13

Biological Data Modelling and Scripting in R

Srinivasan Ramachandran et al.*
G.N. Ramachandran Knowledge Centre for Genome Informatics,
Institute of Genomics and Integrative Biology, Delhi,
India

1. Introduction

In this age of Systems and Integrative Biology, development of high throughput genome sequencing techniques and other large-scale experimental methods, are generating large amount of biological data. Bioinformatics enables us to generate added value to these datasets in the form of annotation, classification and pattern extraction. These developments demand adequate storage and organization for further analysis.

In order to unravel the trends and patterns present in such diverse data sets, computational platforms with capability for carrying out integrative analysis are required for rapid analysis. R language platform is an example of one such platform allowing integrated rapid analysis process. The R is a High-level interpreted language suitable for developing new computational methods (R Development Core Team. 2010). Computational Biologists use R extensively because of the availability of numerous functions and packages including the well-known Bioconductor package (Gentleman et al., 2004). The rich inbuilt functions and the facility to write functions as well as object oriented programming facilities enable development of new packages for rapid analysis.

2. R platform

R is a programming language integrated with an R environment, facilitating easy and rapid data analysis with the help of its integrated suite of software facilities. Several computational biology packages have been developed in R language. Developing computational packages in R provides advantage as to carry out the analysis locally and also build further tools and scripts. Thus both new applications and extension of existing applications can be achieved. R helps accomplishment of complex tasks using simple scripts with the help of inbuilt suit of operators aiding in calculations. Also R environment provides graphical facilities for data analysis and display. Another major advantage of preparing datasets and computational biology tools in R is that a large set of statistical and mathematical tools can be applied on the datasets for analysis. R being an open source controlled by GNU General Public License allows future developments and customizations

*Rupanjali Chaudhuri, Srikant Prasad Verma, Ab Rauf Shah, Chaitali Paul, Shreya Chakraborty, Bhanwar Lal Puniya and Rahul Shubhra Mandal
G.N. Ramachandran Knowledge Centre for Genome Informatics, Institute of Genomics and Integrative Biology, Delhi, India

more widely. R is maintained by a core group of experts, thus ensuring its availability for long life. R in its repository also has a number of packages useful in various fields of biology. These packages help solve biological problems in well-structured manner saving time and money.

3. Data modeling for R

Data modeling for R involves identification of the datasets required for the corresponding problem undertaken. The data in the datasets needs to be structured into relevant rows and columns. For each field or column only one data type is allowed either character or numeric data type. Thereafter standardization or pre-processing of the data in datasets needs to be done. This involves checking the data for any inconsistencies- e.g., removal of blank cells by replacing with "Not known" or "None", checking header names for unwanted symbols like ?@$%*^ #/, checking columns for single data-type etc. The datasets may be then made into R object. Thus data modeling for R plays an important role to make data easily and properly read and operated with scripts in R platform. The data type in each column must conform to same format for all cells in that column.

4. S4 object oriented programming

S4 is the 4th version of S. The major development of S4 over S3 is the integration of functions, which allows considering S as an object oriented language. The object system in S4 provides a rich way of defining classes, handling inheritance, setting generic methods, validity checking and multiple dispatches. This allows development of easy to operate packages for rapid data handling and organized structured framework.

4.1 Setting class and reading data into S4 objects

Classes with specific representations are created in S4. Thereafter new object belonging to the set class may be created. Generic functions may also be made using object of the class:

1. setClass() is used to set the class of a data
2. new()is used to create objects of the class set
3. setGeneric() helps define generics
4. setMethods() is used to set methods

5. Decision tree

A decision tree (Maimon et al., 2005) is a tree like graph that a decision maker can create to help select the best amongst several alternative courses of action. Biological problems can be solved with help of well-structured and optimized algorithms. These algorithms can be represented in the form of decision trees to get better and clear understanding of the algorithm process followed to solve the biological problem.

6. Bioinformatics tools to retrieve biological data

Bioinformatics in its repository has a large number of tools developed to address diverse biological questions. These include investigating relationship between protein structure and function, immune response, development of potential vaccine candidates, modeling pathways, discovery of drug targets and drugs.

6.1 Immunoinformatics data

The immunoinformatics branch of bioinformatics deals with applying bioinformatics principles and tools to the molecular activities of the immune system. Immunoinformatics provides databases and predictive tools, useful to fetch data on cells of immune system. This data is termed immunological data and can be broadly split into epitope data and allergen data. This data is useful for aiding in vaccine discovery, referred to as computer aided vaccine design. An important aim here is antigen identification or identification of epitopes capable of eliciting immune response. There are various immunoinformatics databases available for aiding this process (Chaudhuri et al., 2008; Chaudhuri et al., 2011; Vivona et al., 2008).

An epitope, also known, as 'antigenic determinant' is a surface localized part of antigen capable of eliciting an immune response. A B-cell epitope is region of the antigen recognized by soluble or membrane bound antibodies. B-cell epitopes are further classified as either linear or discontinuous epitopes. Linear epitope is a single continuous stretch of amino acids within a protein sequence, whereas epitopes whose residues are distantly placed in the sequence but are brought together by physico-chemical folding are termed as discontinuous epitopes.

T cell epitope is a short region presented on the surface of an antigen-presenting cell, where they are bound to MHC molecules. These epitopes can be characterized into two types based on their recognition by either MHC Class I molecule or Class II molecule.

Epitope prediction tools form the backbone of immunoinformatics. The main aim of these tools is to aid in reliable epitope identification. Various sophisticated T cell epitope prediction tools have been developed which help successful epitope prediction. Some of these algorithms are based on artificial neural networks and weight matrices such as NetMHC (Lundegaard et al., 2008), predictive IC(50) values IEDB-ARB method (Bui et al., 2005; Zhang et al., 2008), predicted half-time of dissociation Bimas (Parker et al., 1994), quantitative matrices ProPred (Singh et al., 2001). Reliable and accurate B-cell epitope prediction is still in development although we have some tools such as ABCpred (Saha et al., 2006) and BcePred (Saha et al., 2007). These tools help build the epitope data from protein sequences.

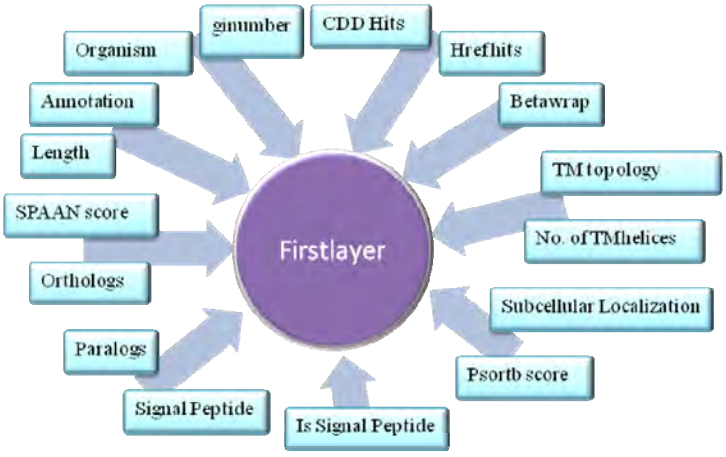
Allergen identification holds major importance in vaccine discovery problem, as it is desirable that a candidate vaccine is non-allergic. Allergens are substances (proteins, carbohydrates, particles, pollengrains etc.) to which the body mounts a hypersensitive immune response typically of Type I.

Various tools of immunoinformatics have been developed with aim to predict allergenic proteins. AlgPred (Saha et al., 2006) allows prediction of allergens through either singly or in combination of support vector machine, motif-based method, and searching the database of known IgE epitopes. Allermatch (Fiers et al., 2004) performs BLAST search against allergen representative peptides using a sliding window approach. The data fetched constitute allergen data. The building of Dataclasses with their representations is described in Figures 1-4.

6.1.1 Identification of potential immunogens useful as vaccine candidates

Immunogen is a substance capable of eliciting an immune response. It possesses epitopes, which binds to the B cells or T cells to elicit the response. To identify protein immunogens useful as vaccine candidates, bioinformatics approach may be undertaken. There are various B-cell and T-cell epitope prediction tools available as mentioned in the previous section. These algorithms provide prediction of the epitopes present in the submitted protein sequence. Each prediction comes with associated score representing the confidence of

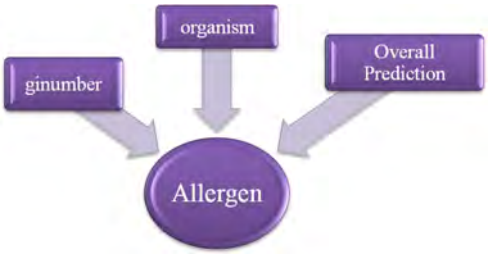
prediction. A cutoff score can be set to select the high scoring epitopes and subsequently proteins can be identified with high scoring epitopes. Thus the filtered orfids of proteins with high scoring B-cell and T-cell epitopes can be selected. The individual results of orfids may be analyzed by using the 'intersect' operator of R to get the final list of orfids representing the proteins meeting conditions of multiple features. It is desirable for the candidate vaccine to be non-allergic. Allergen data for the proteins may be fetched using allergen prediction immunoinformatics tools to obtain list of non-allergic proteins. Thus list of non-allergen proteins with high scoring B-cell and T-cell epitopes may be obtained. B-cell and T-cell data have been captured as **Secondlayer** data. As an example from the **Firstlayer** data certain sub problems to target a potential adhesin vaccine candidate can be stated as- the protein should be an adhesin, the protein should not be intracellularly located, it should not have similarity to human reference proteins, it should not have more than one transmembrane helix thereby facilitating proper cloning and expression. The set of proteins fulfilling all the **Firstlayer** conditions can be intersected with the set of non-allergen proteins. This whole process is depicted as decision tree (Figure 5). Similarly the decision tree describing the steps for obtaining proteins with high scoring B-cell and T-cell epitopes is shown in Figure 6.




```
setClass("FirstLayer", representation(ginumber = "numeric", annot =
"character", length = "numeric", spaanscore = "numeric", paralogs =
"character", omcl = "character", signalp = "numeric", is_signalp =
"character", psortbscore = "numeric", subcelllocal = "character",
tmhelices = "numeric", topotmhelix = "character", betawrap =
"character", Hrefhits = "character", cddhits = "character"))
```

```
readdata.firstlayer<-
function(xz){xa<-
readLines(con = xz);
tempy<- NULL;for (i in
seq ( along =
xa)){tempx<-
unlist(strsplit(xa[i],"\t"))
;tempy<- c(tempy,
new("FirstLayer",
ginumber =
as.numeric(tempx[1]),
annot = tempx[2], length
= as.numeric(tempx[3]),
spaanscore =
as.numeric(tempx[4]),
paralogs = tempx[5],
omcl = tempx[6],
signalp =
as.numeric(tempx[7]),
is_signalp = tempx[8],
psortbscore =
as.numeric(tempx[9]),
subcelllocal =
tempx[10],  tmhelices =
as.numeric(tempx[11]),
topotmhelix =
tempx[12], betawrap =
tempx[13], Hrefhits =
tempx[14], cddhits =
tempx[15]  )) };
return(tempy)}
```

