UNIVERSITY OF CENTRAL GREECE
DEPARTMENT OF COMPUTER SCIENCE
AND BIOMEDICAL INFORMATICS

Intelligent Mining of Biomedical Data for the Creation of Integrated Physiological Models

Konstantinos Moutselos, BSc, MSc

PhD Dissertation

Supervisor
Prof. Ilias Maglogiannis

February 2013

Dedication

To my beloved wife Edlira and parents Christos and Foteini.

Acknowledgements

I take immense pleasure in thanking Prof. Ilias Maglogiannis for his guidance and supervision. Also, I wish to express my deep gratitude to Prof. Aristotelis Chatziioannou for being a constant source of inspiration, advice and support in carrying out the project work and to my entire scientific endeavour. I would like to thank also Professors Dimitrios Koutsouris and Vasilios Plagianakos, members of my Advisory Committee, and Antonios Aletras, Panteleimon Bagos, and Leonidas Alexopoulos, members of my Examining Committee.

I would like to give special thanks to all the colleagues from the Metabolic Engineering - Bioinformatics research group of the Institute of Biology, Medicinal Chemistry & Biotechnology of the National Hellenic Research Foundation for their cooperation and friendship. Finally, yet importantly, I am highly indebted to the staff and personnel from the Department of Computer Science and Biomedical Informatics of the University of Central Greece for providing a stimulating and welcoming academic environment.

Intelligent Mining of Biomedical Data for the Creation of Integrated Physiological Models

Konstantinos Moutselos, PhD

University of Central Greece, 2013

Supervisor: Ilias Maglogiannis

Abstract. In the context of contributing to the creation of Integrated Physiological Models, three novel approaches are proposed and developed, targeting at different levels of biomedical modelling. At the lower molecular level, the integration of metabolic pathways in a way capable of deriving descriptive and predictive dynamic models resulted in the creation of the KEGGconverter tool. KEGGconverter is capable of producing integrated analogues of metabolic pathways appropriate for simulation tasks, by inputting only KGML files and having an SBML model as output. The automated introduction of four case-specific default kinetic mechanisms in the models provides for the addition of the layer of kinetic equations, which is directed by a rule-based algorithm that was implemented for the specific task. At a higher level, the novel GOrevenge algorithm exploits the Gene Ontology to map genes to specific cellular pathways and vice versa in order to further infer the putative functional role of specific genes, through an associative context, starting from the results of various statistical enrichment analyses. Although this implementation utilizes the GO annotations, the algorithm can be generically extended to accommodate any biological ontology or controlled vocabulary definition. This is due to the exploitation of the encapsulated underlying topological network information and the interplay among the annotations and the annotated subjects.

Lastly, a novel methodology on multi-modal data fusion regarding separate datasets has been developed utilizing a dataset of features from skin lesion images and a dataset regarding microarray data. Both datasets were the output from studies on cutaneous melanoma, but involved different patients. The multivariate analysis applied to the unified datasets that were produced by this method indicated the better discrimination performance achieved in predicting the class of healthy/disease samples. This could lead not only to the creation of better analytical models of the specific disease, but also in dealing with modelling other complex diseases having multi-modal datasets as well.

Table of Contents

o

List of Tables

o

List of Figures

CHAPTER 1. INTRODUCTION

Biomedical data and physiological models are two sides of the same coin, in the effort of understanding and describing the human body. The assimilation of data from different scales of observation, concerning various organs and in varying temporal windows, can lead to models that more accurately simulate the involved functional mechanisms. Models, which are by definition an appropriate abstraction, can give insights regarding the internal workings of the acquired data. They can also direct the scientific community to more specific data acquisition procedures in order to validate and test the limits of a proposed physiological model. The unprecedented technological advances of biological instrumentation gave rise to a deluge of data, and therefore, caused the need of new algorithms for efficient processing, decoding and integration with the existing knowledge. Toward this path, the MMM concept [1] targets to a multivariate, multi-organ and multi-scale biomedical description. Furthermore, the P4 medicine extends the components of the clinical practice apart from the Personalized medicine to include also the Predictive, Preventive and Participatory characteristics [2]. Underpinnings to these contemporary trends are the Physiome Project [3], as well as the Virtual Physiological Human (VPH) Project of the EU within ICT for Health [4, 5]. The vision of assembling all the available and newly attained knowledge in a common computational infrastructure, in tandem with the creation of detailed physiological models, coincides with the aspiration of acquiring the holistic description of the human body.

In the context of an integrated physiological modelling vision, this dissertation deals with the following research questions:

- How the information residing in the Kyoto Encyclopedia of Genes and Genomes (KEGG) regarding different metabolic pathways of several organisms can be integrated and further processed in a way where dynamical modelling of higher parts of physiology and development of systemic hypotheses are feasible?

- How ontologies, such as Gene Ontology (GO), can further contribute to the illumination of data coming from wet lab experiments, such as microarrays?
- Is there a way of fusing separate data sources (possibly from different clinical or laboratory observations regarding multi-modal bio-medical data) related to the same pathology/disease in order to extract better biomarkers and built predictive models of higher accuracy?

In order to properly address the aforementioned questions, three novel approaches were proposed and developed, targeting different levels of biomedical modelling. At the lower molecular level, the integration of metabolic pathways in a way capable of deriving descriptive and predictive dynamic models resulted in the creation of the KEGGconverter tool. KEGGconverter is capable of producing integrated analogues of metabolic pathways appropriate for simulation tasks by inputting only KGML files.

At a higher level, the GOrevenge algorithm exploits the Gene Ontology to map genes to specific cellular pathways and vice versa in order to further infer the putative functional role of specific genes, through an associative context, starting from the results of various statistical enrichment analyses. Although this implementation utilizes the GO annotations, the algorithm can be generically extended to accommodate any biological ontology or controlled vocabulary definition. This is due to the exploitation of the encapsulated underlying topological network information and the interplay among the annotations and the annotated subjects.

Lastly, a novel methodology on multi-modal data fusion regarding separate datasets has been developed utilizing a dataset of features from skin lesion images and a dataset regarding microarray data, both concerning studies on cutaneous melanoma, yet involved different patients. The multivariate analysis performed on the unified datasets produced by this method indicated the better discrimination performance achieved in predicting the class of healthy/disease samples. This could lead to the creation of better analytical models of the specific disease at first, but also in dealing with modelling other complex diseases having multi-modal datasets as well.

The structure of this work is as follows:

The Chapter 2: Background section describes the VPH framework, the related scientific challenges in general, regarding ICT in particular, and the projected timeline of the anticipated impact the VPH activities will have. Next, the structure of the KEGG databases is presented (spanned by the three axes concerning Chemical, Genomic and Systems information). The new developments in KEGG, ways of accessing and retrieving the available data, plus issues that make the exploitation of this valuable data source difficult – in relation to the first research question. The background regarding the second research question of this work concerns the Gene Ontology structure and form, as well as the mechanisms of annotations and the relative evidence codes. Measures of similarity between GO terms and genes are introduced. The use of GO in a prioritization mechanism of genes, based on the genes' annotations, is presented, which is the starting basis of GOrevenge. The Gene Expression Omnibus (GEO) database is described next, its contents (formats and standards), organization, and available tools. Experimental data retrieved from GEO is used in dealing with the third research question of this work regarding data fusion procedures. Lastly, related work on the issue of fusing separate, multi-modal datasets is presented as well as background information on the cutaneous melanoma, a complex multigenic and multifactorial disease selected as a case study in the proposed new data fusion methodology. Also, the applied feature selection technology is presented, and this concludes the Background information section.

The Chapter 3: Material and Methods section first describes the KEGGconverter tool in detail, the mechanisms of merging KGML KEGG pathways, the conversion to SBML, and the insertion of kinetic information. Then the GOrevenge workflow is presented, the GO data and organisms' annotations assimilation procedure, and the computational and biological methodology used for the validation of the method. Lastly, there is a description of the two separate datasets used in the proposed data fusion method (image data and dermatoscopy and microarray data relating to cutaneous melanoma), the imputation techniques applied, and the AI algorithms employed for the multivariate statistical analysis of the unified datasets.

The Chapter 4: Experiments and Results section presents the outcomes from integrating 6 KEGG pathways central to human organism by using KEGGconverter in comparison to other available tools. Reports on the results of the statistical and biological evaluation of the GOrevenge algorithm on a Pancreatic Cancer gene set and to an Acute Lymphoblastic Leukemia gene set follows. Lastly, there is an analysis of the stability and the class-discriminating capability of a set of biomarkers retrieved regarding cutaneous melanoma by the proposed multi-modal fusing/imputation method.

Finally, at the Chapter 5: Discussion and Conclusion section, there is a summary of the reported outcomes for the three research directions addressed in this thesis in the context of integrated physiological models, the novel contributions of this work, and proposed future work directions.

Important parts of code in Java, Python, and R, developed during this work, are presented in line with the documentation and in the Appendices.

CHAPTER 2. BACKGROUND

2.1. THE OVERARCHING VISION: Virtual Physiological Human (VPH)

> “The Virtual Physiological Human (VPH) is a methodological and technological framework that, once established, will enable collaborative investigation of the human body as a single complex system. VPH is not ‘the supermodel’ that will explain all possible aspects of human physiology or pathology. It is a way to share observations, to derive predictive hypotheses from them, and to integrate them into a constantly improving understanding of human physiology/pathology, by regarding it as a single system.” [6]

The above description of the VPH concept delineates the ambitions as well as the boundaries of the EuroPhysiome Initiative. Amidst the technological advancements of the available instrumentation aiming at various physiological scales, in parallel with the development of new algorithms and methods of analysing the resulting deluge of data, the VPH vision emerges as a strategic approach for the conduction of targeted research. This approach dictates the orchestration of all potential contributors from a broad area of scientific disciplines in a coherent and synergistic manner in order to work towards the deciphering the physiology of the whole human body. This understanding, the construction of the whole picture encompassing the various hierarchical levels of physiological functions and their interactions, is the crucial missing factor from the current scientific endeavours and therefore the objective of the VPH.

According to the roadmap to the VPH [6], the framework will be: descriptive (i.e., regarding the depth and the breadth of observation recording), integrative (i.e. spanning different disciplines and scales), and predictive (i.e., allowing hypotheses to be verified from the output of constructed models).

2.1.1. The VPH Framework

2.1.1.1. Logical structure

2.1.1.1.1. Standards for Services

Web Services are described as the bonding interface among the different components of the framework, interconnecting data and algorithms through various platforms in a transparent way for the user, still well defined and easily extendable from the side of the software developer. As such, the services should be easy to find and amenable to assembling into more compounded workflows by pipelining, wrapping and nesting existing services.

2.1.1.1.2. Standards for MLs (Mark-Up Languages)

XML applications, specific for the modelling of various physiological functions, can describe models in machine-readable ways. These mark-up languages (e.g. SBML, CellML, etc.) can offer clear delineation of the different constituent parts as well as easy integration up to higher-order modelling.

2.1.1.2. Towards a collection of integrative "Core Models"

The 1972 model regarding the cardiovascular system's regulation by Guyton, Coleman and Granger [7] serves as a paradigm of the value and the importance of developing models which can adequately describe parts of physiology. Of course, such models cannot be expected to encompass all of the details for each aspect of an organism. Models, per se, contain the notion of abstraction. The important thing is that these models can be interconnected. Higher or coarser models should be able to delegate execution to lower and finer ones or the other way around. This notion captures both the top-down and bottom-up approaches.

2.1.1.3. Implementation and relevance to this work

The vision of VPH cannot, of course, be covered through the advances of a single project. On the contrary, the strategy is to allow for the progression of diverse projects to be aligned towards the common goal of integration on a long-term basis.

Potential directions for these developments are in areas such as:

- Data processing (conversion, big data analysis and data mining methods, and ontology-based data interoperability covering heterogeneous sources and data fusion),
- modelling (dealing with missing data and validity ranges, multi-level and multi-domain integration of dynamic systems),
- visualisation (virtual or augmented reality environments, ‘living simulations’),
- and knowledge management (methods for cross-level ontologies, semantic representation and integration).

At Figure 1 the hierarchical analysis of VPH is shown, regarding the various aspects of implementation. The nodes are full expanded only for the first level of the tree (i.e. after the root node), and the nodes Modelling and Data Processing for the second level. In the Figure, there are nodes that indicate how this work is connected under the VPH umbrella, according to the relative specifications of the Roadmap to the Virtual Physiological Human [6]. KEGGconverter work is relative to the modelling of metabolic processes in systemic scale. SBML models are capable of simulating dynamic systems due to the use of MathML for describing kinetic equations. The GOrevenge algorithm uses ontologies to pinpoint hub genes or GO nodes of importance, utilising semantic similarity distances. The third part of this work is connected to the category of feature extraction algorithms, using data mining methods for classification of healthy/disease cases of clinical datasets on cutaneous melanoma.

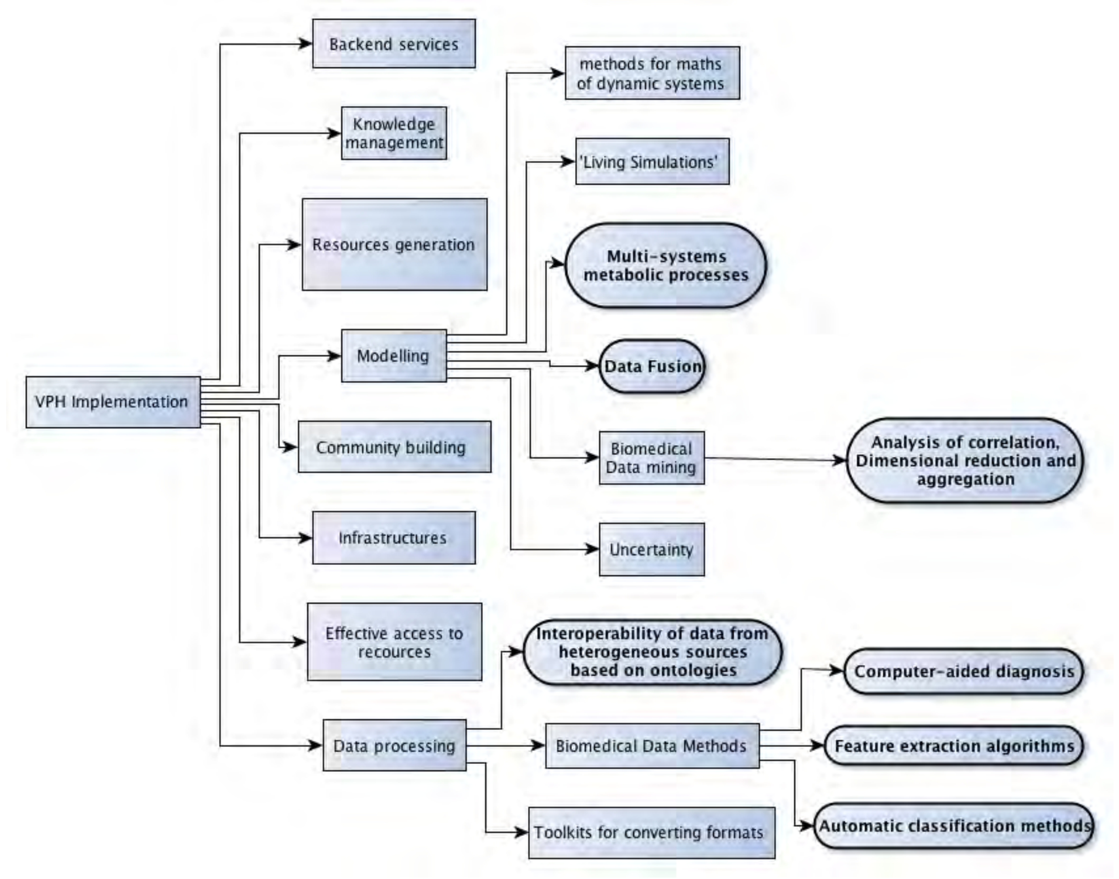

Backend services
Knowledge management
Resources generation
Modelling
VPH Implementation
Community building
Infrastructures
Effective access to recources
Data processing
methods for maths of dynamic systems
'Living Simulations'
Multi-systems metabolic processes
Data Fusion
Biomedical Data mining
Analysis of correlation, Dimensional reduction and aggregation
Uncertainty
Interoperability of data from heterogeneous sources based on ontologies
Biomedical Data Methods
Computer-aided diagnosis
Feature extraction algorithms
Automatic classification methods
Toolkits for converting formats

Figure 1: Implementation directions for the VPH. Aspects that have been tackled in this thesis are in oval shapes and bold.

2.1.2. Scientific challenges

The roadmap to VPH specifies various categories of scientific encounters regarding the realization of its goals: description, prediction, integration and ICT challenges. At Figure 2, the hierarchical tree of this analysis is shown.

2.1.2.1. Challenges in Description

The task of assembling precise input data for any system under consideration is of decisive importance: without accurate descriptions, all the attempts for model creation would be futile. The issues affecting this area:

- Data collection standards and protocols: parameters and conditions that should be recorded under various experiment categories, in order for the retrieved data to be reproducible and amenable to statistical integration.

- Data quality and error checking techniques to ensure the validity of the acquired datasets.
- Data fusion methods regarding heterogeneous data (such as from different clinical practices, cohort studies, acquisition instruments), along with data-reduction algorithms.
- Development of new acquisition instruments and techniques allowing for better spatiotemporal resolution.

2.1.2.2. Challenges in Prediction

The crucial part is to produce predictive models capable of “providing virtual cells, tissues, organs and systems that can be used in the development of novel drugs and treatments, and, ultimately, for patient-specific care regimes” [8]. Pertaining issues:

- Complexity adjustments regarding the modelled system (granularity)
- Interacting multi-phase and multi-scale models so as higher-order system properties, traversing scientific disciplines and organ boundaries can be deduced.
- Effects of variations among patients: normal, pathological, and normal versus pathological variations.
- Parameter estimation for cases where comprehensive or accurate experimental data are not available.

