mesirov JP. *Gene set enrichment analysis: a knowledge-based approach for interpreting genome-wide expression profiles.* Proc Natl Acad Sci USA ;102:15545–15550. doi: 10.1073/pnas.0506580102, 2005.

[37] Holger Schwender. *siggenes: SAM and Efron's empirical Bayes approaches.* R package version 1.10.1, 2007.

[38] Benjamin Georgi, Matthias Heinig, Stefan Roepcke, Sebastian Schmeier and Joern Toedling. *macat: MicroArray Chromosome Analysis Tool.* R package version 1.10.0, 2006.

[39] Huang DW, Sherman BT and Lempicki RA. *Systematic and integrative analysis of large gene lists using DAVID Bioinformatics Resources.* Nature Protoc.;4(1):44-57., 2009.

[40] Dennis G J , Sherman BT, Hosack DA, Yang J, Gao W, Lane HC, Lempicki RA. *Database for Annotation, Visualization, and Integrated Discovery.* Genome Biol;4(5):P3, 2003.

[41] Da Wei Huang, Brad T Sherman, Qina Tan, Jack R Collins, W Gregory Alvord, Jean Roayaei, Robert Stephens, Michael W Baseler, H Clifford Lane, Richard A Lempicki. *The DAVID Gene Functional Classification Tool: a novel biological module-centric algorithm to functionally analyze large gene lists.* Genome Biology 8:R183, 2007.

[42] Golub T.R., Slonim D.K., Tamayo P., Huard C., Gassenbeek M., Mesirov J.P., Coller H., Loh M.L., Downing J.R., Caligiuri M.A., Bloomfield C.D., Lander E.S. *Molecular classification of cancer: class discovery and class prediction by gene expression monitoring.* Science, 286(15):531–537, 1999.

[43] Ian H. Witten and Eibe Frank. *Data Mining: Practical machine learning tools and techniques.* Morgan Kaufmann, San Francisco, 2nd Edition edition, 2005.

8 Apendices

8.1 Appended CD

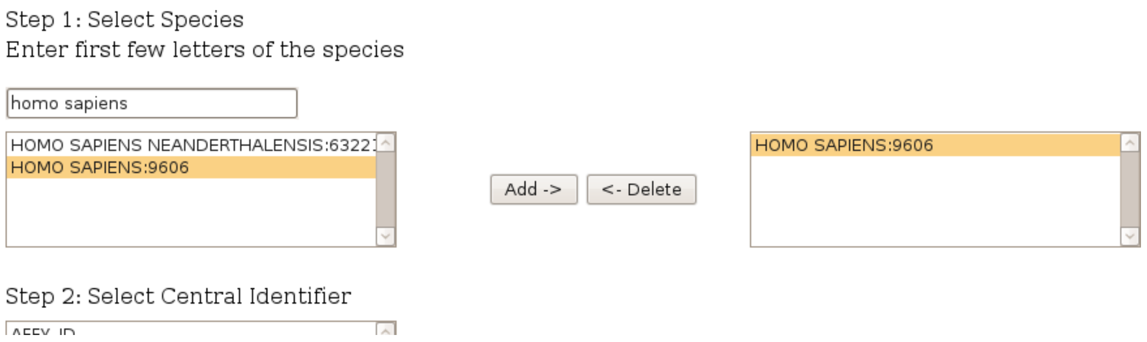
data	Contains sample gene lists, expression and distance matrices and all data used by example scripts.
doc	Contains this paper in pdf format.
examples	Sample scripts: • build_ann_package.R: Shows, how to build an annotation package. In this case, package for DAVID's demolist2 is built. • build_expressionset.R: Demonstrates creation of ExpressionSet object. It is an R class, that contains all information about microarray assay. • cluster_demolist2.R: Clusters all genes from DAVID's demolist2. • cluster_given_list.R: Clusters given gene list with 20 probes using external distance matrix. • clusters2arff.R: Creates metagenes from random expression matrix. • import_david.R: Briefly shows, how to integrate DAVID knowledgebase into an R annotation package. Important note: before you run any of these scripts, R working directory must be changed to directory, where the scripts are stored. Otherwise they will not work.
src	Source codes of fuzzy clustering algorithm and auxilliary scripts.

8.2 Used software

Bioconductor	Open source software project to provide tools for the analysis and comprehension of genomic data.
DAVID	The Database for Annotation, Visualization and Integrated Discovery, offers plenty of functions. We have used mainly Functional Classification Tool and Functional Annotation Table.
R	Interpreted language and environment for statistical computing and graphics. Except some small (but important) portions of code all scripts are written in this language.
Weka	Collection of machine learning algorithms for data mining tasks. All classifiers in this work were created in Weka.

8.3 Download of DAVID knowledgebase

If we have already built an R annotation package, we may to want include there also annotation data found at DAVID knowledgebase. The first step is the download. We need to register first, but it is for free and just a few textboxes are required to fill in. Registration page is located at http://david.abcc.ncifcrf.gov/knowledgebase/register.htm. After we are registered, go to http://david.abcc.ncifcrf.gov/knowledgebase/login.html and login using the email address, we have entered in registration. After that, the licence must be accepted and it is definitely useful to read it in order to avoid future misunderstandings. After clicking the "Accept" button we are redirected to download page. Type "homo sapiens" into the first textbox and then add "HOMO SAPIENS:9609" into the listbox rightwards (see figure 8.1).


Step 1: Select Species
Enter first few letters of the species

homo sapiens

HOMO SAPIENS NEANDERTHALENSIS:6322
HOMO SAPIENS:9606

HOMO SAPIENS:9606

Step 2: Select Central Identifier

AFFY_ID

Figure 8.1 DAVID knowledgebase download page

Then select Central Identifier. It should be ENTREZ_GENE_ID, because R annotation packages are entrez-centric and we can also use prepared script in order to merge the data (see /examples/import_david.R on appended CD). Finally select wanted annotation categories, click to "Submit" and after a while we will receive an email shortly with details on how to retrieve the requested data. Download them and unzip. Linux and *nix users are nearly done in this point. They just load /src/importDAVID.R into an R session and then type

```
> importDAVID("packagename","/directory/where/data/were/unzipped")
```

and the script automatically change permissions to package database[20] and insert the data. Windows users must probably change database permissions manually, but the script was not tested on Windows.
The script creates several new database tables using following query:

```
CREATE TABLE IF NOT EXISTS prefixTable
              (_id INT NOT NULL,
               annTerm CHAR(255) NULL,
               FOREIGN KEY (_id) REFERENCES genes (_id))
```

The value of "prefix" depends on a category of annotation data.

8.4 Create arff file with metagenes from gene clusters

This is actually not very difficult task: just load /src/clusters2arff.R into the R session and call

```
> setwd("/directory/with/cluster2arff/file")
> cluster2arff(clusters,"packagename",expressionSet,"arff_output_filename",
+ "class0")
```

with proper arguments (see example script /examples/clusters2arff.R for more information). However newly created arff file cannot be immediately loaded by Weka and it needs one little change:
Open the file in your favourite text editor and find row beginning with

```
@attribute class
```

This row should be replaced by

```
@attribute class {0,1}
```

After this change, the file can be analysed by Weka.

[20] SQLite databases are stored in single files.