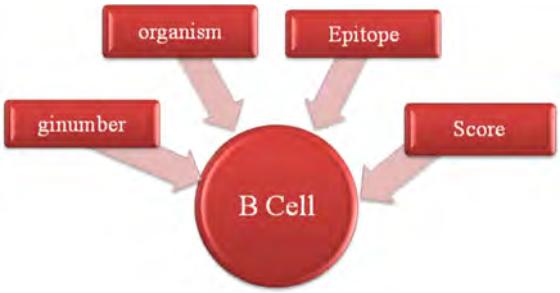
Fig. 1. Representation of S4 Class "FirstLayer" and the R scripts to accomplish the construction.




```
setClass("Algpred", representation(ginumber =
"numeric" , organism="character", ovpr=
"character",IGEPred="character",IgEepitope="cha
racter",Seqmatched="character",position="numer
ic",PID="numeric", MASTRESULT="character",
SVMPRED="character",SVMScore="numeric",SV
MThold="numeric"
,SVMPPV="character",SVMNPV="character",SV
MDipepPRED="character",
SVMDipepScore="numeric",SVMDipepThold="n
umeric",SVMDipepPPV="character",SVMDipep
NPV="character",BLASTPred="character",HitAR
Ps="character")); setClass("Allermatch",
representation(ginumber = "numeric" ,
organism="character", prediction
="character",hit_no= "numeric", db ="character",
allermatch_id ="character",best_nit_index
="numeric",no_hits_ident_gt35
="numeric",perc_hits_gt35 ="numeric",
perc_ident ="numeric", seq_len_fasta_aligned=
"numeric",external_link ="character", link_db
="character",genus_name ="character",
spc_name="character"))
```

```
readdata.algpred<- function(xz){xa<-
readLines(con = xz); tempy<- NULL;for (i in seq
( along = xa)){tempx<-
unlist(strsplit(xa[i],"\t"));tempy<- c(tempy,
new("Algpred", ginumber =
as.numeric(tempx[1]), organism = tempx[2],
ovpr= tempx[3] ,IGEPred=tempx[4]
,IgEepitope=tempx[5] ,Seqmatched=
tempx[6],position= as.numeric(tempx[7]) ,PID=
as.numeric(tempx[8]), MASTRESULT=tempx[9],
SVMPRED=tempx[10], SVMScore=
as.numeric(tempx[11]),SVMThold=
as.numeric(tempx[12]) ,SVMPPV=
tempx[13],SVMNPV=
tempx[14],SVMDipepPRED= tempx[15],
SVMDipepScore=
as.numeric(tempx[16]),SVMDipepThold=
as.numeric(tempx[17]),SVMDipepPPV=
tempx[18],SVMDipepNPV=
tempx[19],BLASTPred= tempx[20],HitARPs=
tempx[21]))};return(tempy)}
readdata.allermatch<- function(xz){xa<-
readLines(con = xz); tempy<- NULL;for (i in seq
( along = xa)){tempx<-
unlist(strsplit(xa[i],"\t"));tempy<- c(tempy,
new("Allermatch",ginumber =
as.numeric(tempx[1]), organism = tempx[2],
prediction= tempx[3], hit_no =
as.numeric(tempx[4]),db= tempx[5],
allermatch_id= tempx[6], best_nit_index=
as.numeric(tempx[7]), no_hits_ident_gt35=
as.numeric(tempx[8]),perc_hits_gt35=
as.numeric(tempx[9]),perc_ident=
as.numeric(tempx[10]), seq_len_fasta_aligned=
as.numeric(tempx[11]), external_link=
tempx[12], link_db= tempx[13], genus_name=
tempx[14], spc_name=
tempx[15]))};return(tempy)}
```

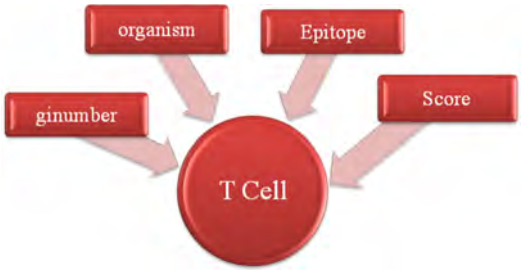
Fig. 2. General representation of S4 Class for Allergen data. The script for reading the data in is given for AlgPred. Similarly the data can be read for Allermatch class with appropriate data representation.




```
setClass("Bcepred", representation(ginumber =
"numeric",organism= "character", property= "character",
sequence= "character", length=
"numeric"));setClass("ABCpred", representation(ginumber
= "numeric" , organism="character", rank="numeric" ,
sequence="character", position="numeric",
score="numeric"))
```

```
readdata.bcepred<- function(xz){xa<-
readLines(con = xz); tempy<- NULL;for (i
in seq ( along = xa)){tempx<-
unlist(strsplit(xa[i],"\t"));tempy<-
c(tempy, new("Bcepred", ginumber =
as.numeric(tempx[1]), organism =
tempx[2], property = tempx[3], sequence =
tempx[4], length =
as.numeric(tempx[5])))}; return(tempy)}
readdata.abcpred<- function(xz){xa<-
readLines(con = xz); tempy<- NULL;for
(i in seq ( along = xa)){tempx<-
unlist(strsplit(xa[i],"\t"));tempy<-
c(tempy, new("ABCpred", ginumber =
as.numeric(tempx[1]), organism =
tempx[2], rank = as.numeric(tempx[3]),
sequence = tempx[4], position =
as.numeric(tempx[5]), score=
as.numeric(tempx[6])))};return(tempy)}
```

Fig. 3. General representation of S4 Class for B Cell epitope data along with R scripts.




```
setClass("Propred", representation(ginumber =
"numeric",organism= "character", Allele=
"character", Rank= "numeric", Sequence=
"character", Position= "numeric", Score= "numeric"))
```

```
readdata.propred <-function(xz){xa<-
readLines(con = xz); tempy<- NULL;for (i in
seq ( along = xa)){tempx<-
unlist(strsplit(xa[i],"\t"));tempy<- c(tempy,
new("Propred", ginumber =
as.numeric(tempx[1]), organism =
tempx[2],Allele= tempx[3], Rank=
as.numeric(tempx[4]), Sequence= tempx[5],
Position= as.numeric(tempx[6]), Score=
as.numeric(tempx[7])))};return(tempy)}
```

Fig. 4. General representation of S4 Class for T Cell epitope data.

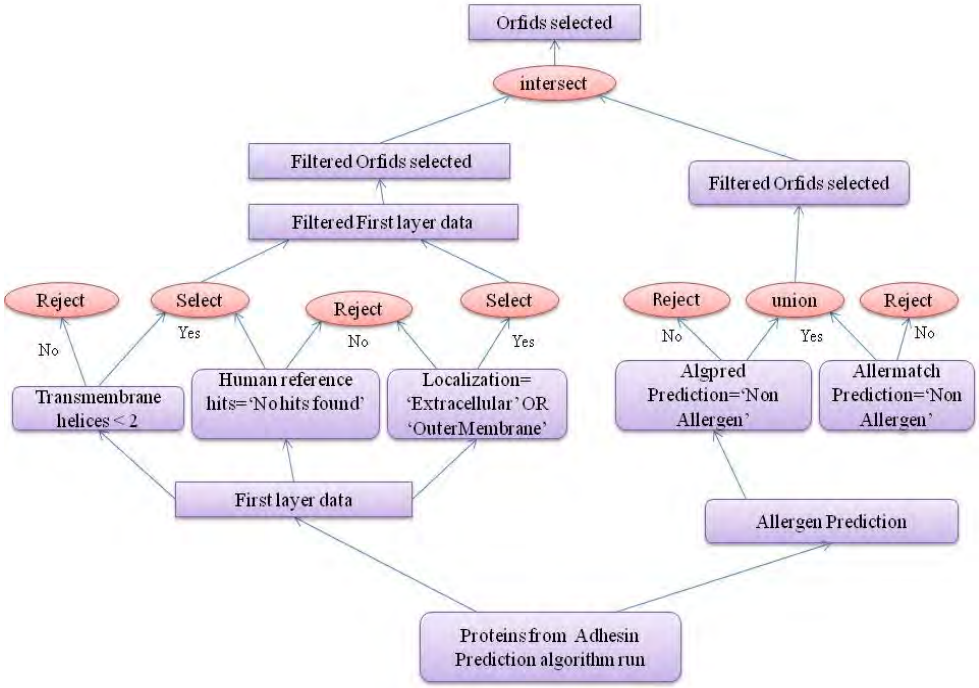


Fig. 5. Decision tree to identify non-allergen proteins fulfilling all first layer conditions. The R scripts are shown in the following two boxes.

```
S4 Methods
setGeneric("getfl_filtered",function(object)
standardGeneric("getfl_filtered"));setMethod("getfl_filtered","FirstLayer",function(obj
ect){if ((object@tmhelices < 2) && (object@Hrefhits== "No Hits found") &&
((object@subcelllocal == "Extracellular") | | (object@subcelllocal ==
"OuterMembrane"))) {return (object@ginumber)}else{return(FALSE)}})
setGeneric("nonallergen_algpred",function(object)
standardGeneric("nonallergen_algpred"));setMethod("nonallergen_algpred","Algpre
d",function(object){if
( object@ovpr == "Non Allergen") {return (object@ginumber)}else{return(FALSE)}})
setGeneric("nonallergen_allermatch",function(object)
standardGeneric("nonallergen_allermatch"));setMethod("nonallergen_allermatch","Al
lermatch",function(object){if
( object@prediction == "Non Allergen") {return
(object@ginumber)}else{return(FALSE)}})
```