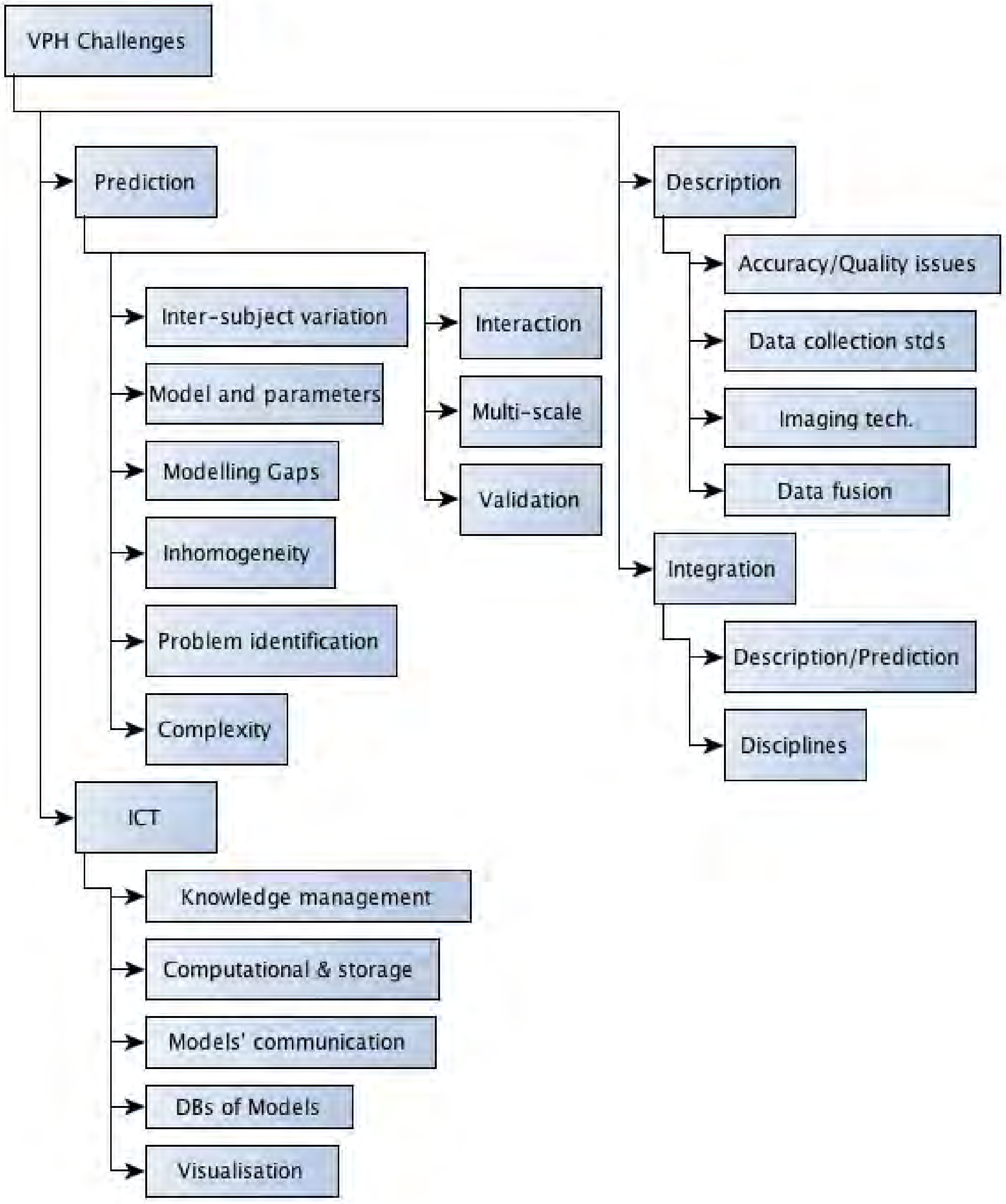


Figure 2: VPH Challenges, in compact layout.

2.1.2.3. ICT Challenges

A description of ICT tools that will help VPH integration follows:

- A comprehensive model repository that can assimilate all existing models.
- Development of a general model-coupling framework. It is suggested that this coupling could provide for a general coarse model to be enhanced by several

higher-resolution and specific models that could be fitted in the first, and respectively delegate required functionalities. In this direction, XML applications such as SBML and CellML offer the capability of structuring simulation-capable models that can embed other models through the XML semantics.

- Knowledge management software that can deal with the increasing volume of data and the resulted interpretations. This encompasses literature sources, as well as biomedical/clinical digital libraries necessary for the building and the validation of the developed models.

2.1.3. Timelines for the virtual physiological human

The achievements towards the VPH up to now and a list of current VPH projects with their scopes are documented in P. Hunter et al [9]. It is expected that the impact from the outcomes of these projects will affect at first the biomedical research community, the healthcare industry, and then the general public, sequentially. It is estimated that until 2014 an organization for the coordination of EU actions could be formed in tandem with directed research funding (e.g. FP8). It is projected a European large-scale action on personalized healthcare will be in operation in the period of 2014 – 2019.

2.2. KEGG: A VALUABLE RESOURCE FOR PHYSIOLOGICAL MODELS

KEGG (Kyoto Encyclopedia of Genes and Genomes) is a comprehensive network of databases, which aims to the integrative collection of biological information. It has been developed by the Institute for Chemical Research, Kyoto University (Kanehisa Laboratory), and its scope spans from cell to organisms. It organizes relevant biological knowledge in three categories: the chemical, the genomic and the network space. While the first two directions assimilate the available knowledge regarding molecules and genes, the third describes higher-order functions, i.e. the interaction networks regarding cellular tasks and operations. Its aspiration is to become a framework, which will enable the in-silico analysis of biological systems [10, 11].

2.2.1. Overview of KEGG

KEGG consist of 16 databases grouped into the three directions of biological knowledge [12]:

- Chemical Information (KEGG LIGAND). It encompasses five databases:
    - COMPOUND (metabolites: chemical structures of metabolic, pharmaceutical and environmental compounds. Manually curated and identified by the C accession number)
    - GLYCAN (carbohydrate structures, identified by the G accession number. Most records are derived from the CarbBank database and some are manually entered.)
    - REACTION (enzymatic reactions identified by the R accession number)
    - RPAIR (reactant pair transformations)
    - ENZYME (enzyme nomenclature)
- Genomic Information. It encompasses five more databases:
    - GENOME

  - GENES (by combining various sources (e.g. GenBank, NCBI RefSeq, EMBL and organism-specific databases) followed by internal re-annotation)
  - SSDB (computed sequence similarities, plus ortholog and paralog cluster analysis)
  - DGENES and EGENES (supplementary genes catalogs)

- Systems Information, with 6 databases:
  - PATHWAY (manually drawn general pathway maps representing important biological functions. By filtering the presented nodes by the organism of origin, organism-specific pathways can be automatically produced. The metabolic pathways are described using the KEGG Markup Language (KGML) specification of the pathway graph objects)
  - BRITE (functional hierarchies)
  - MODULE (pathway modules)
  - DISEASE (lists of disease genes and interacting molecules)
  - DRUG (chemical structures of prescription drugs, and molecular networks regarding drug pathways and chemical modifications)
  - ORTHOLOGY (orthology groups by computational analysis and manual curation, which is an overall replacement of the EC numbers. Each entry has an individual K number)

The contents of the systems information databases are manual curated and are presented in three formats, depending on the database: as graphs (e.g. pathways), as simple lists (e.g. diseases) or as hierarchical lists (e.g. functions). Pathway maps are graphs that describe molecular networks, where nodes can represent entities such as genes, gene products, orthologs, metabolites etc., and edges that can represent relations and interactions between the nodes. Simple lists are used to define memberships when no further networking information is available or to describe functional units. Hierarchical lists are used to denote structured memberships, as in the case of ontologies (e.g. the BRITE database) or the orthologies (KO database). The pathway maps, for instance, are classified through the KEGG orthology, having six entries at the top level:

- Metabolism
- Genetic Information Processing
- Environmental Information Processing
- Cellular Processes
- Organismal Systems
- Human Diseases

Each level is further divided to subcategories, where the third level of the hierarchy consists of relevant pathway maps.

Furthermore, the KEGG orthology (KO) hierarchy is included at the third level (under Genes and Proteins → Network hierarchy) of the functional hierarchies and binary relationships of BRITE, where there are five top-level entries:

- Pathways and Ontologies
- Genes and Proteins
- Diseases and Drugs
- Compounds and Reactions
- Organisms and Cells

2.2.2. New Developments in KEGG

2.2.2.1. MEDICUS

KEGG MEDICUS is constructed of assembly of DISEASE and DRUG databases, along with relevant portions from PATHWAY and BRITE databases, developed specifically for the analysis of disease networks. For instance, the relevant pathway maps in MEDICUS contain networks for human diseases and disorders as well as drug metabolism pathways [13].

2.2.2.2. ATLAS

The combination of about 120 basic metabolic pathways residing in KEGG PATHWAY facilitated the creation of a global metabolism map. Central to the construction of the map is the notion of ‘net-elements’, which are manually identified isolated pathway segments. These elements represented as lines on the map, can overlay information from different pathways of origin as they interconnect various compounds. The total network map is created as an SVG file [14].

For perusing the map, a web application has been developed, where the user can see selected parts in higher resolution and, through links, can be directed to the relevant pathway-entry and its compounds.

One interesting feature of the global viewer is the capability to highlight pathways that are specific to an organism. This is accomplished by dimming portions of the global map that do not apply to the selected organism. As the genes in the global map are represented through KO identifiers (e.g. orthology), it is feasible to trace organism-specific gene products accordingly. Since January 2012, a three-dimensional viewer of the map has been developed. Furthermore, the application offers the capability of mapping user data on the global map. Depending on the entered list of genes and corresponding preferred colours, the tool can highlight the positions of these genes on the map. This feature is especially suited for entering gene expression or genome sequence data as it reveals the pathways that the given input list is involved in.

2.2.3. Accessing KEGG

2.2.3.1. Web and FTP

KEGG is the major component of the Japanese GenomeNet, which is served by the Kyoto University Bioinformatics Center. The other GenomeNet services, including DBGET and BLAST/FASTA searches, are now primarily developed and used to support KEGG. The official URL for GenomeNet has been modified to http://www.genome.jp/, but the former URL http://www.genome.ad.jp/ will still be available. To download the KEGG data, users may use the GenomeNet FTP site (recently, only after paid subscription).

2.2.3.2. KEGG API

The KEGG API service has become an increasingly popular mode of access. It is the SOAP/WSDL interface to KEGG, enabling users to write their own programs to access, customize, and utilize KEGG.

2.2.4. Issues with KEGG integration in the VPH context

The Kyoto Encyclopedia of Genes and Genomes PATHWAY [15] database is a valuable comprehensive collection of manually curated pathway maps for metabolism, genetic information processing and other functions. Still, the production of simulation capable metabolic networks from KEGG data in order to run in-silico reaction models is a complicated work, regardless of the numerous tools that are distributed for this task. Access to its data is enabled through various options like the KEGG ftp site [16] (for flat files, and recently, only after paid subscription) in combination with KEGG2SBML [17], by accessing already converted models to SBML [18], through the KEGG API [19] (SOAP/WSDL interface), or through KEGG KGMLs [20]. Nevertheless, the in-silico construction of detailed reaction pathways in particular of specific organisms, entails many steps and the use of different software tools. These steps make the integration process more difficult and time consuming, as it requires compatibility among them.

An example of the intricacies involved in the selection of the proper data source, when attempting to integrate KEGG metabolic pathways, is the following:

- already converted SBML [21] models from [18] are usually out-dated,
- the use of KEGG API for this scope requires extra programming,
- handling the flat files is unwieldy,
- the use of KEGG2SBML tool for their conversion to SBML has its own drawbacks since it is operating in Unix-like platforms only,

- and the download of the relevant flat files for the conversion requires the knowledge of the exact URL address and path to the respective directories of the KEGG ftp server.

The KEGG2SBML tool was developed under the ERATO-SORST Kitano Symbiotic Systems Project [22] in order to create SBML models by parsing KEGG data found in the KEGG LIGAND database. This tool was implemented using the Perl programming language in 2004 and managed to accurately transform about 93% [23] of the total metabolic pathway models available in the database at that time. A new version (1.5.0), released in 2008, is capable of exporting SBML models in both Level1 and Level2 formats while also supporting the annotation scheme of CellDesigner [24], which retains the original visual layout of KEGG models. CellDesigner is a powerful tool for designing and simulating SBML pathway models, compliant with the Systems Biology Workbench (SBW) [25]. It has an easy-to-use graphical interface, which provides distinct representation of various entity types through the use of metadata imported as annotations into standard SBML models.

KEGG Markup Language (KGML) files provide graph information that can be used to computationally reproduce and manipulate KEGG pathway maps [12]. Nevertheless, two major issues should be stressed here, concerning the transformation of KEGG metabolic pathways to biochemical models, appropriate for in-silico functional simulation of various aspects of the cellular physiology. First of all, KGML, which is an XML based format specific for data representation, does not include kinetic information (rate laws) for the pertaining reactions. It is oriented in providing illustrations of the pathways, mainly appropriate for visualization purposes, but can also provide graph operation capabilities through the use of appropriate software tools like KGML-ED [26] and KEGGgraph [27]. For dynamic in-silico modelling efforts though, a limitation arises regarding the extent of exploitation of this valuable knowledge source, as KGML model files are inappropriate for simulation purposes. This holds even though KGML models form a compendium of pathways, fully compliant with the traditional biochemical functional characterization. KGML format cannot carry information regarding the stoichiometry and the kinetic properties of the reaction equations of the illustrated pathway, an essential prerequisite for the construction of a simulated biochemical reaction model. Manually adding the kinetic laws in biochemical models represents a time exhaustive effort, depending on the size of the pathways to be modelled, in terms of encompassed reactions, substrates and modifiers.

Secondly, the orientation of the KGML models, residing in the KEGG pathway database mainly for visualization purposes, incurs serious pitfalls regarding the biochemical consistency of the described models, for simulation purposes. In order to attain a nicer graphical layout, in terms of clarity of the graph, many compounds and reactions are included more than once in the same pathway. This is a problem with respect to the biochemical consistency of the model, given the fact that all reactions take place in the same cell compartment, unless modelling a multi-compartmental model, which results in erroneous stoichiometries for the respective reaction network. Also, the existence of ambiguities regarding the naming of the compounds described in a model, which represents the outcome of unresolved inconsistencies in the used nomenclatures, results in problematic definition of the quantitative characteristics of the participating metabolic pools (i.e. overall concentration of an enzyme or compound). This incurs additional flaws and increases the complexity of the model. Unfortunately, all of these problems remain unresolved in the KEGG2SBML approach where the transformed models are still simple mappings of the KEGG database to static SBML models, with no addition of kinetic properties and not any correction regarding their descriptive problems.

Furthermore, when scrutinizing the KGML and SBML transformed through KEGG2SBML versions of KEGG-residing metabolic pathways, differences are found between them with respect to the composition and the stoichiometry of the resulting reaction networks. Indicatively, the SBML version includes fewer reactions than the KGML model and for each reaction it incorporates species such as H2O, H^+, NADH, NAD^+, ATP, ADP, CO_2, and CoA. For simulation purposes, elimination from the network of such compounds which participate in numerous reactions simultaneously, and which pools are considered abundant, is an acceptable model simplification practice [28]. Therefore original KGML files are considered the preferred data source for transformation purposes, and the accurate integration of KEGG metabolic networks.

Later in this work, we will describe the implementation of the KEGGconverter tool, which executes and integrates - for the first time - the following procedures: pathway merging, conversion to SBML format, proper renaming of the species and addition of default kinetic equations, using as sole input the set of selected KGML KEGG pathway files. The resulted output is an SBML model that can be tested in a simulation environment and can be used as a reference point for further improvement and accurate tuning of the model.

2.3. DESCRIBING GENES: THE GENE ONTOLOGY (GO) PROJECT

The initial attempts to describe gene functions through the use of natural language proved inefficient. Since each laboratory could use different methodologies to annotate genes, the ability to assemble data, conduct database searches and assess similarity among genes was becoming even more difficult. As new data were produced, the adaptation of controlled vocabularies for annotating genes became necessary. The enzyme classification (EC) system that is still used today employs a four-digit number scheme, each digit corresponding to a more specific entry in a hierarchical list of available gene functions. However, as even more datasets became available and the need for accurate descriptions increased, the Gene Ontology project, currently the largest relevant resource, was created [29], [30], [31-36].

The use of common syntax in the description of genes and gene products, in tandem with knowledge of the whole repertoire of genes for several organisms, made the comparison of gene functions across species possible. The existence of conserved functions across organisms can help one to infer roles of newly discovered genes and integrate biological knowledge from different databases.

2.3.1. GO Structure

The GO encompasses three aspects: Molecular Function (MF), Biological Process (BP) and Cellular Component (CC). Each can be thought of as the respective answers to the three questions: how, what and where an enzyme operates on. The GO terms (represented as nodes in a graph) that are available for each aspect are interconnected through hierarchical relations (represented as edges). This forms a Directed Acyclic Graph (DAG), where a child node is connected to a parent node by a directed edge pointing to the more general term (the parent). As the terms acquire more specialized meanings they are more distanced from the initial term (the root). When a gene is being annotated by a GO term, it is also being characterized by all of the parents of that term (which is called the true path rule). The edges can indicate four kind of relations: ‘is_a’, ‘part_of’, ‘has_part’ and ‘regulates’. The ‘is_a’ relation is by far the one most frequently used, and there are concurrent versions of the GO (the filtered versions) that does not contain any ‘has_part’ or ‘part_of’ relations. Although more recent versions of the GO introduced relations between the aspects, most computational tools strictly use the ‘is_a’ relation. This is due the fact that the notion of specialization is more general to its use (and relevant to the inheritance idea of the Object Oriented terminology), whereas the depiction of parts applies mostly to the CC aspect.

As biological knowledge regarding the functionalities of genes and gene products is expanding, new terms and new relations are introduced in the GO. Terms that are becoming obsolete retain their ID, but lose the edges connecting to neighbouring terms, i.e. they become orphan nodes.

2.3.2. Evidence codes

Evidence codes characterize the method used to annotate a gene by a GO Term and it is the main indication to assess the validity of an annotation. There are 18 available evidence codes, and they are subdivided into categories according to the decision procedure that has been followed (presented here in order of significance): Experimental, Author Statement, Curator Statement, Computational Analysis and Automatically Assigned.

The IEA (Inferred from Electronic Annotation) evidence code is the only code that indicates the absence of any manual reviewing. Nevertheless, it ascribes for the ~98% of the annotations, which has implications regarding the validity for the majority of the given annotations.

Most of the computational methods used for the characterization of gene function are based on sequence similarity methods. Additionally, structure similarity, protein profiles, phylogenetic relations and artificial intelligent methods can be employed to improve the annotation procedure [37].

2.3.3. GO Similarities

The annotation of gene function through GO terms, regarding a specific aspect, enables the comparison among terms and, consequently, among genes that are characterized by the terms.

2.3.3.1. Similarity between GO terms

At first, simple graph distance was used to describe the distance between two GO terms, i.e. the smallest number of edges (or hops) between them. Another graph metric is to measure the number of common parents. The problem with these graph metrics is that the first assumes that the same number of edges accounts for the same distance, and the second that the distance from the root term is the only measure of specificity.

To compensate for the fact that the terms in the DAG hierarchy do not have equal in-between semantic distances, a notion from information theory came in handy: the notion of the information content (IC) of a GO term (c):

$$\boldsymbol{IC(c) \coloneqq -log\big(p(c)\big)}, \qquad (1).$$

where p(c) is the probability of having c in a specific corpus. The probability is computed as “the number of annotation instances for the term divided by the total number of annotation instances from the annotation database” [38]. Using this formula, the less frequent a term appears, the more information content it carries.

Resnik [74], using the information content idea with the ontology graph, defined the similarity between two terms c_1 and c_2 as:

$$\boldsymbol{SimResnik(c_{1\,,}c_2) \coloneqq max_{c\in S(c_{1,}c_2)} IC(c)}, \qquad (2).$$

where $S(c_{1,}c_2)$ is the set of all terms that subsume (i.e. are ancestors of) both c_1 and c_2. As the first common parent usually is the Most Informative Common Ancestor

(MICA), this can be used to speed up the computation of Resnik similarities. A normalized version of Resnik's formula has been given by Lin, delimiting the possible values between 0 (no similarity) and 1 (identical):

$$\boldsymbol{SimLin(c_{1,}c_2) := \frac{2 * SimResnik(c_1, c_2)}{IC(1) + IC(c_2)}} \quad (3).$$

As each similarity formula has its limitations, many other methods have been proposed to account for the bias that each metric encounters. The fact that specific areas of the GO DAG have been more actively developed, though, introduces an inherent bias to the IC approaches [29].

2.3.3.2. Similarity between genes

Based on definitions of GO terms similarity, we can move on to definitions for similarities between genes. For instance, the average or the maximum similarity between all possible GO-term pairs can be used:

$$\boldsymbol{GeneSim_{avg}(G_{1,}G_2) \coloneqq avg_{c_1 \in T(G_1), c_2 \in T(G_2)} Sim(c_1, c_2} \quad (4).$$

or

$$\boldsymbol{GeneSim_{max}(G_{1,}G_2) \coloneqq max_{c_1 \in T(G_1), c_2 \in T(G_2)} Sim(c_1, c_2)} \quad (5).$$

Where T(G) is the set of GO terms associated with gene G.

Again, each method has its limitation and other approaches have been suggested, such as the notion of the Term Overlap (counting the number of common terms) or the Jaccard index (the ratio of the size of terms intersection to the size of the terms union).

2.3.4. Use of GO for genes prioritization

The classical approach for the selection of genes with a pivotal role regarding disease onset and evolution emphasizes a firmly described functional context in single genes or specific collinear pathways, whose actions are systematically examined within. Ultimately, this is reflected in the structure and emanates from the recognition of the topological properties of the underlying active biological networks [39]. The scale-free property, as emerged from studies of co-expression networks across species, confirms the fact that the majority of nodes, being part of a network representation of the underlying cellular biochemical pathways, are connected to relatively few neighbours. Additionally, relatively few nodes are highly connected to many others, giving rise to the concept of hub nodes that potentially represent key information control points in the network [40].

The utilization of the information residing in standard biological vocabularies like GO, confers a pathway-centric perspective in the interpretation of results derived from –omic technologies, like DNA microarrays or Next-Generation Sequencing. This approach mitigates the hazard of false positives infiltrating the interpretation procedure and strengthens the biological plausibility of the results. However, the process of uncovering individual molecular actors (genes, proteins, etc.) for targeted biological validation, through a generic, principled, systematic framework, still remains elusive. The biological validation process is obstructed as no specific molecular players for wet-lab validation are highlighted. This ultimately results in huge volumes of tedious biological experimentation. Several identification algorithms have been proposed to assist in the task of the automated selection or prioritization of candidate genes, making use of different biological data sources and having generic or specific disease targeting scopes in mind tools [41, 42], [43-46].

The functional annotation of complete genomes for an ever-growing number of organisms renders possible the assessment of their relevance by exploiting their mappings in the GO tree structure. In this way, a subset of selected molecular entities, derived from a high-throughput experimental dataset, or an established molecular list reliably relating to a specific physiological state together with their respective GO terms can act as a starting point for a systems level exploration of their interrelations [47]. This represents an alternative approach to the bi-graph representation of biological information of genes [48].

Reverse engineering the pivotal role of certain molecular actors, as derived from a holistic view of their involvement in numerous cellular procedures simultaneously, would impart a systems-level understanding in the clarification of the cellular physiology. It would also help the identification of functional couplings and bottlenecks, among procedures, which are considered so far unrelated and examined in isolation from all the rest [49].

The GOrevenge algorithm described in the next chapter is a new stepwise biphasic mechanism, which utilizes the GO annotations and its hierarchy. It can be applied for instance, after the completion of the analysis of a microarray experiment with respect to its differential expression profile and its relevant enrichment analysis. Having derived a set of significantly differentially expressed genes corresponding to a set of significantly enriched GO terms, the algorithm utilizes the duality of the mapping between genes and GO terms to highlight genes, which may have evaded the initial significant gene list. These omissions may have occurred because of numerous reasons, such as the lack of statistical power due to small replicate numbers, despite them showing considerable participation in multiple molecular functions and biological processes. Moreover, this is based on objective measures of their effects in the cellular physiology, naturally yielding a performance score, which leads to a prioritized discovery process, exploiting the underlying GO tree topology.

2.4. ARCHIVING GENE EXPRESSIONS: THE GENE EXPRESSION OMNIBUS (GEO) DATABASE

GEO is one of the largest data repositories regarding gene expression datasets, especially data derived from microarray experiments. It was initiated by the National Center for Biotechnology Information (NCBI) in 2000, and it is used by the scientific community to submit results from laboratory projects. Until now, GEO hosts results from about 20.000 studies, which supported publication of over 10.000 manuscripts, covering a wide variety of biological phenomena.

As the data residing in GEO continue to grow, scientists have access to this information source using the same common interface and processing tools. Although the diversity of experiments, instruments and methods is getting even wider, the target of study might be quite similar in many cases (same biological functions). Thus, GEO offers the leveraging of existing data for creating and validating new hypotheses, utilizing different sources, views and modes on a larger scale [50-53].

2.4.1. Organization of the Database

In order to characterize the data submitted to the repository, four sections of descriptions are available. The Platform section (carrying a unique GPL id) defines the array that was used in the experiment. For each feature in the array, there are links for the descriptions of the relative gene-tags. The Sample section (GSM id) contains the measured data along with acquisition procedures. The Series section (GSE id) groups the Samples of an experiment, and finally there is the option of submitting the attained raw data. The most useful feature of the GEO, though, is the DataSet section (GDS id), where a higher information component is assembled by curation for each experiment. There, a pre-processed dataset is available, along with the results organized per feature subgroup.

The DataSet section offers two perspectives of the data: The first provides the assimilation of the experimental results as a whole, together with graphs and analysis tools, and the second, a gene-centred view of the results, where the expression behaviour of a specific gene is displayed across all of the experimental conditions. The last perspective is called GEO Profile and it is best suited for exploration and verification of the behaviour for specific targeted genes.

2.4.2. Accessing GEO Data

GEO data can be retrieved either through the http protocol or through ftp. The first offers user-friendly interface and utilities for searching by specifying parameters of interest; the second is destined for bulk downloads. The GEO data can be queried using the Entrez search engine, which also accesses 23 other NCBI databases.

In dealing with huge amounts of genomic and proteomic data, it is of crucial importance to the researcher to easily pinpoint specific regions of interest and retrieve information regarding only the task at hand. GEO http interface provides an arsenal of query mechanisms towards this goal. Apart from searching by nucleotide sequences, gene names, platforms and arbitrary text context, it is feasible to apply Boolean operators to further qualify the results. The search results can be directed either to retrieve relevant experiments (Datasets), or toward specific genes (Profiles).

2.4.3. GEO Tools

When the user narrows his or her list of retrieved genes to those complying with the particular entered parameters, GEO also proposes other genes that show similar behaviour. The similarity can be based on:

1. Profile (i.e. correlating gene expression patterns)
2. Sequence (BLAST application)
3. Homology (through the HomoloGene application)

Two more tools exist in the GEO apparatus: Group comparisons and Heat Maps. Since the usual target of a study is the determination of differentially expressed genes, the user can select subsets of a DataSet and compare two groups of experimental modes with each other. Heat Maps cluster the x-axis (samples) and the y-axis (genes) independently, using multiple clustering algorithms. The colour assigned to each cell is dependent on the expression level; thus, the patterns that appear on the heat-map indicate groups of similar response.

2.4.4. Submission Formats

There are many ways for submitting experimental data to GEO, but in each case the annotation of the contents should be according to the Minimum information about a microarray experiment - MIAME [54] checklist. A list of upload format options includes: Web deposit (for small datasets), the Simple Omnibus Format (SOFTtext or SOFTmatrix), the MINiML format (MIAME Notation in Markup Language), and the MIAGE-ML format (MicroArray Gene Expression Markup Language).

2.5. INTEGRATING SOURCES: FUSION OF MULTI-MODAL DATA

Integration of multi-modal and multi-scale data is of known importance in the context of personalized medicine and electronic health records. Much effort is being put into assessing the appropriate data fusion schemes that could utilize most of the available information contained in these datasets. In the context of the VPH vision, an integrated framework should make it possible to interconnect predictive models defined at different scales, with different methods, and with different levels of detail, into systemic networks that solidify systemic hypotheses [55].

Information-fusing algorithms can be categorized as being either combination of data (COD) or combination of interpretations (COI) [56]. COD methods aggregate features from each source into a single feature vector before classification, while COI methods classify the data from each source independently and then aggregate the results. Rohlfing et al [56] compared the two methods to combine information sources in different biomedical image analysis applications, while Haapanen and Tuominen [57] followed a COD approach for the combination of satellite image and aerial photograph features for forest variable estimation. On the other hand, Jesneck et al [58], on a COI path, optimized a decision-fusion technique to combine heterogeneous breast cancer data. Lee et al [59] proposed a Generalized Fusion Framework (GFF) for homogenous data representation and subsequent fusion in the meta-space using dimensionality reduction techniques. The meta-space is created by projecting the heterogeneous data streams into a space where these scale and dimensionality differences are alleviated. Such meta-space representation approaches, which transform data into a homogeneous space allowing for direct combination of modalities, are embedding projections and kernel space projections [60].

GFF algorithms assume that we have raw data from sources $\mathrm{S_i}(\mathrm{x}_1, \mathrm{x}_2, .., \mathrm{x_k})$, where $\mathrm{x}_1, \mathrm{x}_2, .., \mathrm{x_k}$ represent the k observations in a study and i represents one of the N data sources, where $i \in \{1,2,...,N\}$. While this could be the case for specific studies or electronic patient records, most available databases contain single modal data from different patients. The number of observations from each source S_i differs. Nonetheless, the modalities from each source reflect information regarding the same disease, and it is an open question as to how these interconnections could be exploited.

In this work, we study the behaviour of a feature selection algorithm (random forests) as obtained by fusing multi-modal data from different subjects. We refer to these data as *separate* datasets. The integration of separate datasets referring to the same disease is an innovative approach that can contribute significantly towards the extraction of better biomarkers involved in various diseases.

2.6. TARGETING A COMPLEX DISEASE: CUTANEOUS MELANOMA

Cutaneous melanoma (CM) is considered a complex multigenic and multifactorial disease that involves both environmental and genetic factors. It is the most life-threatening neoplasm of the skin, and its incidence and mortality have been increasing worldwide. CM tumorigenesis is often explained as a progressive transformation of normal melanocytes to nevi that subsequently develop into primary cutaneous melanomas (PCM). However, the molecular pathways involved have not been clearly elucidated, although considerable progress has been made [61]. Despite the success of genomics in defining genomic markers or gene signatures for other kinds of cancers (such as breast cancer), there has been no similar progress related to malignant melanoma.

The microarray studies that have been performed on CM by different groups used different microarray platforms in highly heterogeneous patient cohorts and pathological sample collections [62]. These differences make comparisons quite difficult and result in a reduced total cohort size and diversity, since independent cohorts from different studies are hard to sum [63].

Regarding the clinical diagnostic methods for melanoma, there are several standard approaches for analysis and diagnosis of lesions. Some of them are: the

Menzies scale, the Seven-point scale, the Total Dermoscopy Score based on the ABCD rule, and the ABCDE rule (Asymmetry, Border, Colour, Diameter, Evolution). In these methods, digital images can serve as a basis for the medical analysis and diagnosis of lesions under consideration. As there is a general lack of precision in human interpretation of image content, advanced computerized techniques can assist doctors in the diagnostic process [64]. A review of image acquisition and feature extraction methods utilized in the literature regarding existing classification systems can be found in [65].

2.7. EMPLOYING ARTIFICIAL INTELLIGENCE: FEATURE SELECTION WITH RANDOM FORESTS

Feature selection techniques do not alter the original representation of the variables but merely select a subset of them, in contrast to other dimensionality reduction techniques like those based on projection (e.g. principal components analysis) or compression (e.g. using information theory). Thus, they preserve the original semantics of the variables, offering the advantage of interpretability by a domain expert [66].

The main objectives of feature selection are a) to avoid overfitting and improve model performance, b) to provide faster and more cost-effective models and c) to gain a deeper insight into the underlying processes that generated the data.

Regarding the used feature selection procedure in this study, a wrapper type technique was applied (sequential backward elimination) using the random forest algorithm [67], which utilizes ensembles of decision trees. In addition, a multivariate filter was used as an option to reduce the co-linearity among features of the microarray dataset, prior to the application of the wrapper method. This filtering, together with the imputation procedure, is a departure from a merely COD method, towards a GFF approach, although here no further transformation is applied to the feature vectors.

The random forest algorithm, among other ensemble learning methods, is reported to be successful in variance reduction which is associated with overfitting [68]. In addition, we utilized the option of stratifying the bootstrapped samples with an equal number of cases per class [69]. This is compatible with the Balanced Random Forest (BRF) approach, which is computationally more efficient with large imbalanced data, since each tree uses only a small portion of the training set to grow. Additionally, BRF is less vulnerable to noise (mislabelled class) than the Weighted Random Forest (WRF) were a heavier penalty is placed on misclassifying the minority class [70]. BRF alleviated the class imbalance problem, which is a common problem in disease diagnosis where the disease cases are rare as compared with normal populations. The recognition goal is to detect people with the disease; thus, a favourable classification model is one that provides a higher identification rate on the disease category.

CHAPTER 3. MATERIAL AND METHODS

In the context of dealing with the research questions of this study, that is the creation of tools and methodologies which contribute to the creation of integrated physiological models, three novel approaches has been developed targeting at different levels of biomedical modelling. At the metabolic pathways level the KEGGconverter tool, at the ontology level the GOrevenge algorithm, and lastly a new method of integrating multimodal biomedical data. Each of these approaches is described next.

3.1. KEGGCONVERTER

KEGGconverter is a de novo developed software application, implementing a novel algorithmic workflow (Figure 3) to achieve the proper transformation of the set of original KGML files to be fused together, in a SBML file, using a combination of XSL and Java procedures. The three distinct phases are: KGML merging, conversion to SBML, and the addition of kinetics.

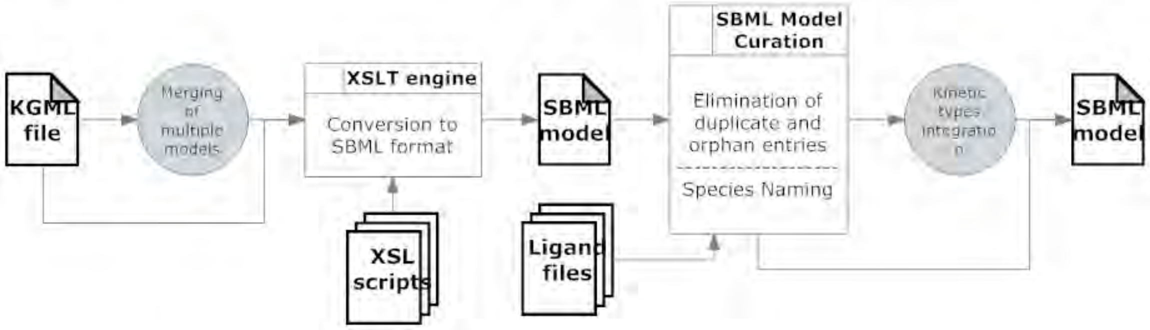

KGML file
Merging of multiple models
XSLT engine
Conversion to SBML format
XSL scripts
SBML model
Ligand files
SBML Model Curation
Elimination of duplicate and orphan entries
Species Naming
Kinetic types integratio n
SBML model

Figure 3: KEGGconverter: KGML to SBML conversion stages

3.1.1. Merging KGML pathways

In order to enable the fusion of many interrelated KEGG pathways, for the scope of building system-level models, an algorithm was implemented to merge selected KGML models. This entails the concatenation into a single file, of all the entries, relations and reactions elements from the selected pathways. During this concatenation, the ‘id’ attribute values from the <entry> elements and the ‘entry1-2’ attribute values form the <relation> elements are modified, so as to incorporate an indication of the file number they have originated from. In addition, during the concatenation process, <relation> elements are removed, which target attribute (‘entry2’) is one of the pathways that are to be merged. Thus, discrepancies in the connecting arrows of the resulting integrated pathway are avoided, as these will not be directed (map links) to the already incorporated pathways in the model.

3.1.1.1. Converting to SBML

Subsequently the conversion of the single file derived from the previous step to a SBML file, is handled within two stages, firstly a combination of XSL transformations and then DOM processing through Java programming. This procedure is depicted at Figure 3.

At first, the elements of <entry> group in the KGML file are forming the <species> group in the SBML format including compounds, genes or enzymes and neighbouring pathways. Accordingly, the <reaction> elements in SBML are created, using residing information in both <reaction> and <relation> elements of the KGML input file, which specify how the compounds of the network interconnect with each other. Specifically, the <reaction> elements in KGML define the reactants and products, while modifiers are defined in <relation> elements, which link to certain enzymes in the <entry> group, as well as to anchor points of the neighbouring pathways.