R Scripts
res1<- sapply(ecalgpred,nonallergen_algpred); res2<-
sapply(ecallermatch,nonallergen_allermatch)
resA<- union(res1,res2); resB <- sapply(ecflnew,getfl_filtered); resC<-
intersect(resA,resB)

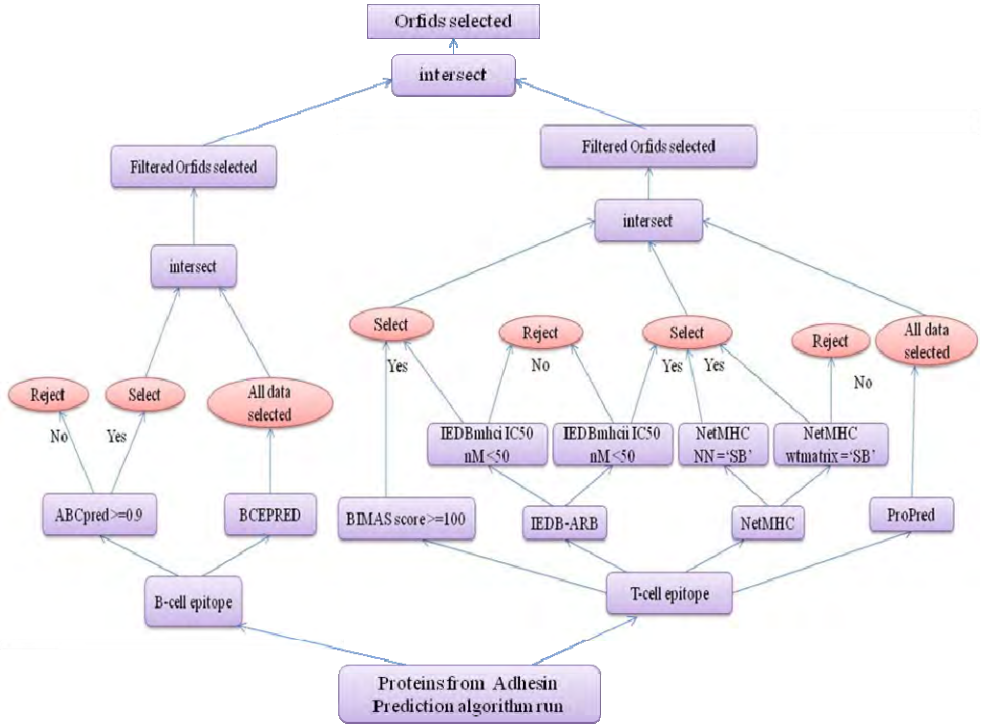


Fig. 6. Decision tree to identify high scoring B cell and T cell epitopes. The R scripts follow in the next two boxes.

```
S4 Methods
setGeneric("getgibce", function(object)
standardGeneric("getgibce"));setMethod("getgibce","Bcepred",function(object){
object@ginumber})
setGeneric("getgi_abcepitopes", function(object,x)
standardGeneric("get_abc_epi_gi"));setMethod("get_abc_epi_gi","ABCpred",fu
nction(object,x){if ( object@score >= x) {return
(object@ginumber)}else{return(FALSE)}})
setGeneric("getgipropred", function(object)
standardGeneric("getgipropred"));setMethod("getgipropred","Propred",functio
n(object){object@ginumber})
setGeneric("getgi_bimasepitopes",function(object,x)
standardGeneric("getgi_bimasepitopes"));setMethod("getgi_bimasepitopes","Bi
mas",function(object,x){if ( object@Score >= x) {return
(object@ginumber)}else{return(FALSE)}})
setGeneric("getgi_NetMHCNNepitopes",function(object)
standardGeneric("getgi_NetMHCNNepitopes"));setMethod("getgi_NetMHCN
Nepitopes","NetMHCneuralnet",function(object){if (object@Bind_level == "SB")
{return (object@ginumber)}else{return(FALSE)}})
setGeneric("getgi_NetMHCwtepitopes",function(object)
standardGeneric("getgi_NetMHCwtepitopes
"));setMethod("getgi_NetMHCwtepitopes
","NetMHCwtmatrix",function(object){if (object@Bind_level == "SB") {return
(object@ginumber)}else{return(FALSE)}})
setGeneric("getgi_iedbmhciepitopes", function(object,x)
standardGeneric("getgi_iedbmhciepitopes
"));setMethod("getgi_iedbmhciepitopes ","IEDB_mhci",function(object,x){if (
object@IC50 < x) {return (object@ginumber)}else{return(FALSE)}})
setGeneric("get_iedb_mhciiepi_gi", function(object,x)
standardGeneric("get_iedb_mhciiepi_gi"));setMethod("get_iedb_mhciiepi_gi","I
EDB_mhcii",function(object,x){if ( object@IC50 < x) {return
(object@ginumber)}else{return(FALSE)}})
```

```
R Scripts
resabc<- sapply(eclabc,getgi_abcepitopes,0.9); nrresABC<- union(resabc,resabc);
resbce<- sapply(eclbce,getgibce); nrresBCE <- union(resbce,resbce); resfl1<-
intersect(nrresABC,nrresBCE); resbimas <-
sapply(ecbimas,getgi_bimasepitopes,100); nrresBIMAS <-
union(resbimas,resbimas); resiedb<-
sapply(ec_iedb_mhci,getgi_iedbmhciepitopes,50); resiedbmhc1 <-
sapply(ec_iedb_mhcii, getgi_iedbmhciiepitopes, 50); nrresIEDBMHC2 <-
union(resiedbmhc1,resiedbmhc1); resnetmhcnn<-
sapply(ecNetMHCneuralnet,getgi_NetMHCNNepitopes); nrresNETMHCNN <-
union(resnetmhcnn,resnetmhcnn); resnetmhcwtmat <-
sapply(ecNetMHCwtmatrix,getgi_NetMHCwtepitopes); nrresNETMHCWTMAT
<- union(resnetmhcwtmat,resnetmhcwtmat); respropred<-
sapply(ecpropred,getgipropred); nrresPROPRED<-
union(respropred,respropred); nr1<- intersect(nrresBIMAS,nrresIEDB); nr2<-
intersect(nr1,nrresIEDBMHC2); nr3<- intersect(nr2,nrresNETMHCNN); nr4<-
intersect(nr3,nrresNETMHCWTMAT); nr5<- intersect(nr4,nrresPROPRED);
selectedgis<- intersect(resfl1,nr5)
finalgis<- intersect(selectedgis,resC)
```

6.2 Systems biology data

Systems biology deals with a system-level understanding of biological systems. A system can be defined by a set of interacting entities, which are linked to each other by direct and indirect interactions. A biological system is a very complex network, which cannot be described by reductionist's approach because it gives us a limited knowledge of a particular gene or protein that is insufficient to understand the complex behavior of a biological network. There is a need to integrate all the knowledge and comprehend new networks, which provide the overall picture of a system. These inferred networks can be used for further computational analysis and if found promising, can be validated through experiments. System level understanding requires the integration of experimental and computational biology. Modeling is the best method to represent a pathway and is the easiest way to understand a complex network. A network is modeled as a graph, which is the formal mathematical representation of the network and consists of nodes and edges. The network can be shown diagrammatically by using *classical graph theory*. All type of pathways (e.g. Gene regulatory network, signal transduction and metabolic pathways) can be modeled using various modeling techniques. A modeler uses two types of approaches- Data driven pathway modeling and Knowledge based pathway modeling, depending on the presence/absence of sufficient literature (Viswanathan et al., 2008). If the knowledge is limited, data driven pathway modeling becomes the best choice. These modeling techniques are also known as qualitative (Data driven) and quantitative (Knowledge driven) modeling approaches. Data driven pathway modeling requires the DNA microarray data set. For example, the Gene Regulatory network (GRN) can be inferred by using logical networks like Boolean networks, probabilistic Boolean network and dynamic Bayesian networks (Li et al., 2007).

A quantitative model describes a system with a set of mathematical equations. Recently, many software tools have been developed for quantitative modeling of biological systems. We know that all physiochemical reactions follow a physical or chemical principle. For example a given enzyme catalysis reaction may follow the Michaelis Menten kinetics (Nelson et al, 2000). Thus, every reaction in kinetic model is represented in kinetic equation, which is then solved by the ordinary differential equation. In other words, a model is represented as a system of ODEs (Ordinary Differential Equations) for each of the reactions involved in the pathway (Tyson et al., 2001). If kinetic parameters are available, ODE based modeling becomes the best tool to understand dynamics of network.
There are variety of bioinformatics tools available for modeling systems in many platforms. (Table 1 and Table 2)

Task	Tools	Web address
Model construction	CellDesigner Jarnac Jdesigner Gepasi	http://www.celldesigner.org/ http://sys-bio.org/ http://sys-bio.org/ http://www.gepasi.org/
Simulation	CellDesigner COPASI Gepasi SBaddon (MatLab tool)	http://www.celldesigner.org/ http://www.copasi.org/ http://www.gepasi.org/ http://www.mathworks.com/
Model Analysis	MatLab, R- environment	http://www.mathworks.com/ http://www.r-project.org/

Table 1. Bioinformatics tools for systems modeling in different platforms.

Package Name	Application
BoolNet	Generation, reconstruction, simulation and analysis of synchronous, asynchronous, and probabilistic Boolean networks
odesolve	Solver for ordinary differential equations
lpSolve	Interface to solve linear/integer programs
nlme	Linear and non-linear mixed effect model
SBML-R	SBML are R interface analysis tool

Table 2. Tools for systems modeling in R platform

6.2.1 Examining the expression pattern of genes in clinical strains, an example

This process is initiated by first collecting the microarray data from public repository. Next data normalization needs to be done. Log (base=10) transformed data can be used to normalize by using classical Z-score transformation method (Cheadle et al., 2003). Z-score reflects the relative expression condition of the genes. On the basis of z-score values we can categorize genes in many categories like highly expressed, moderately expressed and genes with low expression. We can also filter those genes having the z-score values higher than given cutoff in all samples or strains. The consistency of expression across the different samples or strains can be explained using Heatmap. R scripts may be used to obtain genes having z-score above 1 which would provide genes which are highly expressed. Heatmap

can be generated by using the R scripts. These heatmaps are false color image and very helpfull for visual comparison of different datasets. Dendrogram can be added on rows and columns by defining the heatmap arguments. The function Heatmap is provided by Bioconductor (Gentleman et al, 2004).