A core idea in this stage is the conversion of the <relation> KGML elements, which point to other pathways, to additional SBML reactions. By this way, the map link information of the KGML pathway files is retained during this processing. If the map link points at an included pathway, the redundant SBML ‘pathway’ <species> entry will be eliminated at the second stage of the conversion. Otherwise, these entries will be retained at the SBML file, considered as conventional reactions in the final simulation, to indicate group reaction flows towards the neighbouring pathways, and thus providing more accurate constraining, with respect to the putative biochemical cross-talks of the integrated pathway model with neighbouring pathways, which have not been fused in the final model. Of course these reactions, having distinct ‘id’ and ‘name’ values from the rest of KEGG reactions, are fully identifiable and can thus be easily modified, erased or included fully or partially to the model from the resulting SBML file. Optionally an XSL file is created, which removes completely these reactions, in case the user of the tool selects so.

After the initial XSL transformation stage, the resulting file is already complying with the SBML format, and can be read and edited by every tool supporting this format.

However, there are two main problems with respect the thus created SBML models, namely the nomenclature adopted for the species, and the redundancy of both species and reactions. Utilizing the source information from KEGG database, all the species of the model are named according to their enzyme commission numbers (EC) or its KEGG number such as: ‘ec:2.3.1.180’ and ‘cpd:C00229’, instead of informative biochemical names. This clearly represents an essential limitation with respect to their comprehension and naturally the correct curation of the model by the experts. Responsible for the species and relation entries redundancy is the use of multiple graphical instances (nodes for species and arrows for relations) of these elements for the purpose of neat graphical representations of the KEGG pathways, as well as due the previous merging step, during which multiple pathway xml files are integrated in one file and thus their particular species and relation entries are added to the final file, without been checked if they refer to the same species (biochemical compounds) or relations (reactions), which take place at the same cellular compartment.

In order to eliminate these problems the model is further processed through the use of Java procedures, which utilize the Xerces XML parser and the DOM. The curation of the models follows two distinct steps:

3.1.1.2. Elimination of duplicate and orphan entries

At this stage, all species are checked based on their KEGG name. If the same name is found in more than one species, then all these species take as 'id' value the first value found. An example can be seen at Figure 4 where an example of two different reactions is given, each having one reactant, one product and one modifier. In the case that the modifier s5 is the same with the s6, the first will replace the second in its reaction.

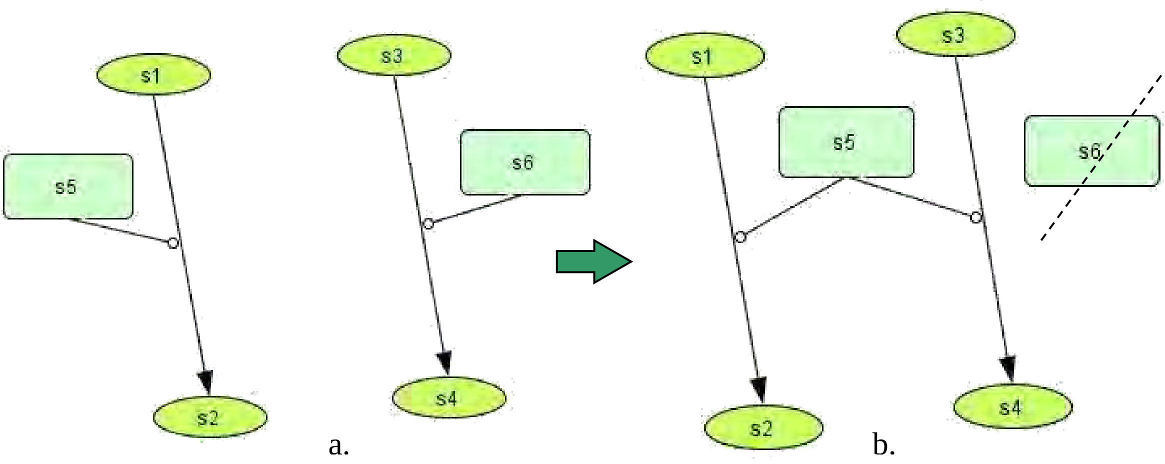


Figure 4: a. Initial state of two reactions with the modifiers s5 and s6 having different ids but representing the same entity (by having the same name). b. The final state, where s5 is defined as modifier for the second reaction too, and s6 is erased.

The same method is applied also for duplicate <reaction> elements where all their attributes – such as name, reactants, products and modifiers – are the same. This usually happens when fusing more than one neighbouring pathway models, which have overlapping sections.

Performing this substitution however, deprives some species completely of their connection to any reaction. These “orphan” nodes are eliminated, together with other already “orphan” nodes from the initial model (often in KEGG pathways appear metabolites and enzymes, with not any defined connection to the presented reaction network).

3.1.1.3. Naming <species>

After the removal from the model of all the unnecessary entries, renaming of the species according to the KEGG database nomenclature follows. Although it is possible to retrieve the name of species by submitting queries directly to the dbget system (e.g. for instance for the ec:1.3.1.9 name http://www.genome.jp/dbget-bin/www_bget? enzyme+ 1.3.1.9), the solution adopted here was instead that of parsing selected flat files from the KEGG database and creating four tab delimited files with all the necessary information. The reason for that was the time profit thus attained, as the time needed to parse a tab delimited file and load it into memory, is far less than this needed to submit massively, internet queries. The four tab delimited files are: Compounds.tab, EnzymeNames.tab, GlycanNames.tab and Enzyme-Organism.tab.

During this step, the values of the ‘name’ attribute of the <species> elements are being searched through the appropriate .tab file for a match. In order to decide if the ID in question is a compound, enzyme, glycan or gene, the prefix of the ID is examined. If the name starts with the prefix “cpd:” it refers to a compound, “glycan:” is for a Glycan, “ec:” refers to Enzyme and if it has organism id followed by “:” it’s a gene. For example “hsa:893” it’s the human ortholog gene with index 893, (homo sapiens = hsa). In reference pathways the reactions are catalysed by enzymes, while in organism specific pathways, genes are shown as reaction modifiers. Using the

Enzyme-Organism.tab, we can correlate gene ortholog for the specific organism, with the enzyme that it encodes and both are assigned as a name to the entity. In many cases though, an enzyme is encoded by more than one gene, but only the first of them found in the database is included in this renaming procedure, so as to avoid extremely long nametags.

As an example of the KGML model of a pathway, two parts of the human Glycolysis/Gluconeogenesis metabolic model (with id hsa00010) are shown next:

At first the definition of the pathway and two <entry> tags:

```
<?xml version="1.0"?>
<!DOCTYPE pathway SYSTEM "http://www.genome.jp/kegg/xml/KGML_v0.6.1_.dtd">
<!-- Creation date: Jan 15 2009 00:59:40 +0900 (JST) -->
<pathway name="path:hsa00010" org="hsa" number="00010"
        title="Glycolysis / Gluconeogenesis"
        image="http://www.genome.jp/kegg/pathway/hsa/hsa00010.gif"
        link="http://www.genome.jp/dbget-bin/show_pathway?hsa00010">
    <entry id="1" name="hsa:217 hsa:218 hsa:219 hsa:220 hsa:223 hsa:224
             hsa:501" type="gene" reaction="rn:R00710"
                    link="http://www.genome.jp/dbget-
             bin/www_bget?hsa+217+218+219+220+223+224+501">
        <graphics name="ALDH2..." fgcolor="#000000" bgcolor="#BFFFBF"
             type="rectangle" x="282" y="950" width="46" height="17"/>
    </entry>
    <entry id="2" name="hsa:55902 hsa:84532" type="gene" reaction="rn:R00235"
        link="http://www.genome.jp/dbget-bin/www_bget?hsa+55902+84532">
        <graphics name="ACSS2..." fgcolor="#000000" bgcolor="#BFFFBF"
             type="rectangle" x="139" y="918" width="46" height="17"/>
                                        …
```

then follows a sample of the definitions of two reversible reactions, and the closing <pathway> tag:

```
          <reaction name="rn:R00710" type="reversible">
        <substrate name="cpd:C00084"/>
        <product name="cpd:C00033"/>
    </reaction>
    <reaction name="rn:R00235" type="reversible">
        <substrate name="cpd:C00033"/>
        <product name="cpd:C00024"/>
    </reaction>
</pathway>
```

The conversion to SBML of this metabolic pathway using KEGGconverter results to the respective model, which adheres to the level 2, version 1 of the SBML specification. The start of the converted model is shown next:

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?><sbml
xmlns="http://www.sbml.org/sbml/level2" level="2" version="1">
<model id="model_Glycolysis___Gluconeogenesis">
        <listOfCompartments>
           <compartment id="default" name="default"/>
           <compartment id="uVol" name="uVol"/>
        </listOfCompartments>
        <listOfSpecies>
           <species compartment="uVol" id="s1" initialAmount="1.0" name="aldehyde dehydrogenase
           (NAD+) [1.2.1.3] [hsa:217]"/>
           <species compartment="uVol" id="s2" initialAmount="1.0" name="acetate---CoA ligase
           [6.2.1.1] [hsa:55902]"/>
           <species compartment="uVol" id="s3" initialAmount="1.0" name="aldehyde dehydrogenase
           (NAD+) [1.2.1.3] [hsa:218]"/>
           <species compartment="uVol" id="s4" initialAmount="1.0" name="Acetate"/>
           <species compartment="uVol" id="s5" initialAmount="1.0" name="Pentose phosphate
           pathway"/>
                                                        …
```

At the declaration of the <species> the human readable names are apparent, along with the original ‘hsa:xx’ coding of the KGML file. New ids have been created, and a default value for the ‘initialAmount’ has been given to each specie. Later, the definitions of the reactions follow:

```
<reaction id="R00235" name="R00235" reversible="true">
        <listOfReactants>
           <speciesReference species="s4"/>
        </listOfReactants>
        <listOfProducts>
           <speciesReference species="s67"/>
        </listOfProducts>
        <listOfModifiers>
           <modifierSpeciesReference species="s2"/>
        </listOfModifiers>
</reaction>
<reaction id="R02569" name="R02569" reversible="true">
        <listOfReactants>
           <speciesReference species="s70"/>
           <speciesReference species="s67"/>
        </listOfReactants>
        <listOfProducts>
           <speciesReference species="s69"/>
        </listOfProducts>
        <listOfModifiers>
           <modifierSpeciesReference species="s13"/>
        </listOfModifiers>
</reaction>
```

Here, we have two reversible reactions: the first is of type 1-1-1, i.e.: a reaction having one reactant, one product and one modifier, and the second of type 2-1-1. We note that there are no definitions of kinetic equations in this step up to now, since KGML does not include such information in its specification

3.1.2. Massive Introduction of Kinetic Information

The final phase to the derivation of a fully capable pathway model for dynamic simulation purposes is the addition of the layer of kinetic equations. For this purpose, we used a proprietary kinetic library, fruit of our previous development effort [71], as a plug-in on SBMLeditor [72]. This library includes 17 types of kinetic equations for reversible reactions and 15 for non-reversible ones, but is customized to include also user-defined kinetic mechanisms. By exploiting the reaction type (reversible or non-reversible) and the stoichiometric relations of each reaction's reactants, products and modifiers, KEGGconverter enables the automated introduction of 4 case-specific default kinetic mechanisms in the model, according to the following rules:

- If a reaction is of 1-1-1 type (which means: 1 reactant, 1 product, 1 modifier) then the hyperbolic modifier rate-law is being added. If the reaction is reversible, the reversible hyperbolic modifier equation is being added.

- For all the remaining types of reactions, the mass action rate law is being added (appropriately modified for the reversible or irreversible cases). If there are modifiers in the reaction they are retained in the SBML derived file, although the mass action mechanism does not foresee modifiers, so that no loss of information occurs. In this case, it is left upon the user to decide later, how to tackle each specific case.

Part of the definitions of the kinetic equations is shown next.

```
true,Hyperbolic_modifier,7,Kms,Kmp,Vf,Vr,Kd,a,b,1,0,M
(Vf*S/Kms-Vr*P/Kmp)*(1+b*M/(Kd*a))/(1+M/Kd+(S/Kms+P/Kmp)*(1+M/(Kd*a)))
\(\w+\*(\w+\/\w+)\-\w+\*(\w+\/\w+)\)\*\(1\+\w+\*((\w+))\/\((\w+\*\w+)\)\)\/\(1\+\3\/\w+\+\(\1\+\2\)\*\(1\+\3\/\(\5\)\)\)
false,Hyperbolic modifier (irr),5,Km,V,Kd,a,b,1,0,M
V*S/Km*(1+b*M/(Kd*a))/(1+M/Kd+S/Km*(1+M/(Kd*a)))
\b\w+\*(\w+)/(\w+)\*\(1\+\w+\*((\w+))/\((\w+)\*(\w+)\)\)/\(1\+\3/\5\+\1/\2\*\(1\+\3/\(\5\*\6\)\)\)
```

Lines 1 and 4 denote the name, the type and the used constants of the reactions (here is the hyperbolic modifier rate-law for the two cases: reversible and irreversible). At the lines 2 and 5 are the ascii representation of the equations which are later converted to MathML in order to be inserted to the created SBML model, and at lines 3 and 6 are the regular expression patterns used to identify the type of equation, and the place where variables having the proper stoichiometry are to be inserted. The advantage of having these definitions in a text/ascii file, where the java executable program of KEGGconverter opens and reads, is that advanced users can easily modify the equation’s parameters, without the need to modify the source java code and recompile the whole application. In Appendix 1 code for the java procedures of KEGGconverter is shown.

An example of the implemented insertion of the MathML for the reversible case of the hyperbolic modifier rate-law, for the reaction with id=’R00235’, is shown next:

```
<reaction id="R00235" name="R00235" reversible="true">
<listOfReactants>
<speciesReference species="s0_4"/>
</listOfReactants>
<listOfProducts>
<speciesReference species="s0_67"/>
</listOfProducts>
<listOfModifiers>
<modifierSpeciesReference species="s0_2"/>
</listOfModifiers>
<kineticLaw><math xmlns="http://www.w3.org/1998/Math/MathML">
  <apply>
    <divide/>
    <apply>
      <times/>
      <apply>
        <minus/>
        <apply>
          <divide/>
          <apply>
            <times/>
            <ci> Vf </ci>
            <ci> s0_4 </ci>
          </apply>
          <ci> Kms </ci>
        </apply>
        <apply>
          <divide/>
      <apply>
        <times/>
        <ci> Vr </ci>
        <ci> s0_67 </ci>
      </apply>
      <ci> Kmp </ci>
    </apply>
  </apply>
  <apply>
    <plus/>
    <cn type="integer"> 1 </cn>
    <apply>
      <divide/>
      <apply>
        <times/>
        <ci> b </ci>
        <ci> s0_2 </ci>
      </apply>
      <apply>
        <times/>
        <ci> Kd </ci>
        <ci> a </ci>
      </apply>
    </apply>
  </apply>
</apply>
<apply>
  <plus/>
  <cn type="integer"> 1 </cn>
  <apply>
    <divide/>
    <ci> s0_2 </ci>
    <ci> Kd </ci>
  </apply>
  <apply>
    <times/>
    <apply>
      <plus/>
      <apply>
        <divide/>
        <ci> s0_4 </ci>
        <ci> Kms </ci>
      </apply>
      <apply>
        <divide/>
        <ci> s0_67 </ci>
        <ci> Kmp </ci>
      </apply>
    </apply>
    <apply>
      <plus/>
      <cn type="integer"> 1 </cn>
      <apply>
        <divide/>
        <ci> s0_2 </ci>
        <apply>
          <times/>
          <ci> Kd </ci>
          <ci> a </ci>
        </apply>
      </apply>
    </apply>
  </apply>
</apply>
```

```
  </apply>
</math><listOfParameters><parameter id="Kms" name="Kms" value="1"/><parameter id="Kmp"
name="Kmp" value="1"/><parameter id="Vf" name="Vf" value="1"/><parameter id="Vr" name="Vr"
value="1"/><parameter id="Kd" name="Kd" value="1"/><parameter id="a" name="a"
value="1"/><parameter id="b" name="b" value="1"/></listOfParameters></kineticLaw></reaction>
```

3.1.3. Availability

The implemented tool is freely available to the academic community as a web application as well as a java .jar command line tool. The user can select the KEGG pathways he/she wishes to process and apply to them a selection of the following:

- merging to a single KGML model,
- conversion to SBML format,
- conversion to SBML plus the addition of default kinetic equations.

3.1.4. Web Application

In the case of the Web based version of KEGGconverter, users can access it by the homepage. In this way a user-friendly shell is wrapping the inner algorithms of the application thus ensuring easy access and manipulations to its content, as the whole process is transparently and seamlessly performed. On the introductory page the user can set all the necessary properties for performing the conversion.

3.2. GOREVENGE

As a means of considering potential kinship among genes we introduce GOrevenge, a new stepwise, biphasic mechanism, starting from the initial considered gene set and their corresponding GO terms. In the first (agglomeration) phase, genes are collected not only when linked to a given GO term, but also to its neighbouring ones, i.e. its parents and children GO terms. Then, for genes that are annotated by several terms, we consider a pruning phase, where GO terms are eliminated depending on the in-between distance of the terms. For instance, if a gene is annotated by a GO term and its child, we eliminate the parent relation and we keep only the more specific term in distance 1. Previously [73], a preliminary version of the algorithm was presented, which examined the effect of mastic oil treatment in Lewis lung carcinoma mice cells.

GOrevenge incorporates Resnik [74] semantic similarity metrics, in addition to edge-based graph distances. All previous code has been rewritten from R [75] to Python [76],[77] for greater versatility and the relevant modules have been integrated into a web application using the web2py web framework [78]. The ability to probe specific categories of the GO (i.e. Molecular Function-MF, Biological Process-BP, Cellular Component-CC) for specific organisms enabled the generic application and evaluation of the algorithm in a case of a human pancreatic cancer gene set. Python code of the GOrevenge is shown in Appendix 2.

For the retrieval of neighbouring GO terms two different approaches were adopted: An edge-based approach where the edges in the graph between two terms are counted (Graph-BubbleGO and Graph-Pruning methods), and a node-based approach [79] utilizing the information content (IC) of a term (Resnik-BubbleGO and Resnik-Pruning methods). Python functions BubbleGO and BubbleGenes are presented next:

```
#==========================================================================
# BubbleGO
#==========================================================================
def BubbleGO(sem_gos, go_genes, relax=-1, resnik='', t2p=''):
    """
    returns a list of dictionaries, with ranked genes according to the number of gos
        [{'count': 2, 'gene': 'CARS', 'gos': set(['00049', '46872'])},
        {'count': 1, 'gene': 'SMAP2', 'gos': set(['46872'])}, ..]
      relax = -1: no relax, 0:gets all the descendants of each term, 0-1
    """
    mygos=[]
    if resnik:
        mygos = myrelax2(sem_gos, resnik=resnik, relax=relax )
    elif t2p:
        parent2term=defaultdict(set)
        [parent2term[j].add(i) for i in t2p for j in t2p[i] ]
        mygos = myrelax_graph(sem_gos, t2p, p2t=parent2term, relax=relax )

    my_go_genes = genes_from_gos(mygos, go_genes)
    rank = rank_genes(my_go_genes)
    return rank