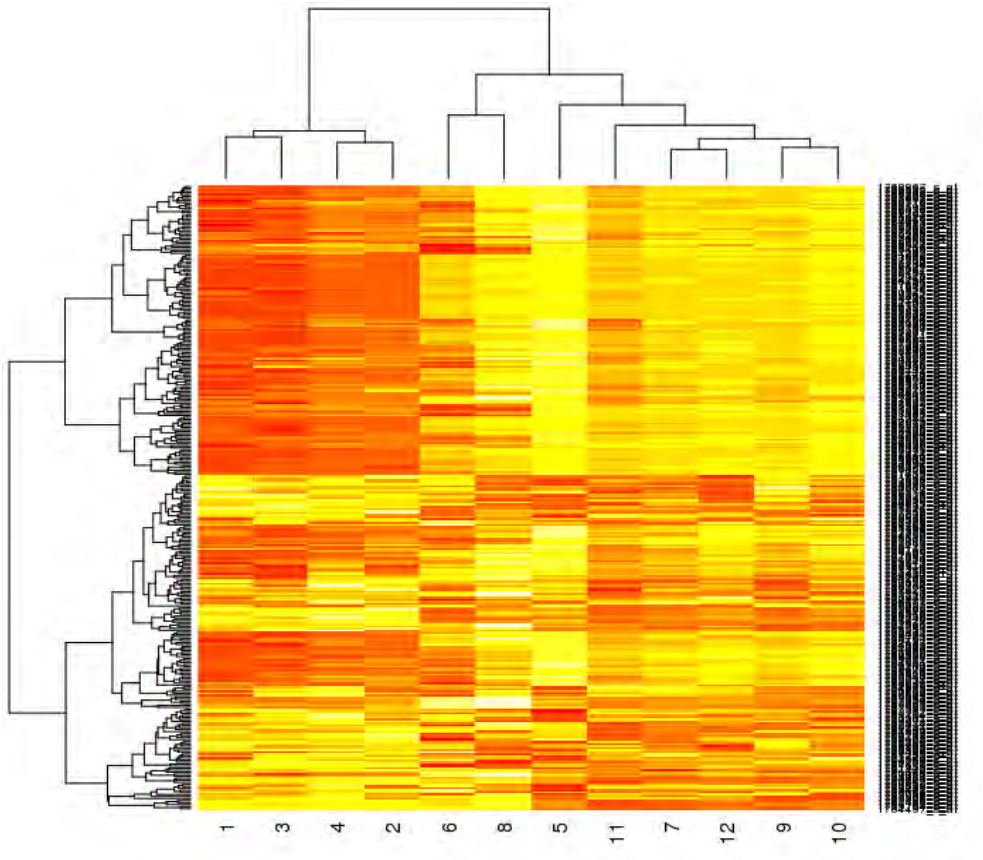


Fig. 7. Heat map of all probesets with z-score greater than 1.0 in all 12 samples. Red – Lower limit, Yellow - Upper limit gene expression Zscores. The sample ids are labelled below.

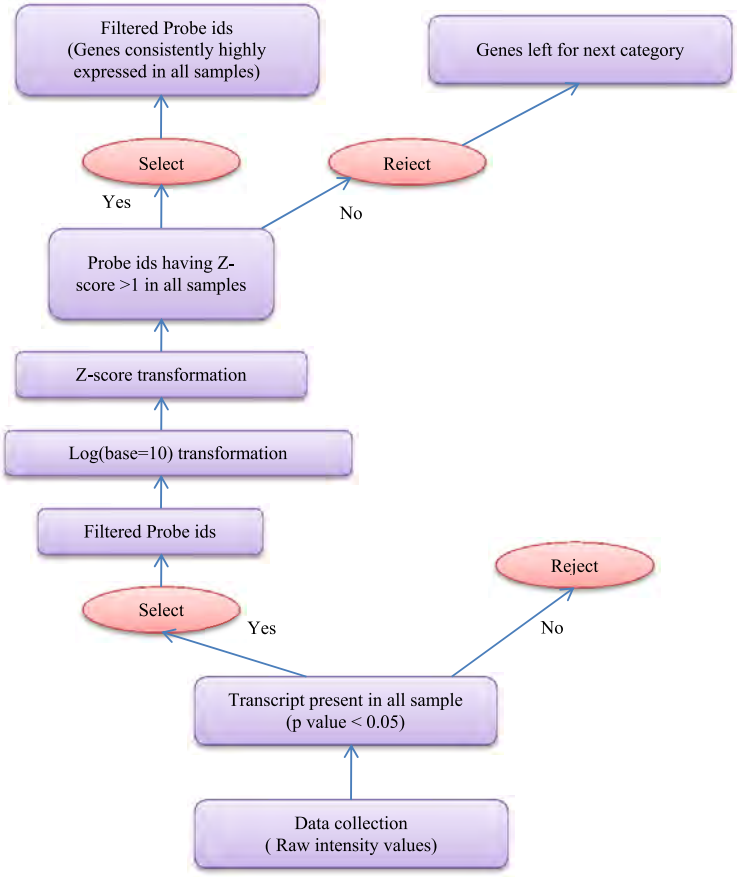


Fig. 8. Decision tree to identify highly expressed genes in clinical strains.


```
setClass("ZscoreEcoli",representation(probe_id = "character", expressionmat = "matrix"))
```

```
readdata.ZscoreEcoliExp<- function(xz){xa<-readLines(con = xz); tempy<- NULL;for (i in seq ( along = xa)){tempx<- unlist(strsplit(xa[i],"\t"));tempy<-c(tempy, new("ZscoreEcoli", probe_id = tempx[1], expressionmat = matrix(c(as.numeric(tempx[2]),as.numeric(tempx[3]),as.numeric(tempx[4]),as.numeric(tempx[5]),as.numeric(tempx[6]),as.numeric(tempx[7]),as.numeric(tempx[8]),as.numeric(tempx[9]),as.numeric(tempx[10]),as.numeric(tempx[11]),as.numeric(tempx[12]),as.numeric(tempx[13])), nrow=1,ncol=12) )) }; return(tempy)}
[11]),as.numeric(tempx[12]),as.numeric(tempx[13])), nrow=1,ncol=12) )) }; return(tempy)}
```

Fig. 9. Representation of S4 Class "ZscoreEcoli".

```
S4 Methods
setGeneric("getmatrix", function(object,x)
standardGeneric("getmatrix"));setMethod("getmatrix","ZscoreEcoli",function(object,x
){tempmat<- object@expressionmat; tempnew<- NULL; if ( ( tempmat[1,1] > x) &&
(tempmat[1,2] > x) && (tempmat[1,3] > x) && (tempmat[1,4] > x) && (tempmat[1,5]
> x)&& (tempmat[1,6] > x)&& (tempmat[1,7] > x)&& (tempmat[1,8] > x)&&
(tempmat[1,9] > x)&& (tempmat[1,10] > x)&& (tempmat[1,11] > x)&&
(tempmat[1,12] > x)) {tempnew <- new("ZscoreEcoli", probe_id = object@probe_id,
expressionmat = tempmat);return(tempnew)} else return(0)})
```

```
R Scripts
eczscore<- readdata.ZscoreEcoliExp("zscoreEcoli")
EcoliMat <- sapply(eczscore,getmatrix,1); EcoliMatFinal <- setdiff(EcoliMat,0)
mymat <- NULL; for (j in seq(along = EcoliMatFinal)){tempomat<-
EcoliMatFinal[[j]]@expressionmat; rownames(tempomat) <-
EcoliMatFinal[[j]]@probe_id; mymat<- rbind(mymat,tempomat)}
heatmap(mymat)
```

6.2.2 Identifications of the attractors in a simple Boolean network using BoolNet package

Biological entities can have 2 possible logical states ON or OFF i.e. transcription of gene being either ON or OFF, protein is either Present or Absent etc. A system is more intuitively understandable by logical assumptions. Mainly Boolean logical network is used for the gene regulatory networks. Here we have implemented it on a more simple metabolic reactions network. Here we have chosen 3 reactions of genes involved in metabolic pathways of *Mycobacterium tuberculosis,* which are consistently highly expressed in 12 different strains (Gao et al., 2005). This Boolean network consists of 8 genes. Simple rules are written for reaction by using AND and OR Boolean operators. Attractors are the points in a network towards which the system is evolved. Attractors can be steady states or cycles. These are the states where system resides most of the time (Müssel et al., 2010).

```
############
Boolean network with 8 genes
Involved genes:
nad coa oaa pyr sdhlam accoa succoa cit
Transition functions:
nad = nad
coa = coa
oaa = oaa
pyr = pyr
sdhlam = sdhlam
accoa = (coa & nad & pyr) | (cit & coa)
```

succoa = (coa & sdhlam)
cit = (accoa & oaa)
##########
Abbreviations
NAD = NAD, COA = Coenzyme-A, OAA = oxaloacetate, PYR = pyruvate, SDHLAM= S-adenosyl-L-methionine, ACCOA= acetyl-CoA, CIT = citrate, SUCCOA = succinyl-coA
Description of network: -

1. AcetylCoA is formed by 2 reactions: Coenzyme A and NAD and Pyruvate OR Citrate and Coenzyme A
2. SuccinylCoA is formed by Coenzyme A and S-adenosyl-L-methionine
3. Citrate is formed by Acetyl-CoA and Oxaloacetate
4. Species considered as constant (whose rules are not defined) e.g. NAD, OAA, PYR, SDHLAM and COA

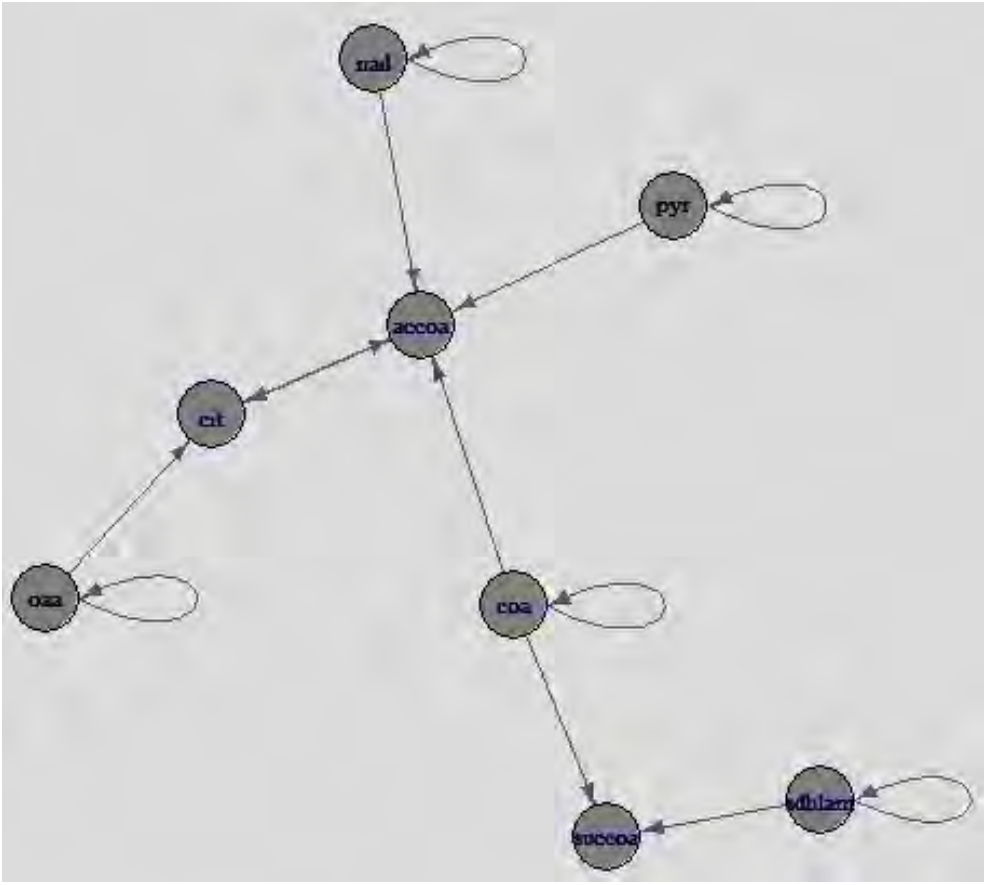


Fig. 10. Dependency among species in the example network of 8 genes

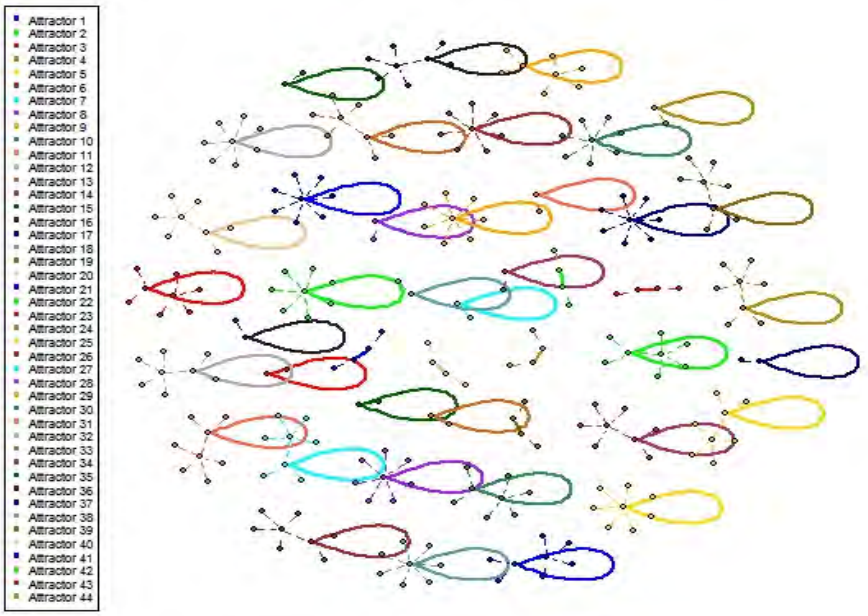


Fig. 11. Plot of different attractors in the network. Node is representing state and line representing state transition. Colour in plot is corresponding to different basin of attraction. We obtained total of 44 attractors for the network of which 39 have single state and 5 attractors have 2 states.

Fig. 12. Plot of Attractors with 2 states (from subsets 39 to 44). Red colour given for inactive genes and green colour is given for active genes. Each attractor given in plot is contributing 1.56% in the network.

Even though this is simple reaction network, yet it is giving the view about the states where system can reside most of the time i.e. attractors in the network. We can study the knockout and over expression in a complex network. Robustness study of the network is also possible by studying network behavior through knocking out genes.

6.3 Chemoinformatics data

The Chemoinformatics branch is the interface between Computer applications and Chemistry and deals with problems of the field of the chemistry. Chemoinformatics concentrates on molecular modelling, chemical structure coding and searching, data visualization etc.

Molecular modelling involves the use of theoretical methods and computational techniques to model or mimic the behavior of molecules. It helps reduce the complexity of the system, allowing many more particles (atoms) to be considered during simulations. Data visualization is the study of the visual representation of data by graphical means.

Chemoinformatics is useful specially to solve drug discovery related problems. Drug-like or Lead identifications are done through various in-silico methods in chemoinformatics. Drug-like compounds refer to the compounds, which follow the lipinski's rule and have structural similarities with the known drugs and bind to the active site of the target but have not been tested in laboratory. There are also various databases available helping in this direction.

Functions	**Tools**	**Links**
Databases for searching Known Inhibitors	Pubchem Pubmed Drug Bank	http://pubchem.ncbi.nlm.nih.gov/search/search.cgi http://www.ncbi.nlm.nih.gov/pubmed http://www.drugbank.ca/
Molecular Visualization	Pymol Rasmol SwissPDBviewer	http://www.pymol.org http://rasmol.org http://spdbv.vital-it.ch/
Structres drawn StructuresViewed	Marvin Sketch Chemmine	http://www.chemaxon.com/products/marvin/marvinsketch http://bioweb.ucr.edu/ChemMineV2
File Format Translator	Smile Translator OpenBabel	http://cactus.nci.nih.gov/translate/ http://openbabel.org/wiki/Main_Page
Drug Designing	AutoDock GOLD	http://autodock.scripps.edu/ http://www.ccdc.cam.ac.uk/products/life_sciences/gold/
Toxicity prediction	Toxtree	http://toxtree.sourceforge.net/
Molecular dynamics simulation	GROMACS	http://www.gromacs.org/

Table 3. Softwares used in Cheminformatics.

Package Name	Principle	Application
ChemmineR	Uses Tanimoto coefficient as the similarity measure	Atom-pair descriptors calculation with the help of functions included within it, 2D structural similarity searching, clustering of compound libraries, visualization of clustering results and chemical structures.
Rcdk	Allow an user to access functions of CDK (JAVA library for Chemiinformatics) on R platform	Reading molecular file formats, performing ring perception, aromaticity detection to fingerprint generation and determining molecular descriptors of various properties of Drug-Like compounds
Rpubcem	Uses various functions for data retrieval	Helps access the datas and assays of compounds from pubchem
ic50	Helps determine the efficiency of a newly found drug like molecule	Calculates IC50
Bio3D	Analysis of protein structure and sequence data	Protein structure analysis, comparative analysis with different proteins, aligns protein sequences.

Table 4. Tools useful in Chemoinformatics in R

6.3.1 Identification of drug-like compounds

In this era, with the rise in number of infectious life threatening diseases due to clever change at sequence level of the pathogenic organism, discovery of new potential drugs holds immense importance. In this direction various in-silico chemoinformatics tools mentioned in Table 3 are helpful. In the problem regarding identification of Drug-like compounds the initial step would include literature search for known inhibitors of the disease target. Similarly a database of publicly available drug like molecules is obtained from various databases like ZINC (Irwin et al., 2005), NCI (Voigt et al., 2001) database etc. Thereafter Marvin sketch (MarvinSketch 5.3.8, 2010) may be used to draw the known inhibitors and their analogs, which need to be saved in file formats: Structure Data Format (SDF) and Simplified Molecular Input Line Entry Specification (SMILES). Using ChemmineR package (Cao et al, 2008) on R platform similarity search of Known inhibitor with the database of publicly available drug-like molecules is done using Tanimoto Coefficient where a desirable cutoff score e.g. 0.6 may be used as cutoff score to obtain a list of similar lead compounds. This list corresponds to a number of compounds similar to the known inhibitors. The properties of these similar compounds are calculated with another package of R called rcdk (Guha et al, 2007). The Lipinski's rule of five can be applied to further shortlist. The solubility of these compounds can be analysed by pHSol 1.0 Server (Hansen et al, 2006). For analyzing protein-ligand binding, docking is done using AutoDock software (Goodsell et al, 1996). If the ligand binds to the active site, only then it may have pontential to interfere with protein function thereby eligible for further testing. Thereafter to test its potential further, energy minimization and simulation are done with various softwares eg. GROMACS (Hess et al., 2008). The ligand fulfilling all the desirable drug like quality and showing good stability over a reasonable time period of simulations (5-10 nanoseconds) may be selected as a candidate for testing. This process is represented as decision tree (Figure 13).

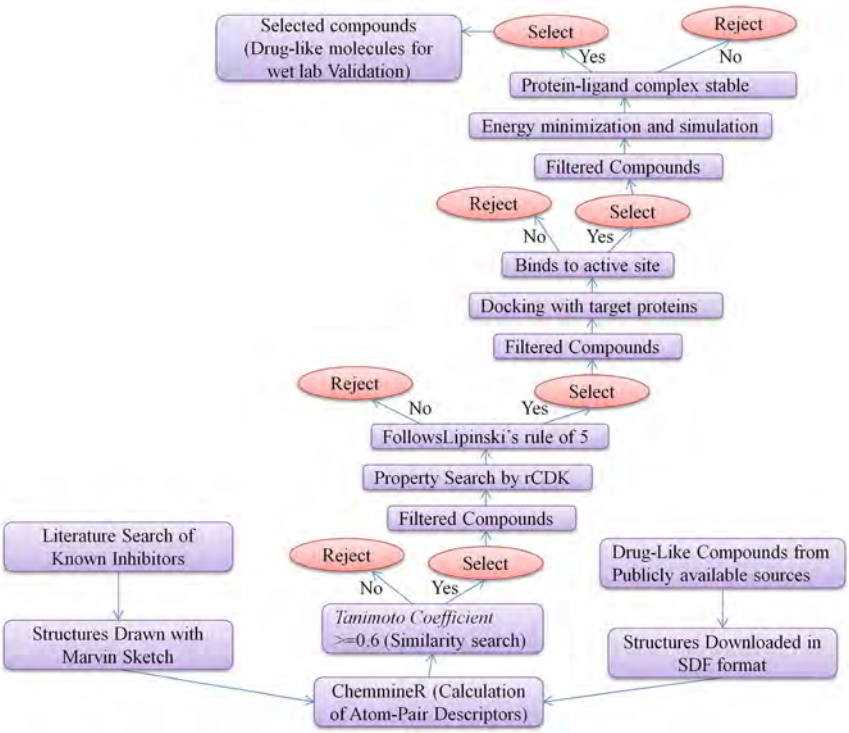
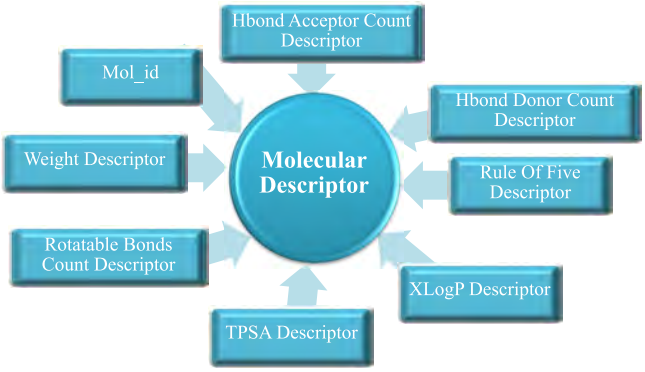