#==========================================================================
# BubbleGenes
#==========================================================================
def BubbleGenes(sem_genes, go_genes, relax=-1, resnik='', t2p=''):
    """
    returns a list of dictionaries, with ranked genes according to the number of gos
        [{'count': 2, 'gene': 'CARS', 'gos': set(['00049', '46872'])}, {'count': 1, 'gene': 'SMAP2', 'gos': set(['46872'])}, ..]
     relax = -1: no relax, 0:gets all the descendants of each term, 0-1
    """
    gene_gos=defaultdict(set)
    [gene_gos[gene].add(go) for go in go_genes for gene in go_genes[go]] #make gene-go dict

    sem_gos = set()
    for gene in sem_genes:
        sem_gos.update(gene_gos[gene])
    rank = BubbleGO(list(sem_gos), go_genes, relax=relax, resnik=resnik, t2p=t2p)
    return rank
```

In general, genes are related to ontological terms, which are part of a given hierarchical structure. Spanning the tree structure of the terms from the root node downstream, concepts are described in more detail. This enables the formation of clauses while at the same time narrowing the number of related genes to those terms. Implicitly, it is assumed that the smaller the graph distance, the higher the similarity (kinship) of terms and the related genes. If a specific physiology is targeted, described by a given group of terms, then the selection process for interesting genes related to this group of terms is the following: find and append the terms that are near (up to a threshold) to the given group of terms. For these terms, collect all genes mapped to this extended set of terms, and sort the genes in decreasing fashion, according to the number of related terms. In the agglomerative process we might incorporate terms by applying an agglomerative distance (step-defined or similarity metrics) to encompass neighbouring terms, considered as putatively functionally relevant, due to distance-inferred similarity. Conversely, after ranking the terms we might consider only clearly different terms in the pruning phase, by eliminating terms, which are considered, above a certain threshold, similar.

Figure 5 illustrates the overall workflow of the suggested method. Below each step is succinctly described.

Step1. BubbleGenes: For each gene of the input, retrieve its GO annotations, and create a terms set.

Step2. BubbleGO agglomeration procedures:

- Graph-BubbleGO: For each GO term, retrieve also the parents and the children of the term for the specified aspect, and repeat this process for a predefined number of times (the distance parameter).
- Resnik-BubbleGO: For each GO term of the input, retrieve other GO terms, which have a normalized simRes value within a predefined radius.

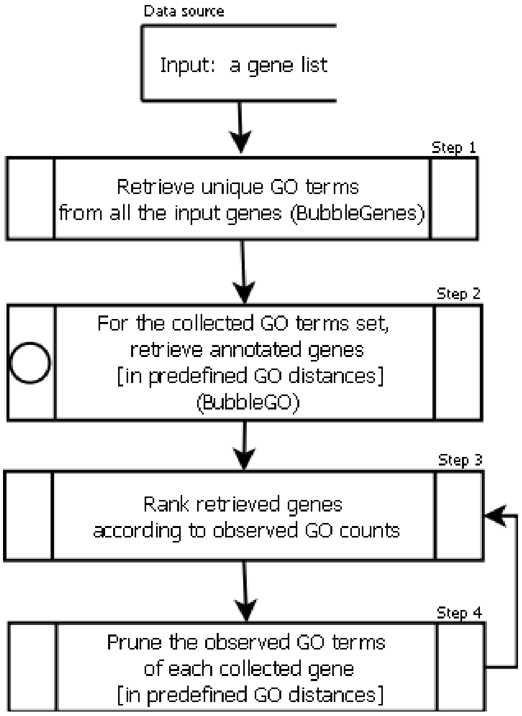

Data source
Input: a gene list
Step 1
Retrieve unique GO terms from all the input genes (BubbleGenes)
Step 2
For the collected GO terms set, retrieve annotated genes [in predefined GO distances] (BubbleGO)
Step 3
Rank retrieved genes according to observed GO counts
Step 4
Prune the observed GO terms of each collected gene [in predefined GO distances]

Figure 5: The workflow of the algorithm.

Step3. For the derived set of GO terms, retrieve the genes, which are annotated by them and rank the genes according to the observed degree. In this way, the genes considered to represent “hub” nodes are highlighted compared to the genes exhibiting less functional cross talk.

Step4. Optional pruning phase, which eliminates GO terms under a certain distance, considered thus to be adequately similar. In this way GO terms considered pretty diverse in terms of distance are selected, i.e. thus implying putative cross-talk of functionally different pathways:

Graph-Pruning: For the observed set of annotations of each gene, if some GO terms possess functional relevance (ancestors or descendants) within a predefined number of steps then keep the more specific (downstream) term. E.g. if we have a gene which has 5 observed GO terms, and 3 of them have a grandparent-parent-child relation, for the predefined distance:2 we keep only the child term, reducing the observed degree of the gene from 5 to 3.

Resnik-Pruning: For the observed set of annotations of each gene, we perform a variation of the complete linkage agglomerative hierarchical clustering [80],[81] regarding the normalized $\mathrm{sim}_{\mathrm{Res}}$ values of those GO terms. Setting the cut-off distance to predefined values, results in reduced sets of annotations. In this way, from neighbouring terms we keep only those with higher IC values.

There is a distinction for the case of ‘original GO degree’ versus ‘observed GO degree’ of a gene. The original GO degree of a gene is simply the total number of GO annotations it has regarding a specific GO aspect (BP, MF, CC). As seen in a bi-graph structure, this represents the degree of the gene, where genes are nodes connected with edges to GO terms. The observed GO degree is the number of GO annotations selected, after applying the GOrevenge algorithm, taking into account the various parameterization options. As a result the observed GO degree is of course smaller or equal to the number of all GO terms, initially linked to this gene.

3.2.1. Data assimilation

In order to achieve the online retrieval of GO databases and the calculations of Resnik similarities, code from GO2Sim [82] was refactored and optimized setting the core of the GOrevenge web application. Even though GO Data and Annotations are maintained in MySQL [75] tables, selected parts were ported in SQLite [83]. Resnik BubbleGO and Pruning methods require the pre-computation of Resnik similarities among thousands of GO terms and in order to compute these, the R package SemSim [84] and the FunSimMat online database [85] were initially tried. The first case performed slowly, while the second had limitations on the size of the data package that could be retrieved for each web service call (~50 Resnik similarity values per call). Even when the full Resnik database became available by the authors of FunSimMat, the handling of the >3GB table was unwieldy and time consuming. Our approach gives the capability of updating the GO and GOA database of GOrevenge for selected species on demand, and computes IC and Resnik similarities on every GO aspect separately, thus attaining significant speed up (from weeks to minutes in a double core 3GHz Intel processor).

The python function to compute the Resnik semantic similarities is displayed next:

```
def myResnik(IC, myancestors, organism, aspect, conn_str):
    ancestors = copy.deepcopy(myancestors)

    startTime=time.time()
    print 'Started'

    ICkeys=set(IC.keys())

    trees = {}

    def AncConstruct(go) : #puts  go with his ancestors
        anc = set()
        try: #if go in ancestors:
            anc = ancestors[go]
            anc.add(go)
        except: anc.add(go) # if it is the top, just add the go
        return anc

    def resnik(mygos) :

        for i, go1 in enumerate(mygos):
```

```
22              #if i%50==0: print 'i=',i
23              for go2 in mygos[i:] :
24
25                  ICcom, ICmax  = [], 0
26                  ICcom_a = ICcom.append
27
28                  anc1 = trees[go1]
29                  anc2 = trees[go2]
30
31                  common = anc1 & anc2 #intersection
32
33                  [ICcom_a(IC[go]) for go in common if go in ICkeys]
34
35                  try : ICmax = max(ICcom)
36                  except ValueError : ICmax = 0 #'-';print go1,go2
37
38                  mytab[mylabels[go2]][mylabels[go1]] = mytab[mylabels[go1]][mylabels[go2]] = str(ICmax)[:4]
39
40      golist = set(IC.keys())
41
42      for go in golist : trees[go] = AncConstruct(go)
43
44      mygos=trees.keys()#[:100]
45
46      mytab=  [[0 for i in range(len(mygos))] for j in range(len(mygos))]
47      mylabels={}
48      mygos.sort()
49      for i,go in enumerate(mygos):
50          mylabels[go]=i
51
52      resnik(mygos)
53
54      elapsed = time.time() - startTime
55      print 'Elapsed:',elapsed
56      return (mylabels,mytab)
```

Since this part of computation is quite heavy, optimisation techniques have been employed after profiling of the relevant code. Such techniques are: the extensive use of list comprehension, dictionaries, set types for the finding of common ancestors, definition of internal functions to minimise the call stack (lines 11 and 19), and pickling or gzip for the saving and the compression of the results.

One of the issues regarding applications relying in annotations is the need for frequent update of the relevant databases. This applies to the annotations of the organisms in use, as well as the structure of the ontology itself. The following code presents two functions, which probe the respective repositories for the dates of the last updates. The URL of the repositories are: ‘ftp://ftp.ebi.ac.uk/pub/databases/GO/goa/’ and ‘http://archive.geneontology.org/latest-termdb’.

```
def scrape_goa_dates():
    """
    Returns a species dict with value the datetime.date creation of goa
    """
    species={'ARABIDOPSIS':'', 'CHICKEN':'', 'COW':'', 'HUMAN':'',
                       'MOUSE':'', 'RAT':'', 'ZEBRAFISH':'' }
    httplib.HTTPConnection.debuglevel = 1
    msg,error = '',''
    for specie in species.keys():
        request = urllib2.Request(
        'ftp://ftp.ebi.ac.uk/pub/databases/GO/goa/'+specie)
        try:
            socket.setdefaulttimeout(3)
            opener = urllib2.build_opener()
            urllib2.install_opener(opener)
            f = opener.open(request)
            data = f.readlines()
        except IOError,value:
            error += str(value)
            msg += specie+': IO ftp Error, using previous \
                                                    saved data (static folder) '
            file_wanted = 'gene_association.goa_' + specie.lower() + \
                                                    '.gz'
            mypath = 'applications/goreveng/static/files/goa/'
            if file_wanted in os.listdir(mypath):
                mystat = os.stat(mypath + file_wanted)
                ctime=datetime.datetime.fromtimestamp(mystat.st_ctime)
                species[specie]=ctime.date()
                size=mystat.st_size
            else: continue
            return (error,species,msg)

        for line in data:
            myline=line.split()
            if 'gene_association.goa_' + specie.lower() + '.gz' in myline:
                for i,word in enumerate(myline):
                    if word in months_choices:
                        month_num = months_choices[word]
                        day_num = int(myline[i+1])
                        try: year = int(myline[i+2])
                        except:
                            if month_num > datetime.date.today().month:
                                year = OurYear-1
                            else: year = OurYear
                        size = int(myline[i-1])
                        break

                species[specie]=datetime.date(year, month_num, day_num)
    print species
    return (error,species,msg)


def scrape_go_date():
    httplib.HTTPConnection.debuglevel = 1
    request = urllib2.Request(
                                    'http://archive.geneontology.org/latest-termdb/')
    opener = urllib2.build_opener()
    f = opener.open(request)
    for i in f :
        try :
            DateLastUp = i.split('<a href="go_daily-termdb-\
       data.gz">go_daily-termdb-data.gz</a></td><td align="right">')[1]
            break
        except IndexError : pass
    DateLastUpGo = DateLastUp.split(' ')[0].split('-')
    day_num = int(DateLastUpGo[0])
    month_num = months_choices[DateLastUpGo[1]]
    year = int(DateLastUpGo[2])

    return datetime.date(year, month_num, day_num)
```

The insertion into SQLite tables of the species annotations entails steps as the retrieval and reading of the compressed files from the repository, dropping of unnecessary tables to reduce the stored size, and using optimised insertion techniques for speeding up the process:

```
def CreateGOAdb(adresse, organisme, conn_str) :
    socket.setdefaulttimeout(5)
    httplib.HTTPConnection.debuglevel = 1
    request = urllib2.Request(adresse)
    request.add_header('Accept-encoding', 'gzip')
    opener = urllib2.build_opener()
    try:
        f = opener.open(request)
        compresseddata = f.read()
        compressedstream = StringIO.StringIO(compresseddata)
        gzipper = gzip.GzipFile(fileobj=compressedstream)
    except urllib2.URLError:
        f = 'downloaded'+os.sep+adresse.split('/')[-1]
        gzipper = gzip.GzipFile(filename=f)
    data2 = gzipper.readlines()
    gzipper.close()

    drop_sql, create_sql = "DROP TABLE IF EXISTS `annotation_" + organisme + "`;", "CREATE TABLE `annotation_" + organisme + "` (  `DB` varchar(255) NOT NULL,  `DB_Object_ID` varchar(255) NOT NULL,  `DB_Object_Symbol` varchar(255) NOT NULL,  `Qualifier` varchar(255) NOT NULL,  `GO_ID` varchar(255) NOT NULL,  `DB_Reference` varchar(255) NOT NULL,  `Evidence` varchar(255) NOT NULL,  `With` varchar(255) NOT NULL,  `Aspect` varchar(255) NOT NULL,  `DB_Object_Name` varchar(255) NOT NULL,  `Synonym` varchar(999) NOT NULL,  `DB_Object_Type` varchar(255) NOT NULL,  `Taxon_ID` varchar(255) NOT NULL,  `Date` int(25) NOT NULL,  `Assigned_By` varchar(255) NOT NULL,  `Annotation_Extension` varchar(255) NOT NULL,  `Gene_Product_Form_ID` varchar(255) NOT NULL) ;"

    Maj2 = data2

    ImportSql(drop_sql, conn_str)
    ImportSql(create_sql, conn_str)

    setinsert=[]
    for i in Maj2 :
        if "gaf-version" not in i:
            myserie = i.split('\t')[:-1]+['']
            assert len(myserie)==17
            setinsert.append(myserie)

    conn = sqlite3.connect(conn_str)
    conn.text_factory = sqlite3.OptimizedUnicode
    curs = conn.cursor()
    curs.executemany("insert into annotation_"+ organisme +
       " values (?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?)" ,setinsert)
    conn.commit()
    curs.close()
```

```
conn.close()
```

3.2.2. Computational Evaluation

Regarding the evaluation of the proposed methodology, the pancreatic cancer gene database (PC-GDB) [86] was used, comprising of 119 genes. Applying a 10 fold cross-validation procedure, 10% of the cancer genes served as input to the GOrevenge algorithm, each time. From the resulting list of genes, which are sorted to the observed GO degree (namely their number of connected GO terms), the input genes were eliminated, so that only the 90% of genes is ranked. In order to avoid the insertion of penalty values for cases where some of the targeted cancer genes were not retrieved, the remaining human genes are added to the list, ordered by their original GO degree. From this ranked list, firstly the positions of the target cancer genes and then their median are derived. This process is also applied for the same cancer genes, to a list of human genes sorted by their original GO degrees, which serves as the control case. The medians of the positions of cancer genes in the list, produced by the algorithm were compared to the medians of the positions of the same cancer genes in the control list, so as to evaluate whether GOrevenge manages to retrieve earlier the remaining cancer genes, by scoring them higher than genes not referring to the pancreatic list. The Wilcoxon signed rank test with continuity correction was employed for the significance test. The statistical evaluation presented in the next Section and the plotting of the results were executed in R [75].

3.2.3. Biological validation

In order to validate biologically the resulting gene sets which are output from GOrevenge, the whole pancreatic cancer gene set (119 genes) was used as input. This gene list, which produces statistical significant results when tested for each GO aspect, depicts genes reverse sorted by the sum of indexes for each aspect. The same method was applied to another gene set related to T-cell Acute Lymphoblastic

Leukemia (ALL), comprising only of 9 genes [87], utilizing the Graph-BubbleGenes algorithm.

3.3. Multi-Modal Data Fusion of Separate Datasets

3.3.1. Assimilation of Image data

The dataset derived from skin lesion images contained 972 instances of nevus skin lesions and 69 melanoma cases. Three types of features are analysed: Border Features which cover the A and B parts of the ABCD-rule of dermatology, Colour Features which correspond to the C rules and Textural Features, which are based on D rules. The total number of features assessed was 31 from the initial set of 32 (one feature was removed as having zero variation across the samples). The relevant pre-processing which produced all the features is described in [88].

3.3.2. Assimilation of Microarray data

The dataset regarding microarray data was taken from the Gene Expression Omnibus (GEO) [51], GDS1375. In that experiment, total RNA isolated from 45 primary melanoma, 18 benign skin nevi, and 7 normal skin tissue specimens were analysed on an Affymetrix Hu133A microarray containing 22,000 probe sets [89]. The dataset contains the values of MAS5-calculated signal intensities after global scaling the average intensity to 600.

The data retrieval from GEO was performed using GEOquery [90] and processed with limma [91] R packages from the Bioconductor project [92], following the main steps as listed in the R script produced by the GEO2R tool [93]. The input contrast levels were differentially expressed genes between melanoma versus skin and melanoma versus nevus. 1701 genes from a linear model fit were extracted setting FDR for multiple testing adjustment, p-value 0.001 and 2-fold changes as thresholds. As a normalization step, after taking the logarithms of the values, the mean values of normal skin were subtracted from the rest of the data, and the normal skin columns were removed from the table:

```
#   Differential expression analysis with limma
library(Biobase)
library(GEOquery)
library(limma)

# load series and platform data from GEO

gds <- getGEO(filename="GEODATA/GDS1375.soft.gz")
anGPL <- getGEO(filename="GEODATA/GPL96.soft") #"GPL96.annot.gz"
gset <- GDS2eSet(GDS=gds,GPL=anGPL)

# group names for all samples
sml <- c("G0","G0","G0","G0","G0","G0","G0","G1","G1","G1","G1","G1","G1",
"G1","G1","G1","G1","G1","G1","G1","G1","G1","G1","G1","G1","G2","G2","G2",
"G2","G2","G2","G2","G2","G2","G2","G2","G2","G2","G2","G2","G2","G2","G2",
"G2","G2","G2","G2","G2","G2","G2","G2","G2","G2","G2","G2","G2","G2","G2",
"G2","G2","G2","G2","G2","G2","G2","G2","G2","G2","G2","G2");