Fig. 13. Decision tree to identify Drug like compounds.




```
setClass(Class="MolDescriptors",representation=representation
(Mol_id="numeric",TPSA="numeric",nHBAcc="numeric",nHB
Don="numeric",nRotB="numeric",LipinskiFailures="numeric",
MW="numeric",XLogP="numeric",SMILE="character"))
```

```
readdata.MolDescriptors=
function(filename){x=readLines(
con=filename);temp=NULL;obj=
NULL;for(i in
2:length(x)){temp=unlist(strsplit(
x[i],"\t"));print
(temp);obj=c(obj,new("MolDescri
ptors",Mol_id=as.numeric(temp[
1]),TPSA=as.numeric(temp[2]),n
HBAcc=as.numeric(temp[3]),nH
BDon=as.numeric(temp[4]),nRot
B=as.numeric(temp[5]),LipinskiF
ailures=as.numeric(temp[6]),MW
=as.numeric(temp[7]),XLogP=as.
numeric(temp[8]),SMILE=temp[
9]));}return(obj);}
```

Fig. 14. Representation of S4 Class "MolDescriptors" with R scripts to accomplish the construction.

```
S4 method to get filtered molecular Descriptor Data
setGeneric("FilterMols",function(obj,x)standardGeneric("FilterMols"));
setMethod("FilterMols",
"MolDescriptors",function(obj,x){if(obj@LipinskiFailures==x){return(obj);}else{return(
0);}})
```

```
Example R Scripts to get filtered molecular Descriptor Data
result=sapply(mol,FilterMols,1); result_final=setdiff(result,"0")
```

6.4 Text mining

During last few decades there has been enormous increase in the size of research data in the form of scientific articles, abstracts and books, online databases and many more. This text data may be structured or unstructured and need of hour is to mine for useful information from text data.

Thus text mining has evolved widely as an interdisciplinary discipline using methods from computer sciences, linguistics and statistics. R provides intelligent ways in accessing and integrating the treasure of information hidden in the scientific journals, papers and other electronic media. It is most often used to perform statistical and data mining analyses and is best known for its ability to analyze structured data. Majority of people read only abstracts of papers so as to save time and avoid in going into details of irrelevant articles. The **tm** R-package provides complete platform that efficiently processes various text documents to extract useful information (Feinerer et al, 2008). The database backend support also minimizes the memory demands to handle very large data sets in R. It accepts text data either from local database or directly from online database.

6.4.1 An example of text mining PubMed abstracts to get frequently appearing gene symbols and drug names

We downloaded abstracts for PubMed query "Glioma" from PubMed. We formatted these abstracts in R so as to get a file containing PMIDs, Titles, Abstract Text and Journal Name in separate columns. We create the S4 class object of this Abstract file. Now if we have to search for any abstract containing a pattern or word of our interest we use *regexpr* function in R to get all those abstracts e.g. pathway.

From these filtered Glioma pathway abstracts a Corpus (collection of large text) is generated using tm package function. This Corpus is subjected to Stemming, Stop word removal, common English word removal using R libraries rJava, RWeka (Hornik et al, 2009), RWekajars, slam, Snowball (Hornik, 2009), Corpora. We can set other controls as well. Now we can create term document matrix from this Corpus, containing all terms as rows and abstracts as columns. From this term document matrix we can extract the terms within different frequency ranges i.e., their number of occurrences in each abstract indicating their importance. Thus we can predict that terms which have higher frequency of occurrence in the matrix are more important and are related in some sense. We have filtered all the terms with frequency of their occurrence greater than 5.

These terms were 'intersect' with HUGO Gene Nomenclature Committee data set (HGNC) (Bruford et al, 2008) and drug names from DrugBank (Knox, et al 2011). We thus get the list of all the gene symbols, gene names, gene aliases and drug names in the said frequency range from term document matrix. We now determine the total count of their occurrence in matrix so as to rank these extracted genes and drugs. We can make clusters from these data and can find gene-drug, gene-gene, and drug-gene interactions in different ranges of correlation. The decision tree is shown in Figure 15. The dataclasses with their representations is described in Figure 16.

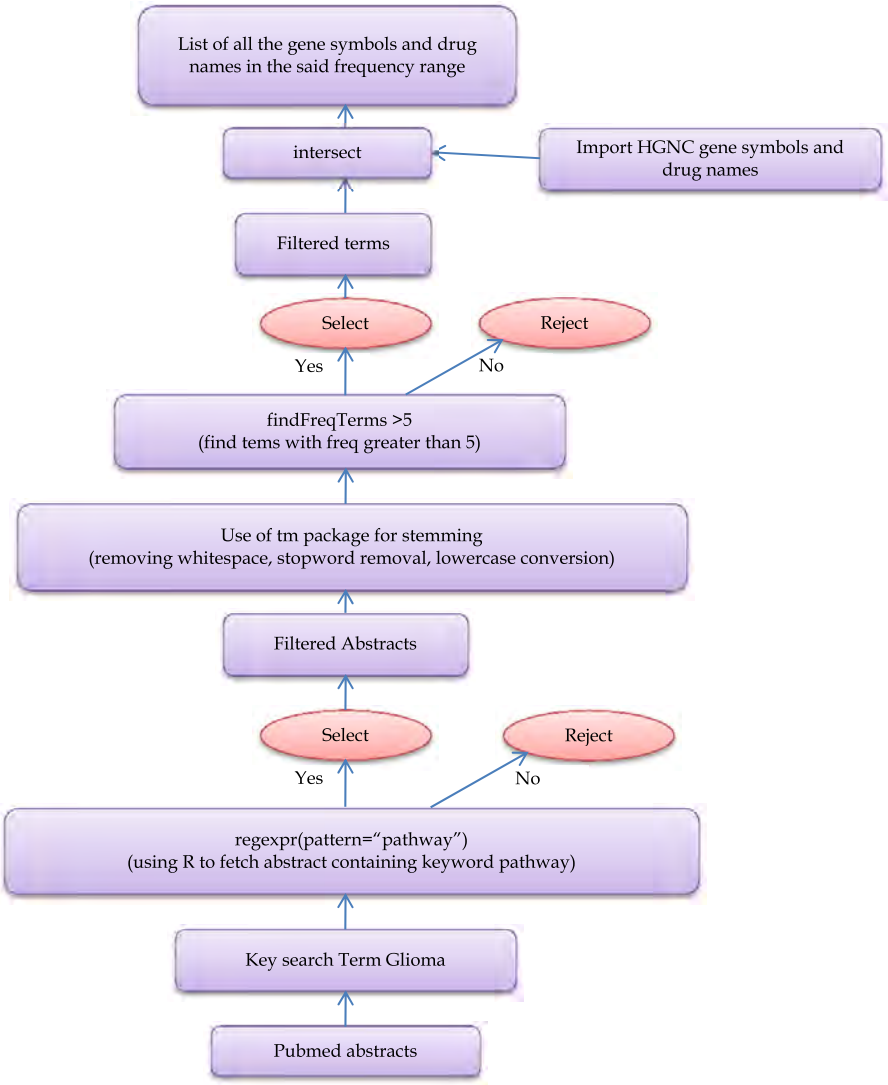


Fig. 15. Decision tree to identify **Glioma** that include pathway in the text.


```
readdata.textmining<-
function(xz){xa<- readLines(con =
xz);tempy<- NULL;for (i in seq ( along
= xa)){tempx<-
unlist(strsplit(xa[i],"\t"));tempy<-
c(tempy, new("TextMining", PMID =
as.numeric(tempx[1]), Title = temp[2],
Abstract = tempx[3], Journal_Name =
tempx[4]))};return(tempy)}
```

```
setClass("TextMining", representation(PMID = "numeric",
Title ="character",Abstract = "character", Journal_Name =
"character"))
```

Fig. 16. Representation of S4 Class "TextMining" with R scripts to accomplish the construction.

```
S4 Methods
setGeneric("getAbstract", function(object) standardGeneric("getAbstract"));
setMethod("getAbstract","TextMining",function(object){if
(regexpr(pattern="pathway",object@Abstract,fixed=TRUE)!=-1){return
(object@Abstract)}});
```

6.5 Getting data from Pfam and PRINTS for a specified domain or pattern name.

Pfam data for a specified domain name or PRINTS data for specified pattern name may be extracted using S4 scripts. The process is summarized below in form of decision tree (Figure 17). The dataclasses with their representations is described in Figure 18 and Figure 19.

```
R Scripts
filtered<- sapply(textfile,getAbstract)
doc1 <- filtered[!sapply(filtered,is.null)]
library(tm);library(rJava);library(RWekajars);library(RWeka);library(slam);library(
Snowball);library(corpora);
doc2<-Corpus(VectorSource(doc1))
setcontrol<-list(minDocFreq = 6, removeNumbers = FALSE, stemming = TRUE,
stopwords = TRUE)
TDM<-TermDocumentMatrix(doc2,control=setcontrol)
Terms<-inspect(TDM[1:17,1])
Terms1<-as.data.frame(Terms)
write.table(Terms1,file="Terms1.txt",sep="\t")
write.table(Terms1,file="Terms1.txt",col.names=FALSE,sep="\t")
Terms2<-read.table("Terms1.txt",header=FALSE,sep="\t")
FilteredTerms<-as.character(Terms2[1:17,1])
load("HGNC_GENE_IDS.RData")
load("DrugNamesDrugBank.RData")
AllGeneSymbols<-c(as.character(HGNC_GENE_IDS[1:19363,2])); AllGeneNames<-
c(as.character(HGNC_GENE_IDS[1:19363,3])); AllGeneAliases<-
c(as.character(HGNC_GENE_IDS[1:19363,6])); AllDrugNames<-
c(as.character(DrugNamesDrugBank[1:6824,1]))
getGeneSymbols<-intersect(AllGeneSymbols,FilteredTerms); getGeneNames<-
intersect(AllGeneNames,FilteredTerms); getGeneAliases<-
intersect(AllGeneAliases,FilteredTerms); getDrugNames<-
intersect(AllDrugNames,FilteredTerms)
z1<- charmatch(getGeneSymbols,FilteredTerms)
for(i in 1:4){b<-NULL;b<-z1[i];sumFreqOccurence<-NULL;sumFreqOccurence<-
sum(inspect(TDM[b,1:22]));write(sumFreqOccurence,file="SumFreqOccurence.txt",
append=TRUE,sep=",")}
```

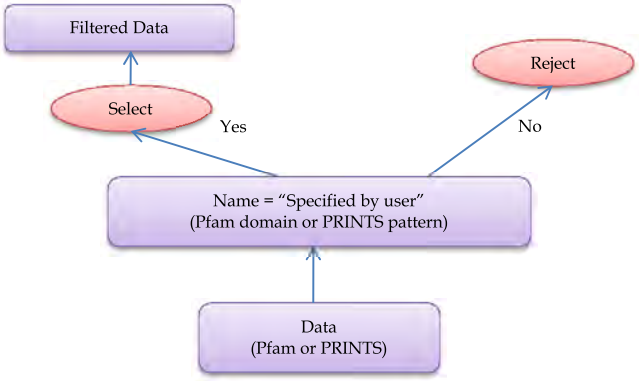
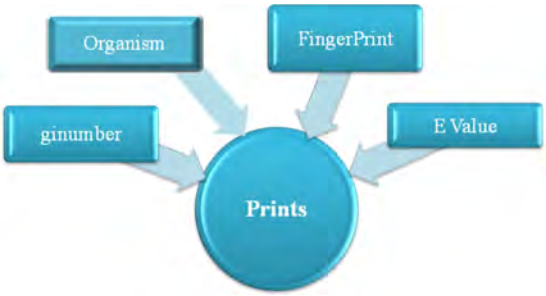


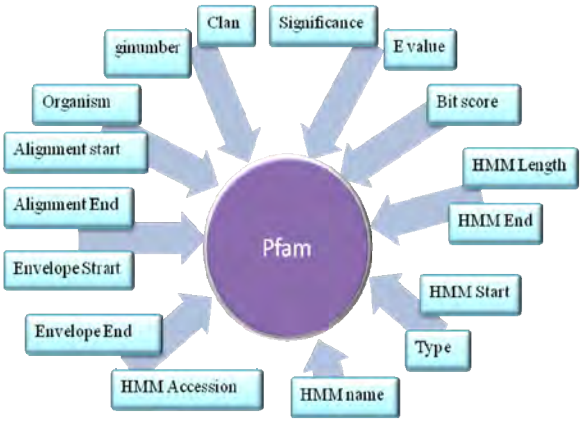
Fig. 17. Decision tree to identify data from Pfam and PRINTS for a specified domain or pattern name.




```
setClass("Prints",representation(ginumber =
"numeric",organism
="character",FingerPrint="character",E_value=
"numeric"))
```

```
readdata.prints<- function(xz){xa<-
readLines(con = xz); tempy<- NULL;for (i
in seq ( along = xa)){tempx<-
unlist(strsplit(xa[i],"\t"));tempy<- c(tempy,
new("Prints", ginumber =
as.numeric(tempx[1]), organism =
tempx[2],FingerPrint=tempx[3], E_value=
as.numeric(tempx[4])))};return(tempy)}
```

Fig. 18. Representation of S4 Class "Prints" with R scripts to accomplish the construction.




```
setClass("Pfam",representation(ginumber =
"numeric",organism ="character",alignment_start =
"numeric",alignment_end = "numeric",envelope_start=
"numeric",envelope_end=
"numeric",hmm_acc="character",hmm_name="character",
type="character",hmm_start= "numeric",hmm_end=
"numeric",hmm_length= "numeric",bit_score=
"numeric",E_value= "numeric",significance=
"numeric",clan="character"))
```

```
readdata.pfam<- function(xz){xa<-
readLines(con = xz); tempy<-
NULL;for (i in seq ( along =
xa)){tempx<-
unlist(strsplit(xa[i],"\t"));tempy<-
c(tempy, new("Pfam", ginumber =
as.numeric(tempx[1]), organism =
tempx[2],alignment_start=
as.numeric(tempx[3]),alignment_end=
as.numeric(tempx[4]),envelope_start=
as.numeric(tempx[5]), envelope_end=
as.numeric(tempx[6]),hmm_acc=
tempx[7],hmm_name= tempx[8],type=
tempx[9],hmm_start=
as.numeric(tempx[10]),hmm_end=
as.numeric(tempx[11]),hmm_length=
as.numeric(tempx[12]),bit_score=
as.numeric(tempx[13]),E_value=
as.numeric(tempx[14]), significance=
as.numeric(tempx[15]),clan=
tempx[16]))};return(tempy)}
```

Fig. 19. Representation of S4 Class "Pfam"along with R scripts to accomplish the construction.

```
S4 Methods
setGeneric("get_prints",function(object,x)
standardGeneric("get_prints"));setMethod("get_prints","Prints",function(object,x){if
( object@FingerPrint == x) {tempo<-
paste(object@ginumber,object@organism,object@FingerPrint,object@E_value, sep="
"); return (tempo)}else {return(0)} })
setGeneric("get_pfam",function(object,x)
standardGeneric("get_pfam"));setMethod("get_pfam","Pfam",function(object,x){if (
object@hmm_name == x) {tempo<- paste(object@ginumber,object@organism,
object@alignment_start, object@alignment_end, object@envelope_start,
object@envelope_end, object@hmm_acc, object@hmm_name, object@type,
object@hmm_start, object@hmm_end, object@hmm_length, object@bit_score,
object@E_value, object@significance, object@clan, sep=" "); return (tempo)}else
{return(0)} })
```

```
R Scripts
result1<- sapply(eclprints,get_prints,"HOMSERKINASE");result1<- setdiff(result1,"0")
result1<- sapply(eclpfam,get_pfam,"DnaJ");result1<- setdiff(result1,"0")
```

7. Parallel computing

When the data size is large (example in millions) and fast information calculation and retrieval is needed, a single modern computational processor fails to fulfill the purpose. In that case, a number of processors are needed to work simultaneously, each carrying out same set of operations on different data objects. This type of approach is called *Parallelization on data level;* the processing time for a single object is not being reduced but a number of data objects are being processed during the same time-interval by separate processors.

The Rmpi package (Yu, 2010) is helpful in this direction. We used ChemmineR (Cao et al 2008) using Rmpi, an interface to MPI (Message Passing Interface). In MPI, processes communicate with each other by sending and receiving messages. We made a library of small molecules of size around 26 millions in SDF file format from various publicly and commercially available sources; these molecules were distributed over 1792 files. We have used ChemmineR's cmp.parse() function to get corresponding atom-pair descriptors and these were stored in .rda files. These .rda files constitute our database. Our aim was to find molecules similar to a given query molecule.

On a typical workstation of 1GB RAM and Intel(R) Pentium(R) 4 CPU 3.40GHz, it takes about 5 hrs to complete a similarity search. In order to increase the speed we have used Rmpi (version: 0.5-8) along with Open MPI (version 1.3.2) – A High Performance Message Passing Library and implemented the Data Level Parallelization on ROCKS (release 4.3) cluster- an open-source Linux cluster distribution with eight nodes having Intel(R) Xeon(TM) CPU 3.60GHz processors. The result was a significant increase in performance and the job was done within 30 minutes. This approach has been shown in Figure 20 as a flowchart.

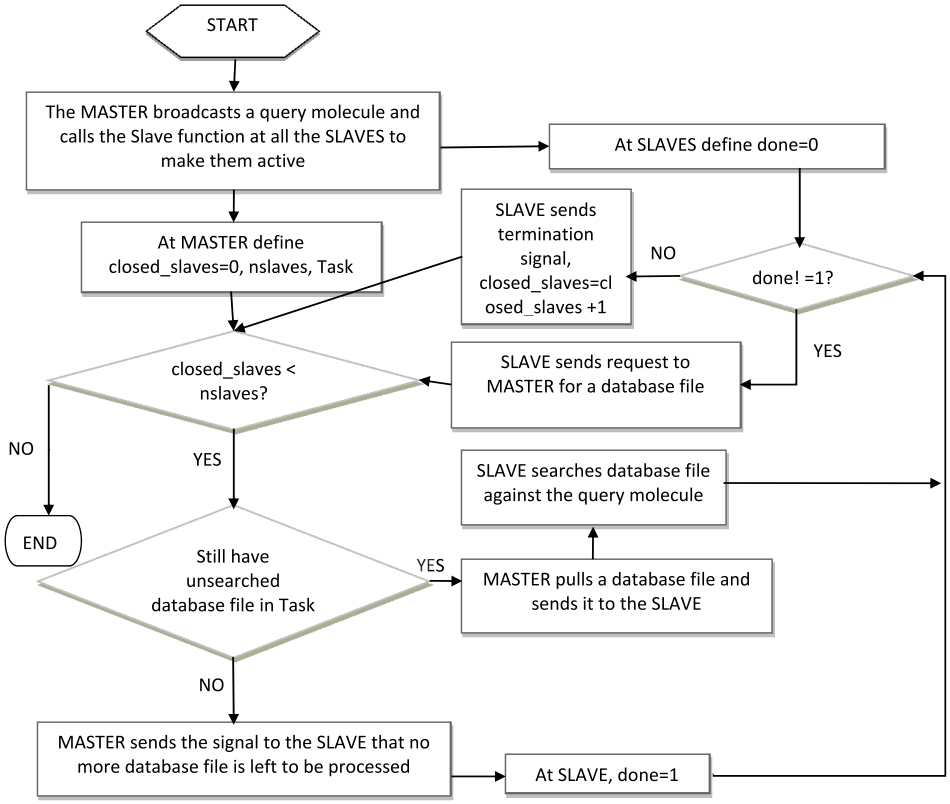


Fig. 20. Figure describing the parallel computing process, parallelization on data level.