# log2 transform
ex <- exprs(gset)
exprs(gset) <- log2(ex)

# set up the data and proceed with analysis
fl <- as.factor(sml)
gset$description <- fl
design <- model.matrix(~ description + 0, gset)
colnames(design) <- levels(fl)
fit <- lmFit(gset, design)
cont.matrix <- makeContrasts(G2-G0, G2-G1, levels=design) #G1-G0
fit2 <- contrasts.fit(fit, cont.matrix)
fit2 <- eBayes(fit2)

result <- decideTests(fit2,p.value=0.001,lfc=1)
allres <- result[result[,1] !=0 & result[,2] !=0,] # & result[,3] != 0,]
vennDiagram(allres)


# normalization
lex <- log2(ex)
mean_skin = rowMeans(lex[,gset$description == 'G0'],)
nor_lex = lex - mean_skin
final_lex = nor_lex[rownames(allres),]
#
colnames(final_lex) <- sml
final_lex <- final_lex[,colnames(final_lex)!='G0'] #remove now the normal skin cols
myannot <- fData(gset)[ fData(gset)$ID %in% rownames(final_lex) ,
                                        c('ID','Gene Symbol')]
myannot$'Gene Symbol' <- as.character(myannot$'Gene Symbol') #not factor
names(myannot) <- c('ID','Symbol') #not 'Gene Symbol'

#clear memory
rm(gds,anGPL)

mylexdf=data.frame(final_lex)
mylexdf$ID  <-  rownames(final_lex)
DSm =merge(mylexdf,myannot) #by ID column

```

```
#check ID - Gene Symbol correctness
stopifnot( (fData(gset)['200001_at','Gene Symbol']) ==
                         DSm[DSm$ID=='200001_at','Symbol'])
stopifnot(subset(DSm,subset=ID=='213029_at',select=Symbol)=='NFIB')

DSm$origin='m' # from microarray
save(DSm, file = "myDATA/DSm.RData")
```

3.3.3. *Data integration*

The two tables containing the microarray and image data were merged to one block sparse matrix with dimensions 1104 rows x 1734 columns, marking the not available values as NA. The rows contain the microarray and image data samples, and the columns microarray and image features plus one binary response variable (0 for nevus and 1 for melanoma). All the programming of the workflow was implemented in R[75]. In Appendix 3 are listed functions for the construction of the workflow.

The R code for construction the unified data table is shown next:

```
if (! 'DSm' %in% ls()) {load("myDATA/DSm.RData");print('DSm loaded')}
if (! 'DSi' %in% ls()) {load("myDATA/DSi.RData");print('DSi loaded')}

#make a dataframe for DSm where genes are columns, and also a class column
#with 1 for melanoma
Gcolumns <- grep('^G.*', names(DSm)) #find the indices of G columns
tDSm <- t(DSm[,Gcolumns])
tDSmDF <- as.data.frame(tDSm)
colnames(tDSmDF) <- make.names(DSm$Symbol,unique=T) #as col names the Gene Symbol
tDSmDF$myOrigin <- rownames(tDSmDF) #make a myOrigin column
tDSmDF$class = 0 #make a class column: 0 [derma&nevus] and 1 [melanoma]
G2rows <- grep('^G2.*', tDSmDF$myOrigin)
tDSmDF$class[G2rows] = 1
#write.table(x=tDSmDF,file="myDATA/DSm.csv",sep=";")

#check transformation is OK
stopifnot(DSm[,"G2.36"][DSm$Symbol == "CAPNS1"] ==
                  tDSmDF[tDSmDF$myOrigin=="G2.36","CAPNS1"])

DSm2 <- tDSmDF #better name
rm(tDSmDF)
save(DSm2, file = "myDATA/DSm_ready.RData") #a dframe only with microar. data in proper form
mClass <- DSm2$class
DSm2 <- subset(DSm2, select=c(-myOrigin,-class)) #only gene cols

# image data
rownames(DSi) <- DSi$NAME
iClass <- DSi$class
DSi2 <- subset(DSi, select=c(-origin,-NAME,-class)) #only image attr cols

#append to DSm2a NA colums as many as the number of image cols
DSm2a <- data.frame(DSm2,check.names=T) # a copy, as not to change DSm2
DSm2a[,colnames(DSi2)]=NA
```

```
stopifnot( ncol(DSm2)+ncol(DSi2) == ncol(DSm2a) ) #check num cols

#insert in front to DSi2a NA colums as many as the number of gene cols
tt <- matrix(nrow=nrow(DSi2), ncol=ncol(DSm2),
        dimnames=list(rownames(DSi2),colnames(DSm2)))
tt <- (as.data.frame(tt))
DSi2a <- data.frame(tt,DSi2)

#check name correspondence
stopifnot(names(DSi2a)[c(1,100,200,ncol(DSi2a))] ==
                names(DSm2a)[c(1,100,200,ncol(DSm2a))])

#for one case X.. that was different
colnames(DSi2a) <- colnames(DSm2a)

#make one DS
DS <- rbind(DSm2a,DSi2a) #first microarray rows and then image rows
myclass <- c(mClass,iClass)
DS$class <- myclass
save(DS, file = "myDATA/DS.RData")
#write.table(x=DS,file="myDATA/DS.csv",sep=";")
```

3.3.4. Missing values imputation

Although there are several software packages implementing advanced imputation methods [94], they could not be utilized in this unified dataset where the multi-modal data have only the class variable column as complete. In this study we considered four simple imputation methods applied per feature and per class:

- “mean value” imputation
- “random normal” imputation
- “uniform” imputation
- “bootstrap” imputation

In the second case, after taking the mean value (m) and standard deviation (sd) of each feature (ignoring the NA values) per class, we randomly filled the missing values sampling from an assumed normal distribution having as parameters: (m, sd). The “uniform” imputation is conducted by sampling uniformly within the range of each feature per class, and the “bootstrap” imputation by independent bootstap of each variable separately per class, until all the NA values are replaced. The last two imputation methods are similar to the way random forests construct synthetic data in order to provide for a similarity measure [69].

For the efficient execution of the imputations, the `plyr` R package was used [95].

Parts of the code that perform the “uniform” and the “bootstrap” imputations are shown next:

```r
#----------------------------
#uniform sampling imputation
#----------------------------
f2u <- function(X){
  ranges <- range(X, na.rm=T)
  isna <- is.na(X)
  c <- sum(isna)
  X[isna] <- runif(c,min=ranges[1],max=ranges[2])
  X
}
DSru <- keeping.order(DS, ddply, .(class), colwise(f2u))
save(DSru, file = "myDATA/DS_runiform_imputation.RData")
hist(DSru[is.na(DS[1]) & DSru$class==0,2])
hist(DSru[!is.na(DS[1]) & DSru$class==0,2])

#----------------------------
#bootstrap imputation
#----------------------------
f2b <- function(X){
  isna <- is.na(X)
  c <- sum(isna)
  X[isna] <- sample(x=X[!isna],size=c,replace=T)
  X
}
DSb <- keeping.order(DS, ddply, .(class), colwise(f2b))
save(DSb, file = "myDATA/DS_bootsrap_imputation.RData")
hist(DSb[is.na(DS[1]) & DSb$class==0,2])
hist(DSb[!is.na(DS[1]) & DSb$class==0,2])
```

3.3.5. Feature Selection

The setup of the in-silico experiment involved the examination of the reported selected feature subsets when: a) applying a co-linearity removal filter to the microarray dataset prior to the execution of the selection algorithm (marked as: Filtered/Unfiltered), and b) setting a 95% tolerance threshold to the best obtained performance criterion (Tolerance/Best). The tolerance in the performance method allows the selection of a subset size that is small enough but without sacrificing too much performance, and can produce good results where there is a plateau of good performance for larger subset sizes. The combination of these two parameters (prior filtering and tolerance threshold) resulted in the examination of four distinct cases.

For each of the four cases, a 10-fold cross-validation procedure was performed with 50 repetitions on six different datasets: only the microarray data (marked as om), the unified dataset produced by the mean imputations (m.i), the unified dataset by normal random imputations for the NA values (nr.i), the unified dataset by the “uniform” imputations (u.i), the unified dataset by the “bootstrap” imputations (b.i) and only the image data (oi). In all the repetitions, the nr.i, u.i and b.i datasets were imputed anew, thus providing more sampling variations. Prior the application of the repetitions, the datasets were centred and scaled as a pre-processing step on the predictors.

The feature selection workflow was setup using the R package `caret` (classification and regression training) [96]. The search algorithm employed in caret uses the recursive feature elimination method on predefined sets of predictors, and in this study the length of the variable subsets was defined as [1 to 10, 15, 20, 25, 30, 35, 40, 45, 50], except for the image only data where the subsets were [1 to 10, 15, 20, 25, 30, 31].

```
...
subsets <- c(1:10,15,20,25,30,35,40,45,50)
...
# ----Caret Parameters------
rfFuncs$summary <- twoClassSummary
rfFuncs$selectSize <- pickST # allow 5% tolerance #pickSizeTolerance   #pickSizeBest
ctrl <- rfeControl(functions = rfFuncs,
method='repeatedcv',verbose=F,returnResamp='none',repeats=1)
...
for (i in 1:mycounts) {
  rfmodel <-
rfe(x,ym,sizes=subsets,rfeControl=ctrl,metric="ROC",maximize=T,strata=ym,sampsize=c(18,18))
  myresnum[i] <- rfmodel$bestSubset
  myresvars[[i]] <- rfmodel$optVariables
  myperf[i] <- subset(rfmodel$results,
                    subset=Variables==rfmodel$bestSubset,
                    select=ROC)
  cat(i,' ')
}
```

For each of the 50 repetitions, the optimum subset number of predictors was recorded, along with the names of the predictors, and the performance attained. As performance metric the area under the ROC curve (auc) was set. The auc of a classifier is equivalent to the probability that the classifier will rank a randomly chosen positive instance higher than a randomly chosen negative instance. This is equivalent to the Wilcoxon test of ranks, and also it is closely related to the Gini coefficient [97].

3.3.6. Multivariate Statistical Analysis

Statistical analysis was performed using multivariate techniques, specifically principal component analysis (PCA), followed by linear discriminant analysis (LDA). For all the cases, the co-linearity removal filter from the `caret` package was applied to the microarray data. PCA was performed by the `prcomp` R function and LDA by the `lda` function from the `MASS` R package [98].

PCA is an unsupervised method and a data reduction technique that allows the major sources of variation in a multi-dimensional dataset to be analysed without introducing inherent bias. PCA provides a direct mapping of high-dimensional data into a lower-dimensional space containing most of the information in the original data. PCA defines new variables, consisting of linear combinations of the original ones, in such a way that the first axis is in the direction containing most variation. Every subsequent new variable is orthogonal to previous variables, but again in the direction containing most of the remaining variation. The new variables are called principal components (PCs). The coordinates of the samples in the new space created by the PCs are called scores [99].

LDA uses class information to maximize the separation between various groups of observations. LDA requires that the classification variables follow a normal multivariant distribution and the covariance matrixes for the observations of each class are equal (homoscedasticity). If these requirements are not met, LDA is not the best possible classifier, but it can still be considered a suitable method to search for projection directions that maximize the separation between elements of different classes and this purely geometric interpretation is not affected by hypotheses on the distribution of data [100].

The `lda` function has two working modes. One having the parameter CV=False (the default), allowing then to obtain an object that includes discriminant scores, and the other with CV=True, were predictions of class memberships are derived from leave-one-out (LOO) cross-validation. Initially, we run the `lda` function at each of the mentioned datasets and retrieved the attainable scores, as well as the svd parameters. Svd are the singular values, which give the ratio of the between- and within-group standard deviations on the linear discriminant variables. Their squares are the canonical F-statistics. Afterwards the `lda` function (having CV=True) was run 50 times at each dataset (unified datasets were imputed anew each time), in order to assess the variability of the attained accuracy performance for the melanoma class. Finally, the `lda` function was run again 100 times (having CV=False) at each dataset in order to assess the stability of the suggested top performing features.

The set of 20 top scoring biomarkers was used to consider the prediction performance for the melanoma case. The LDA (CV=True) and RF (stratified, 18 samples per class) methods were run 50 times anew on each dataset using only the biomarkers' columns and the class as response variable.

The R function `perfeval` that computes the performance evaluations is shown next (Appendix 3.5):

```
perfeval <- function(myfun, mytimes=50, original_data=F, image=F){
  res=list("total_lda"=rep(0,times=mytimes),"total_rf"=rep(0,times=mytimes))
  for (i in 1:mytimes){
    if (original_data){
        mybiomarkers <- biomarkers$names
        if (image){
          mybiomarkers <- mybiomarkers[mybiomarkers %in% colnames(DSi)]
          selrows <- !grepl("[G*]",row.names(DS))
        } else{
          mybiomarkers <- mybiomarkers[mybiomarkers %in% colnames(DSm2)]
          selrows <- grepl("[G*]",row.names(DS))
    }

    DSmi <-  DS[selrows, colnames(DS) %in% mybiomarkers]
    y2 <- factor(DS$class[selrows])
    y2m  <- factor(DS$class[selrows],levels=c(1,0)) #melanoma class fist level in factor
    x2 <- DSmi
  } else {
   DSmi  <- keeping.order(DS[,c(biomarkers$names, 'class', 'Grad.min')],
              ddply, .(class), colwise(myfun))
   y2 <- factor(DSmi$class)
   y2m  <- factor(DSmi$class,levels=c(1,0)) #melanoma class fist level in factor
   x2 <- subset(DSmi, select=c(-class, -Grad.min)) # Grad.min is 0 everywhere
  }

  normalization <- preProcess(x2)
  x2 <- predict(normalization, x2)
  x2 <- as.data.frame(x2)

  xmi_ldacv <- lda(x2,y2m,CV=T)
  res$total_lda[i] <- confusion(y2m,xmi_ldacv$class,printit=F)[1,1]

  rf <- randomForest(x=x2,y=y2m,strata=y2m,sampsize=c(18,18),importance=F)
  res$total_rf[i] <- 1 - rf$con[1,3]
 }
return(res)
```

CHAPTER 4. EXPERIMENTS AND RESULTS

In this chapter the outcomes from integrating 6 KEGG metabolic pathways central to human organism in an automated procedure is presented, using KEGGconverter in comparison to other available tools. Reports on the results of the statistical and biological evaluation of the GOrevenge algorithm on a pancreatic cancer gene set and to an Acute Lymphoblastic Leukemia gene set follows, using both graph and Resnik distances over the three aspect of Gene Ontology (BP,MF,CC). Last, there is an analysis of the stability and the class discriminating capability of a set of highlighted biomarkers proposed by the novel multi-modal fusing/imputation method as applied to the cutaneous melanoma disease.

4.1. KEGGconverter

In order to highlight the application of KEGGconverter, a case study is presented concerning the integration of 6 KEGG pathways, related to the central metabolism of the Homo sapiens organism (human), to a single model:

- Glycolysis / Gluconeogenesis - map00010
- Citrate cycle (TCA cycle) -map00020
- Pentose phosphate pathway - map00030
- Fatty acid biosynthesis -map00061
- Fatty acid metabolism -map00071
- Urea cycle and metabolism of amino groups -map00220.

Next, we implement a comparative analysis of the performance of KEGGconverter, with two state-of-the-art workflows for the implementation of analogous tasks.

Selecting the 6 pathways and transferring them to the input directory of the KEGGconverter is all the necessary data acquisition procedure that must be followed. The user selects whether default kinetics is to be added by using the “makeKinetics” or “justConvert” command parameter to perform just the transformation to SBML, and the conversion begins. At the conclusion of the conversion the integrated model is transferred to the output directory. The model is ready for inspection as well as to serve as a base for subsequent experimental simulations in an appropriate software environment. Figure 6 shows the resulted model inserted in CellDesigner.

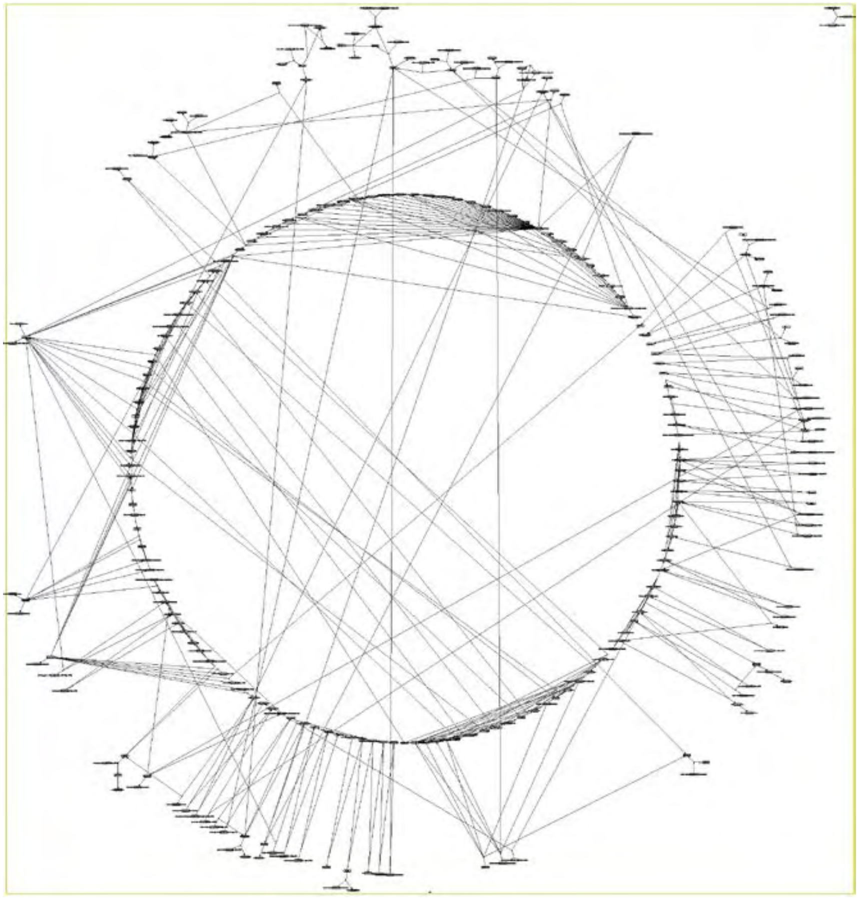


Figure 6: a circular layout diagram of the resulted model from CellDesigner

A sample simulation of the resulted model having default initial concentration values for the species is shown at Figure 7.

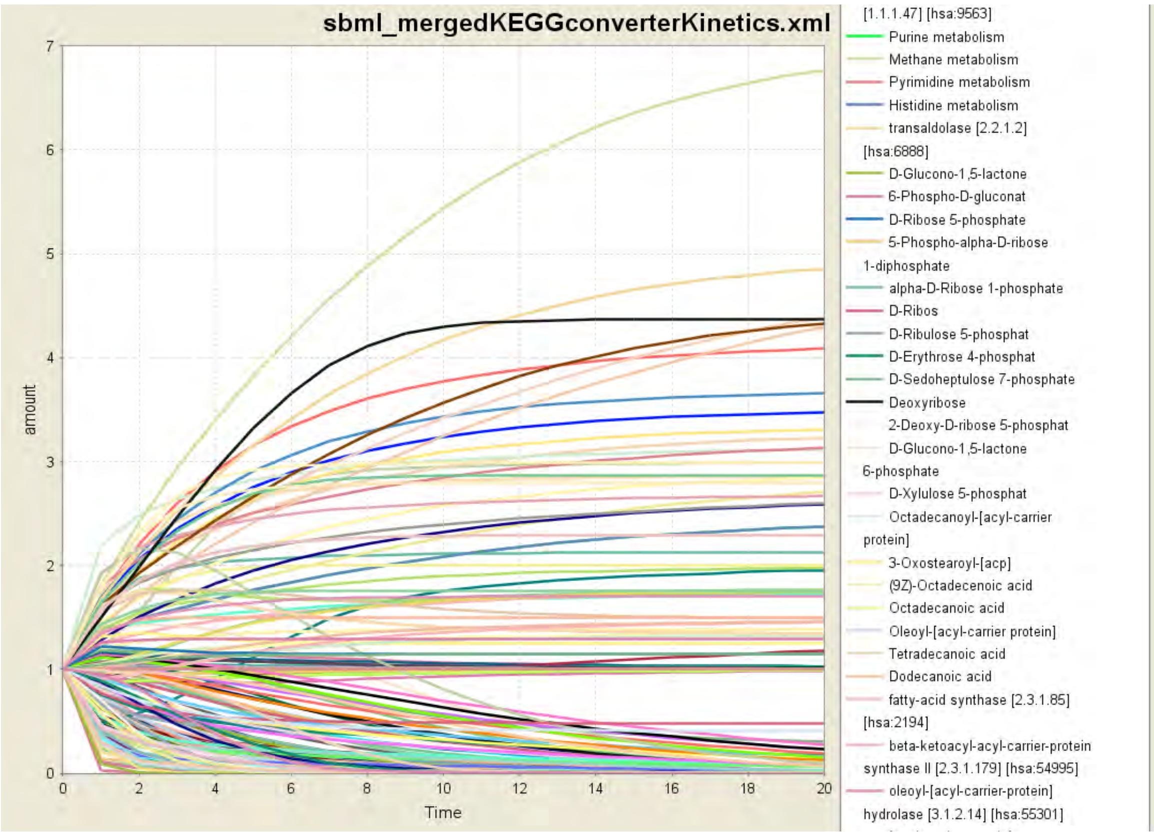


Figure 7: In the legend of the Figure, only part of the participated species is shown. The number of species of the resulted model is 266, including the neighbouring pathways represented as distinct species. For example: the purine, methane, pyrimidine and histidine metabolisms that are shown at the top of the legend. These neighbouring pathways, among others that are not apparent in the legend, take part in the simulation of our resulted model, and offer additional information for the direction of the metabolic flow of the integrated reaction network.

In comparison to this automated procedure, we enlist two attempts to conclude to a similar outcome, either by using KGML-ED, which is capable of importing KGML files, either by a combination of other tools starting with KEGG2SBML for the production of SBML converted models.

4.1.1. The KGML-ED case

Using KGML-ED [26] which can be run as a Java Web-Start application, we can import the six KGML files and merge the model to an integrated network. Selecting ‘Condense into single entities’ the duplicate elements are removed and the resulting pathway can be saved as a file in KGML format. At Figure 8, the graph of the integrated network using KGML-ED is shown. The next step was to use KEGG2SBML, which allegedly according to [17] could use and parse KGML files to SBML. Noteworthy, the recent version of the tool does not make use at all of KGML files as input, but .coord or .conf flat files from the KEGG Metabolic Pathway database and the LIGAND database. So this analysis pipeline could not be further implemented for the specific simulation task.

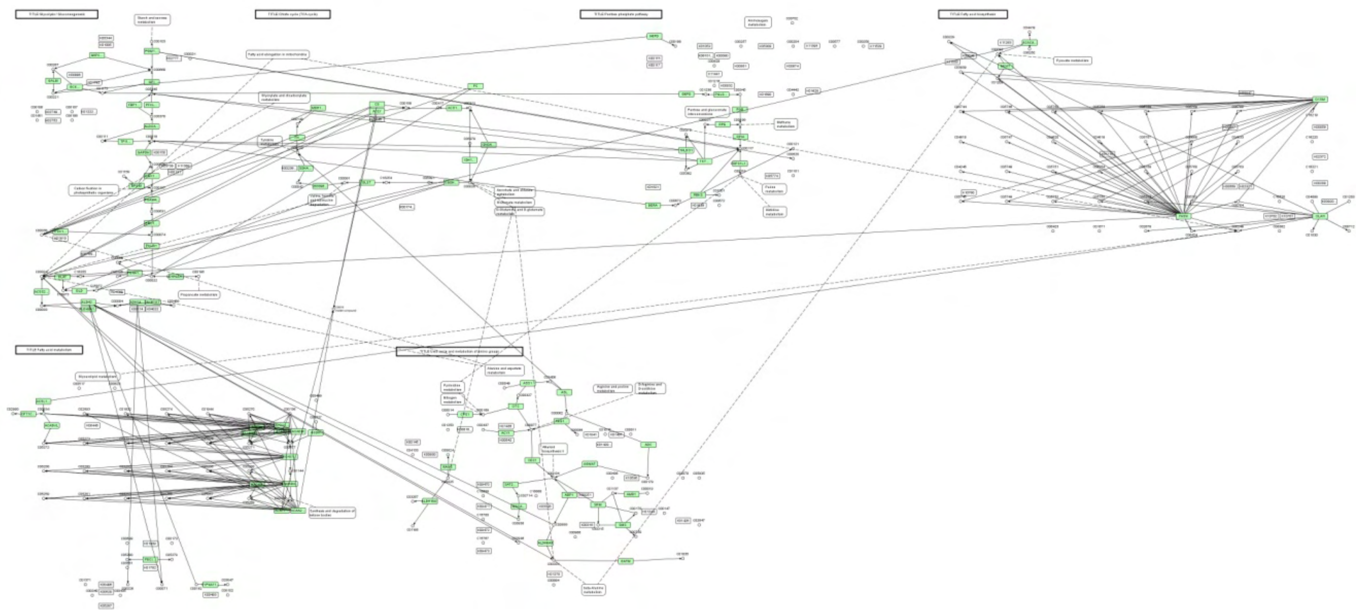


Figure 8: The case study integrated network using KGML-ED

Here it should be mentioned, that tools such as VisANT [101] and VANTED [102] have the ability to load KEGG Pathways, but cannot convert the resulted networks to SBML.

4.1.2. KEGG2SBML – semanticSBML/mergeSBML – SBMLSqueezer

It seems that -apart of the KEGGconverter- the use of KEGG2SBML tool is so far, the only other functional solution for the SBML conversion task. However, this solution requires a UNIX-based operational environment, together with Perl5 and KEGG database files, as previously mentioned. The execution of the conversion is performed through the assignment from the command line of a command like:

“kegg2sbml -l 2 -v 1 -c 0 ./PATHWAY/hsa/hsa00010_cpd.coord”, which produces a corresponding SBML/hsa/hsa00010.xml file. The options we activated through appropriate command switches in the command line are for the production of Level 2 Version 1 SBML files without CellDesigner annotations.

At the next step the converted files were inserted in the semanticSBML [103] tool (version 1 beta). This application completed the merging process and produced an integrated SBML Level 2 Version 3 model.

The installation of semanticSBML entails the preinstallation of other packages such as: Python, Qt4, PyQt3, SOAP.py, libSBML3 [104], Graphviz (currently not working in MS Windows but only required for the view function). There is also an online version of this tool, which produced the merged SBML file after the uploading of the six SBML models.

The inspection of the L2V3 SBML model became possible in CellDesigner 4.0.1 only after the conversion of the model (using Python and libSBML 3) to a lower version (L2V1), since L2V3 SBML is not yet supported in CellDesigner. Scrutinizing the derived model revealed problems introduced by the semanticSBML tool during the SBML conversion of KEGG pathways: the model contained 6 different compartments, one for each of our case pathways (Figure 9). Even after intensive manual editing of the file, for the removal of redundant compartments, the integrated pathway could not be reproduced since without the duplicates elimination procedure, the pathways remain separate, as seen in Figure 10.

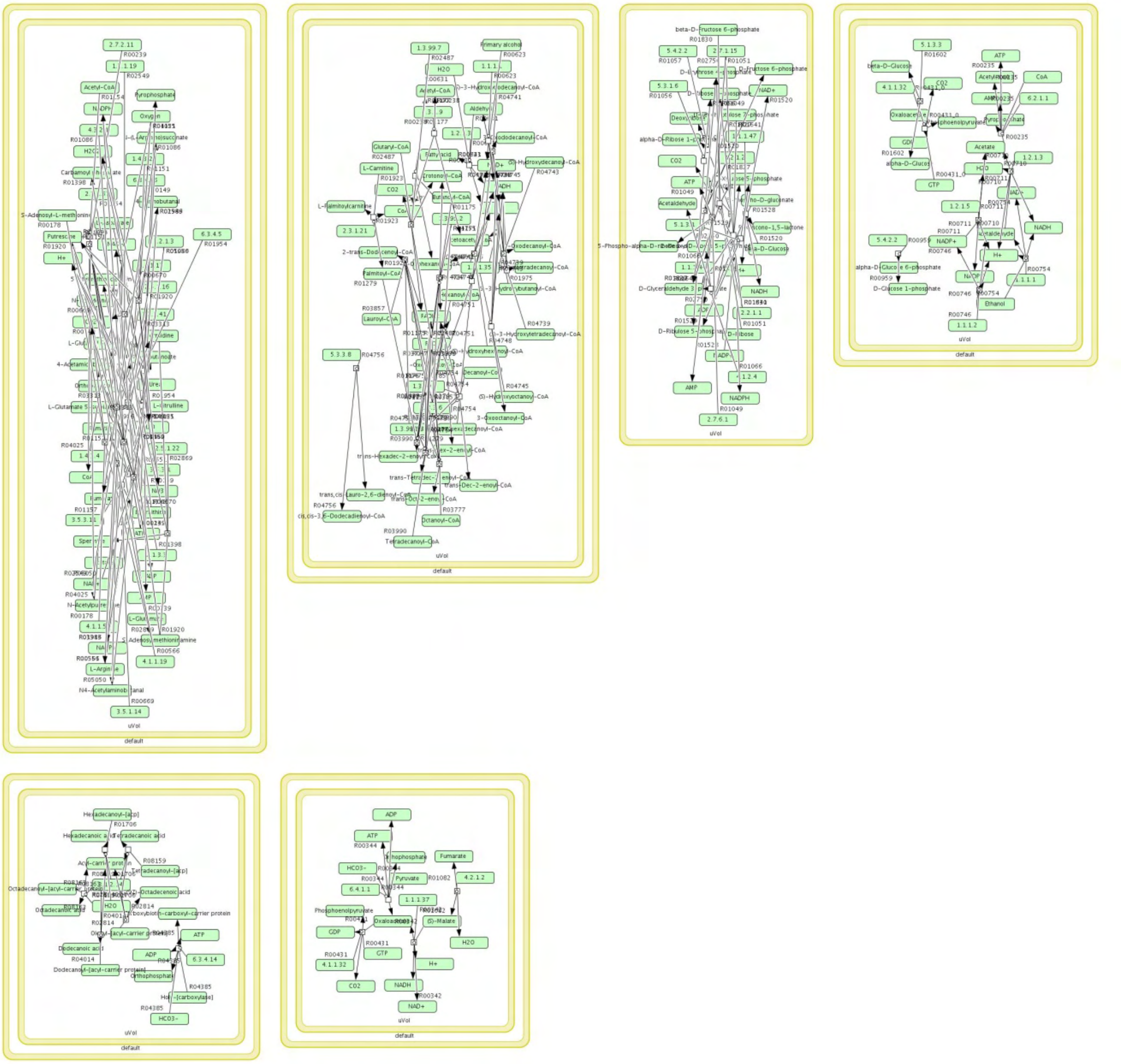


Figure 9: Alternative procedure: KEGG2SBML-> SemanticSBML. Compartmental division of the pathways. The model contained 6 different compartments, one for each of the case pathways

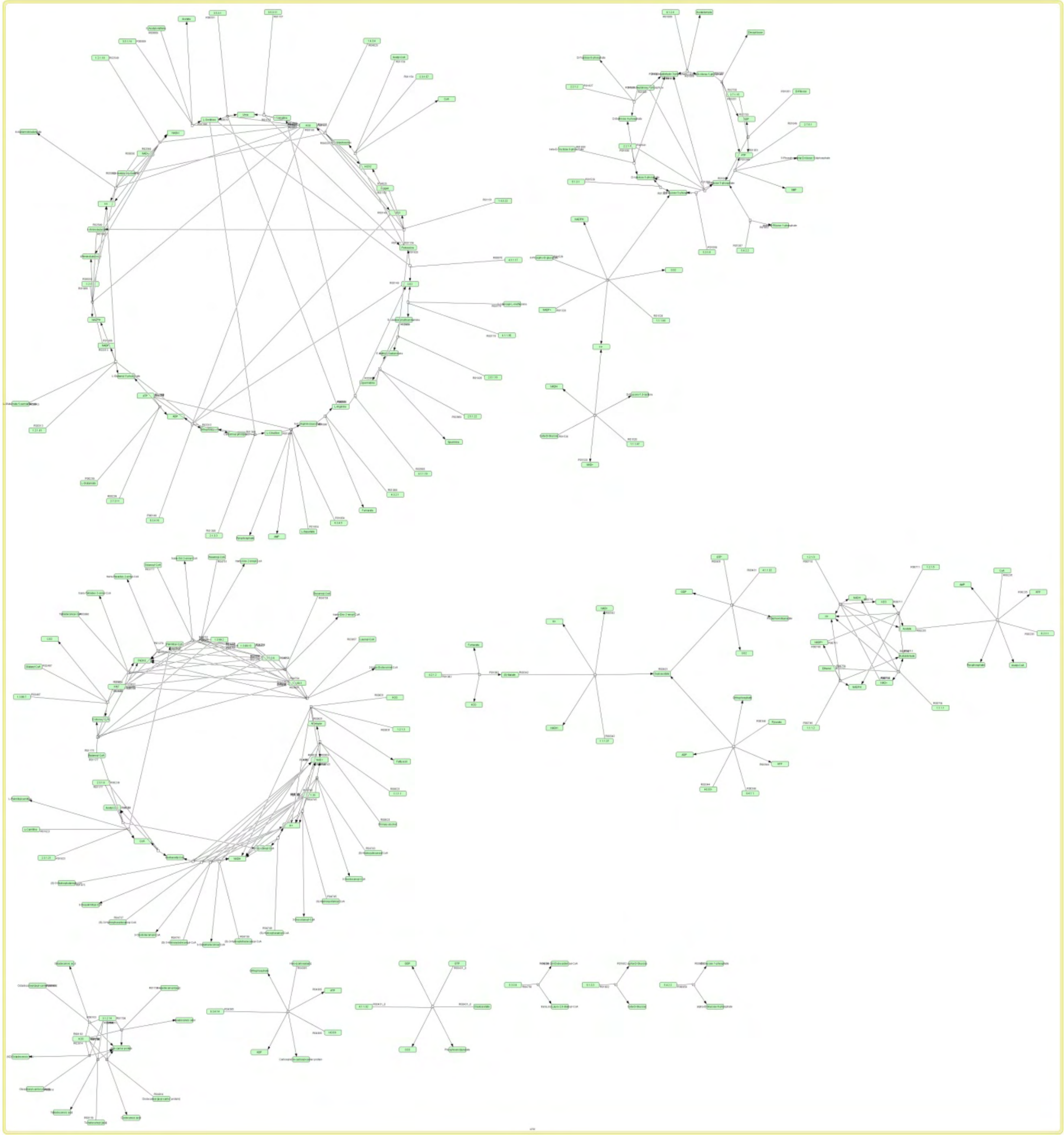


Figure 10: Alternative procedure: KEGG2SBML- SemanticSBML. Unconnected pathways. Even after the manual removal of different compartments, the pathways do not integrate.

An alternative approach for the merging of the six SBML models was made possible by running the simple bash script “mergeSBML.sh” mentioned in [105]. This script produces a merged SBML model that successfully integrates the imported pathways.

Nevertheless, a comparison to the number of relations included in the two integrated models, shows that the model which was constructed with the alternative procedure (KEGG2SBML → SemanticSBML/mergeSBML) contained 70 distinct reactions, while the model from KEGGconverter incorporated 154 reactions, not counting those that refer to the boundary conditions to neighbouring pathways. This dramatic difference is due to the dramatically less information that KEGG2SBML manages to extract from the KEGG flat files, comparing to the KGML files. This difference, of course, seriously deteriorates, as the number of the merged pathways increases.

The last step before the simulation phase for our test case, is the insertion of kinetic information about the pertaining reactions of the model, through the tool SBMLsqueezer [106]. This tool has the ability to massively implement the incorporation of reaction kinetics where missing, by selecting some default values. In this way, it renders the model ready for the simulation phase. It works as a CellDesigner plug-in, and its installation entails only the downloading of the jar file in the plug-in directory of CellDesigner.