8. Building your own package

The various R objects and S4 methods can be encapsulated into .RData named according to the selected problem. All these packages can be sourced from the link http://sourceforge.net/projects/sysbior/.

9. Conclusion

Various packages to handle biological data of diverse type of problems have been developed with the help of S4 object oriented programming. The S4 objects can be handled with ease applying S4 methods to fetch useful data. The process described here may also be modified with novel thought process to create S4 methods to fetch useful analysis data accomplishing other conditions.

10. References

Bui, H.H., Sidney, J., Peters, B., Sathiamurthy, M., Sinichi, A., Purton, K.A., Mothé, B.R., Chisari, F.V., Watkins, D.I. & Sette, A. (2005). Automated generation and evaluation

of specific MHC binding predictive tools: ARB matrix applications, *Immunogenetics*, Vol. 57, No. 5, (June 2005), pp. 304-314.

Bruford, E.A., Lush, M.J., Wright, M.W., Sneddon, T.P., Povey, S. & Birney, E. (2008). The HGNC Database in 2008: a resource for the human genome, *Nucleic Acids Res*, Vol. 36, Suppl No. 1, (January 2008), pp. D445-D448.

Cao, Y., Charisi, A., Cheng, L.C., Jiang, T. & Girke, T. (2008). ChemmineR: a compound mining framework for R, *Bioinformatics*, Vol. 24, No. 15, (August 2008), pp. 1733-1774.

Chaudhuri, R., Ahmed, S., Ansari, F.A., Singh, H.V. & Ramachandran, S. (2008). MalVac: database of malarial vaccine candidates, *Malar J*, Vol. 7, No. 184,(September 2008), pp. 1-7.

Chaudhuri, R., Ansari, F.A., Raghunandan, M.V. & Ramachandran, S. (2011). FungalRV: Adhesin prediction and Immunoinformatics portal for human fungal pathogens. *BMC Genomics*, Vol. 12, No. 192, (April 2011).

Cheadle, C., Vawter, M.P. & Freed, W.J. (2003). Analysis of Microarray data using Z-score transformation, *Mol Diagn*, Vol. 5, No. 2, (May 2003), pp. 73-81.

Feinerer, I., Hornik, K. & Meyer, D. (2008). Text mining infrastructure in R, *Journal of Statistical Software*, Vol. 25, No. 5, (March 2008), ISSN 1548-7660.

Fiers, M.W., Kleter, G.A., Nijland, H., Peijnenburg, A.A., Nap, J.P. & van Ham, R.C. (2004). Allermatch, a webtool for the prediction of potential allergenicity according to current FAO/WHO Codex alimentarius guidelines, *BMC Bioinformatics*, Vol. 5, (September 2004).

Gao, Q., Kripke, K.E., Saldanha, A.J., Yan, W., Holmes, S. & Small, P.M. (2005). Gene expression diversity among Mycobacterium tuberculosis clinical isolates, *Microbiology*, Vol. 151, No.1, (January 2005), pp. 5-14.

Gentleman, R.C., Carey, V.J., Bates, D.M., Bolstad, B., Dettling, M., Dudoit, S., Ellis, B., Gautier, L., Ge, Y., Gentry, J., Hornik, K., Hothorn, T., Huber, W., Iacus, S., Irizarry, R., Leisch, F., Li, C., Maechler, M., Rossini, A.J., Sawitzki, G., Smith, C., Smyth, G., Tierney, L., Yang, J.Y. & Zhang, J. (2004). Bioconductor: open software development for computational biology and bioinformatics, *Genome Biol*, Vol. 5, No. 10, (September 2004).

Goodsell, D. S., Morris, G. M. & Olson, A. J. (1996). Automated Docking of Flexible Ligands: Applications of AutoDock, *J. Mol. Recognition*, Vol. 9, pp. 1-5.

Guha, R. (2007). Chemical Informatics Functionality in R, *Journal of Statistical Software*, Vol. 18, No. 5, (January 2007), ISSN 1548-7660.

Hansen, N.T., Kouskoumvekaki, I., Jørgensen, F.S., Brunak, S. & Jónsdóttir, S. O. (2006). Prediction of pH-dependent aqueous solubility of druglike molecules, *J Chem Inf Model*, Vol. 46, No. 6, (November 2006), pp. 2601-2609.

Hess, B., Kutzner, C., van der Spoel, D. & Lindahl, E. (2008). GROMACS 4: Algorithms for Highly Efficient, Load-Balanced, and Scalable Molecular Simulation, *J. Chem. Theory Comput.*, Vol. 4, No. 3, (February 2008), pp. 435–447.

Hornik, K. (August 2009). Snowball: Snowball Stemmers, In: *R-project.org*, 12.12.2010, Available from: http://CRAN.R-project.org/package=Snowball

Hornik, K., Buchts, C. & Zeileis, A. (2009). Open-Source Machine Learning: R Meets Weka, *Computational Statistics*, Vol. 24, No. 2, (May 2009), pp. 225-232, ISSN: 0943-4062.

Irwin, J.J. & Shoichet, B. K. (2005). ZINC--a free database of commercially available compounds for virtual screening, *J Chem Inf Model.*, Vol. 45, No. 1, (2005), pp. 177-182.

Knox, C., Law, V., Jewison, T., Liu, P., Ly, S., Frolkis, A., Pon, A., Banco, K., Mak, C., Neveu, V., Djoumbou, Y., Eisner, R., Guo, A.C. & Wishart, D.S. (2011). DrugBank 3.0: a comprehensive resource for 'omics' research on drugs, *Nucleic Acids Res.*, Vol. 39, Suppl. No. 1, (January 2011), pp. 1035-1041.

Li, P., Zhang, C., Perkins, E., Gong, P. & Deng, Y. (2007). Comparision of probabilistic Boolean network and dyanamic Bayesian network approaches for inferring regulatory networks, *BMC Bioinformatics*, (November 2007), Vol. 8, Suppl. No. 7.

Lundegaard, C., Lamberth, K., Harndahl, M., Buus, S., Lund, O. & Nielsen, M. (2008). NetMHC-3.0: accurate web accessible predictions of human, mouse and monkey MHC class I affinities for peptides of length 8-11, *Nucleic Acids Res.*, Vol. 36, Suppl. No. 2, pp. W509-W512.

Maimon, O. & Rokach, L. (2005). Decision Trees, In: Data Mining and Knowledge Discovery Handbook, O. Maimon, & L. Rokach, (Ed.), pp. 165-192, Springer, ISBN: 978-0-387-25465-4, United States of America.

MarvinSketch 5.3.8. (2010). In: ChemAxon, 10.11.2010, Available from:
http://www.chemaxon.com

Müssel, C., Hopfensitz, M. & Kestler, H.A. (2010). BoolNet--an R package for generation, reconstruction and analysis of Boolean networks, Bioinformatics, Vol. 26, No. 10, (May 2010), pp. 1378-1380

Nelson, D.L. & Cox, M.M. (2000). Enzymes, In: Lehninger Principles of biochemistry, 3rd Edition, W.H. Freeman, (Ed.), pp. 190-237, Worth Publishers, ISBN 1-57259-9316, New York.

Parker, K.C., Bednarek, M.A. & Coligan, J.E. (1994). Scheme for ranking potential HLA-A2 binding peptides based on independent binding of individual peptide side-chains, J Immunol., Vol. 152, No. 1, (January 1994), pp. 163-175, ISSN: 0022-1767.

R Development Core Team (2010). R: A language and environment for statistical computing, In: R Foundation for Statistical Computing, ISBN 3-900051-07-0, Available from: http://www.R-project.orgSaha, S. & Raghava, G.P.S. (2006). AlgPred: prediction of allergenic proteins and mapping of IgE epitopes, *Nucleic Acids Res.*, Vol. 34, Suppl. No. 2, (July 2006), pp. W202-209.

Saha, S. & Raghava, G.P.S. (2006). Prediction of Continuous B-cell Epitopes in an Antigen Using Recurrent Neural Network, *Proteins*, Vol. 65, No. 1, (October 2006), pp. 40-48.

Saha, S. & Raghava, G. P. S. (2007). Prediction methods for B-cell epitopes, *Methods Mol. Biol.*, Vol. 409, No. 4, pp. 387-394.

Singh, H. & Raghava, G. P. S. (2001). ProPred: Prediction of HLA-DR binding sites, *Bioinformatics*, Vol. 17, No. 12, (December 2001), pp. 1236-1237.

Tyson, J., Chen, K. & Novak, B. (2001). Network dynamics and cell physiology, *Nature Rev. Mol Cell. Biol.*, Vol. 2, No. 12, (December 2001), pp. 908-916.

Viswanathan, G., Seto, J., Patil, S., Nudelman, G. & Sealfon, S. (2008). Getting started in biological pathway construction and analysis, *PLoS comp. biol.*, Vol. 4, No. 2 (e16), (February 2008), pp. 0001-0005.

Vivona, S., Gardy, J. L., Ramachandran, S., Brinkman, F.S., Raghava, G.P.S, Flower, D.R. & Filippini, F. (2008). Computer-aided biotechnology: from immuno-informatics to reverse vaccinology, *Trends Biotechnol.*, Vol. 26, No. 4, (April 2008), pp. 190-200.

Voigt, J.H., Bienfait, B., Wang, S. & Nicklaus, M.C. (2001). Comparison of the NCI open database with seven large chemical structural databases, *J Chem Inf Comput Sci.*, Vol. 41, No. 3, (May-Jun 2001), pp. 702-712.

Yu, H. (November 2010). Rmpi: Interface (Wrapper) to MPI (Message-Passing Interface), In: *cran.r-project.org*, 6.1.2011, Available from:
http://CRAN.R-project.org/package=Rmpi

Zhang, Q., Wang, P., Kim, Y., Haste-Andersen, P., Beaver, J., Bourne, P. E., Bui, H. H., Buus, S., Frankild, S., Greenbaum, J., Lund, O., Lundegaard, C., Nielsen, M., Ponomarenko, J., Sette, A., Zhu, Z. & Peters, B. (2008). Immune epitope database analysis resource (IEDB-AR), *Nucleic Acids Res.*, Vol. 36, Suppl. No. 2, (July 2008), pp. W513-W518.