Currently, KEGGconverter does not add SBML annotations concerning the involved species and reactions, whereas SemanticSBML provide for a MIRIAM [107]-complied data format and SBMLsqueezer incorporates annotations in the form of Systems Biology Ontology (SBO) [108]. [See Table 1: for a comparison of features table relating the case study tools].

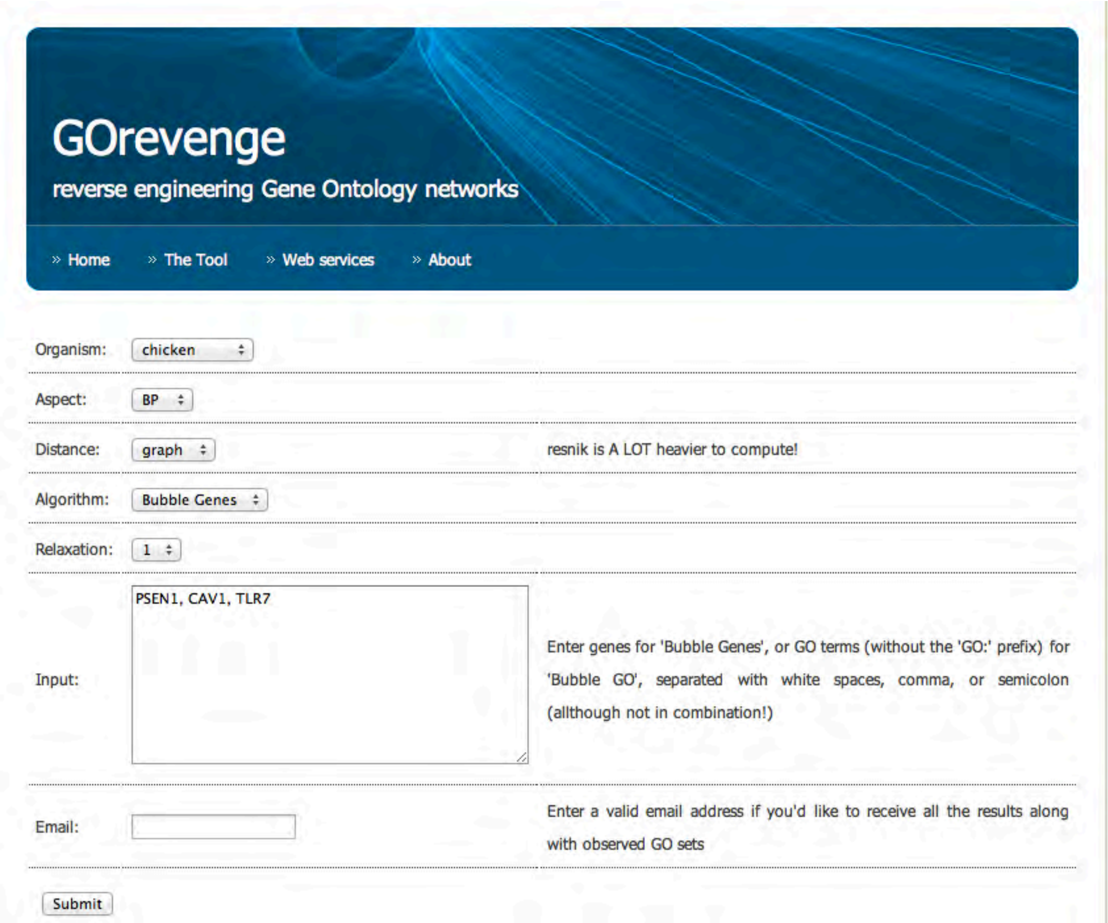
	Reads KGML	**Converts to SBML**	***Merging***	***Redundancy***	***Removes Duplicates***	***Graph editing***	***Kinetics***	***Annotations***	***Environment***
KGML-ED	yes		yes	yes (manually)	yes	yes			independent
KEGG2SBML	No, it uses KEGG DB files	yes							Unix like
semanticSBML		yes L2V3	yes	yes	yes	yes (for simple graphs)		yes	Windows / Linux & on-line
SBMLsqueezer							yes	yes	CellDesigner
KEGGconverter	yes	yes	yes	yes	yes		yes		independent

Table 1: a comparison of features table regarding the applications mentioned in the case study

4.2. GOrevenge

4.2.1. Retrieving hub genes with GOrevenge web application

As a case study for the operation of the algorithm and the results from the corresponding web application, we run the BubbleGenes procedure, having as parameters: organism -> chicken, aspect -> BP, distance->Resnik, relaxation->0.15, and as initial probing genes: `PSEN1, CAV1, TLR7.` The interface of the web application is displayed in Figure 11, along with the available options for the tuning of the searching. The user has the option of providing an email address, as to receive an extensive list of genes (1.000 genes), as well as, the relative GOterms found by the algorithm, and those that were not discovered.



Organism:	chicken	
Aspect:	BP	
Distance:	graph	resnik is A LOT heavier to compute!
Algorithm:	Bubble Genes	
Relaxation:	1	
Input:	PSEN1, CAV1, TLR7	Enter genes for 'Bubble Genes', or GO terms (without the 'GO:' prefix) for 'Bubble GO', separated with white spaces, comma, or semicolon (allthough not in combination!)
Email:		Enter a valid email address if you'd like to receive all the results along with observed GO sets

Figure 11: GOrevenge web application interface

The top 20 scoring genes are presented in Table 2. The output columns of the Table are:

- In: is the indication whether the gene was also part of the input set of genes.
- Gene: the top scoring genes according to GOcount.
- GOcount: the number of GO terms that have been discovered by the algorithm (always a subset of GOcount_orig).
- Prune: the pruned values of GOcount considered under certain distances.
- GOcount_orig: the initial count of GOterms for the respective gene, and aspect, according to the annotation of the organism (GOA).

in	gene	GOcount	prune[0.3]	prune[0.6]	prune[0.9]	GOcount_orig
1	PSEN1	64	63	46	11	64
1	CAV1	46	44	24	8	46
0	BMP4	30	28	18	4	98
0	SHH	29	28	20	5	111
0	Q9YGX6	29	29	15	5	81
0	PSEN2	28	28	23	6	33
0	PS2	28	28	23	6	33
0	Q90Z44	24	22	10	4	40
0	TLR4	22	17	9	3	53
0	BCL2	22	20	17	3	99
0	SNCA	21	17	11	3	44
0	chBcat	20	19	15	5	96
0	ChALK5	19	19	13	3	43
0	il-1beta	17	16	9	2	38
1	TLR7	16	13	6	3	16
0	SFRP1	16	16	10	3	44
0	BMP7	16	16	10	3	45
0	BMP2	16	16	12	2	43
0	akt1	15	15	11	4	40
0	Q9IA31	15	12	11	3	45

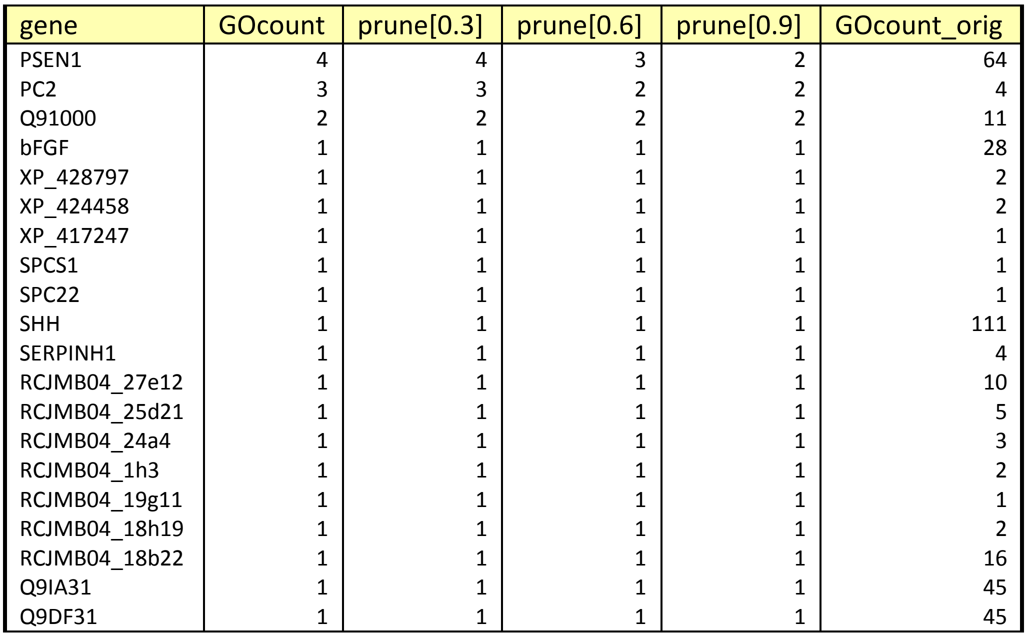
Table 2: Top Resnik-BubbleGenes for the genes PSEN1,CAV1,TLR7

Running the algorithm with the same parameters but using Graph-BubbleGenes this time produced the results shown in Table 3.

in	gene	GOcount	prune[3]	prune[6]	prune[9]	GOcount_orig
1	PSEN1	63	58	58	58	64
1	CAV1	46	43	43	43	46
0	PSEN2	29	29	29	29	33
0	PS2	29	29	29	29	33
0	SHH	19	18	18	18	111
0	BMP4	18	17	17	17	98
0	BCL2	18	17	17	17	99
1	TLR7	16	16	15	15	16
0	TGFB2	14	10	10	10	51
0	Q9YGX6	14	14	14	14	81
0	RCJMB04_19d10	13	13	12	12	13
0	BMP7	13	13	13	13	45
0	chBcat	12	12	12	12	96
0	TLR4	12	12	11	11	53
0	SMAD3	12	12	12	12	44
0	SFRP1	12	12	12	12	44
0	Q9IA31	12	12	12	12	45
0	Q9I9P1	12	12	12	12	44
0	Q9DF31	12	12	12	12	45
0	Q0PQ88	12	12	11	11	23

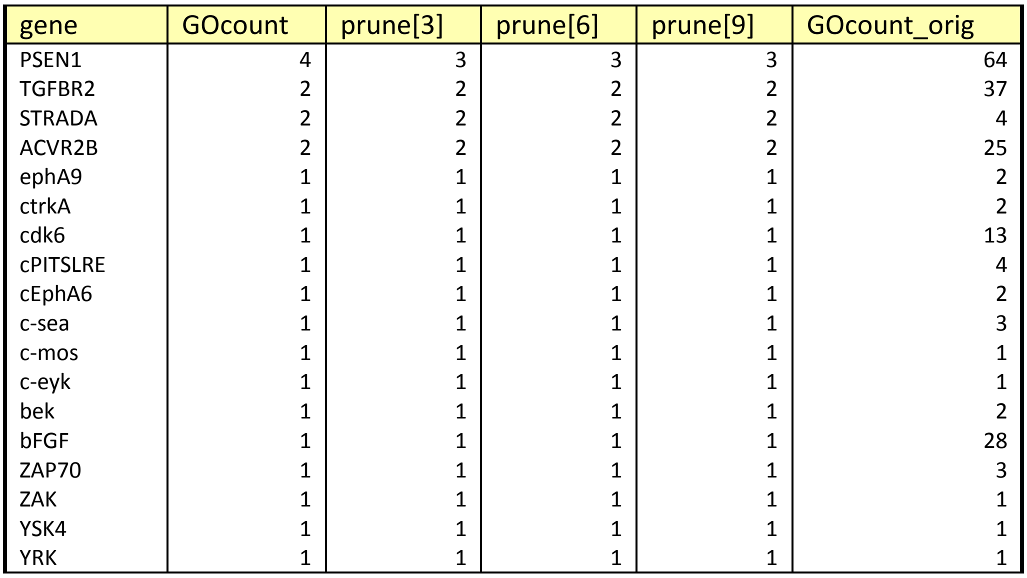
Table 3: Top 20 Graph-BubbleGenes for the genes PSEN1,CAV1,TLR7

We notice that the predefined prune distances for the Resnik case are: 0.3, 0.6, 0.9 (where the range is between 0 and 1), while for the Graph case are: 3, 6, 9 hops. Next we run the Resnik-BubbleGO having as input the BP GO terms: GO:0000186 (activation of MAP kinase kinase activity), GO:0016485 (protein maturation by peptide bond cleavage), and GO:0016080 (synaptic vesicle targeting). The results are in Table 4 and Table 5.



gene	GOcount	prune[0.3]	prune[0.6]	prune[0.9]	GOcount_orig
PSEN1	4	4	3	2	64
PC2	3	3	2	2	4
Q91000	2	2	2	2	11
bFGF	1	1	1	1	28
XP_428797	1	1	1	1	2
XP_424458	1	1	1	1	2
XP_417247	1	1	1	1	1
SPCS1	1	1	1	1	1
SPC22	1	1	1	1	1
SHH	1	1	1	1	111
SERPINH1	1	1	1	1	4
RCJMB04_27e12	1	1	1	1	10
RCJMB04_25d21	1	1	1	1	5
RCJMB04_24a4	1	1	1	1	3
RCJMB04_1h3	1	1	1	1	2
RCJMB04_19g11	1	1	1	1	1
RCJMB04_18h19	1	1	1	1	2
RCJMB04_18b22	1	1	1	1	16
Q9IA31	1	1	1	1	45
Q9DF31	1	1	1	1	45

Table 4: Top Resnik-BubbGO genes for the GOterms: 0000186, 0016485, 0016080



gene	GOcount	prune[3]	prune[6]	prune[9]	GOcount_orig
PSEN1	4	3	3	3	64
TGFBR2	2	2	2	2	37
STRADA	2	2	2	2	4
ACVR2B	2	2	2	2	25
ephA9	1	1	1	1	2
ctrkA	1	1	1	1	2
cdk6	1	1	1	1	13
cPITSLRE	1	1	1	1	4
cEphA6	1	1	1	1	2
c-sea	1	1	1	1	3
c-mos	1	1	1	1	1
c-eyk	1	1	1	1	1
bek	1	1	1	1	2
bFGF	1	1	1	1	28
ZAP70	1	1	1	1	3
ZAK	1	1	1	1	1
YSK4	1	1	1	1	1
YRK	1	1	1	1	1

YES1	1	1	1	1	2
XP_429192	1	1	1	1	2

Table 5: Top Graph-BubbGO genes for the GOterms: 0000186, 0016485, 0016080

4.2.2. Performance evaluation

The evaluation procedure described in 3.2.2 was applied to the three aspects of GO (BP, MF, CC) utilizing all annotation evidence codes, using both graph and Resnik distances, at 6 different agglomeration steps. Figure 12 shows the boxplots for the medians of the indexes of targeted cancer genes, having observed degree (e.g. annotating GO terms) greater or equal to 5 at the MF aspect. For the agglomeration step the Resnik-BubbleGO algorithm was used with distances: 0, 0.1, 0.2, 0.3, 0.4, and 0.5. Calculating the Wilcoxon signed rank test for distance 0, gives: V=55, p-value=0.001953.

At Figure 13 the Resnik-Pruning algorithm was applied, for the agglomeration step of 0, which top performed when compared with other distances. The pruning was made at normalized distances: 0, 0.3, 0.6, and 0.9, where the larger values correspond to more dissimilar terms. Not all executions produced statistically significant results, yet for several cases, like for instance for MF Resnik distance with observed degree ≥5, statistical significance was observed. Likewise, the Graph-BubbleGO algorithm for the CC aspect, at the agglomeration step 0 gave: V=49, p-value=0.002734, while the same algorithm for the BP aspect, at the agglomeration step 4 gave: V=52, p-value=0.009766.

There were cases where the control median was lower than those derived when applied the agglomeration steps, as in CC using edge and Resnik distances (having observed degree ≥5). Possibly, the varying descriptive depth of certain branches of the ontological tree might be related to the performance variation observed.

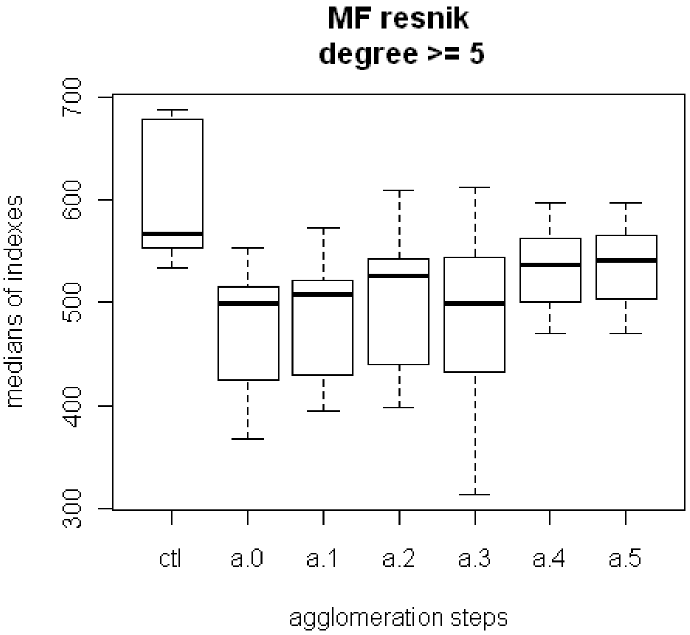


Figure 12: Boxplots for the medians of indexes for target cancer genes having observed degree ≥5. Ctl: is the boxplot for the indexes of the control list. GO aspect: Molecular Function.

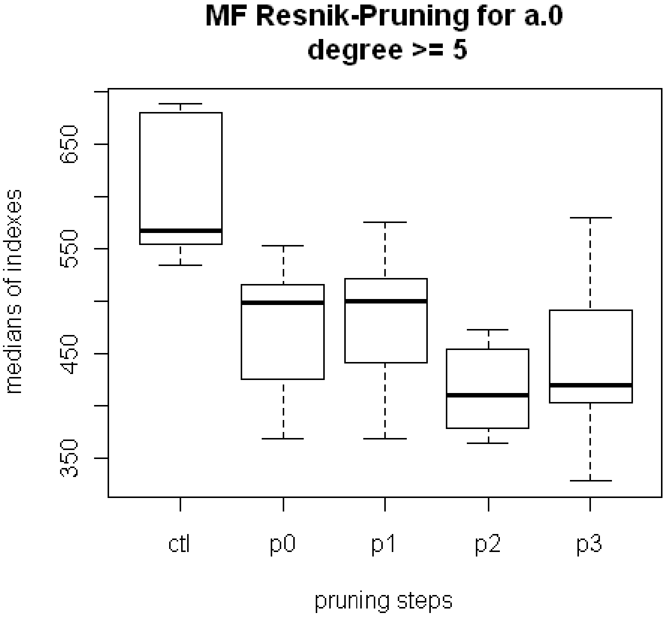


Figure 13: Application of the Resnik-Pruning algorithm to the results of Figure 12, for an agglomeration step of distance 0. Best performance is observed when pruning at step p2.

At Figure 14, the average ROC curves are shown for the 3 GO aspects (derived by using [109]). A binary classifier evaluates the performance of GOrevenge, in retrieving genes from the 90% remaining genes set, in the complete ranked list of genes, for the total false-positive rate range (for an agglomeration step of distance 0, without pruning). The AUC (Area Under Curve) value for MF is 0.93, clearly supporting the overall good performance of the algorithm, in terms of gene retrieval. On average, use of Resnik distances outperformed Graph distances, as shown in Figure 14.

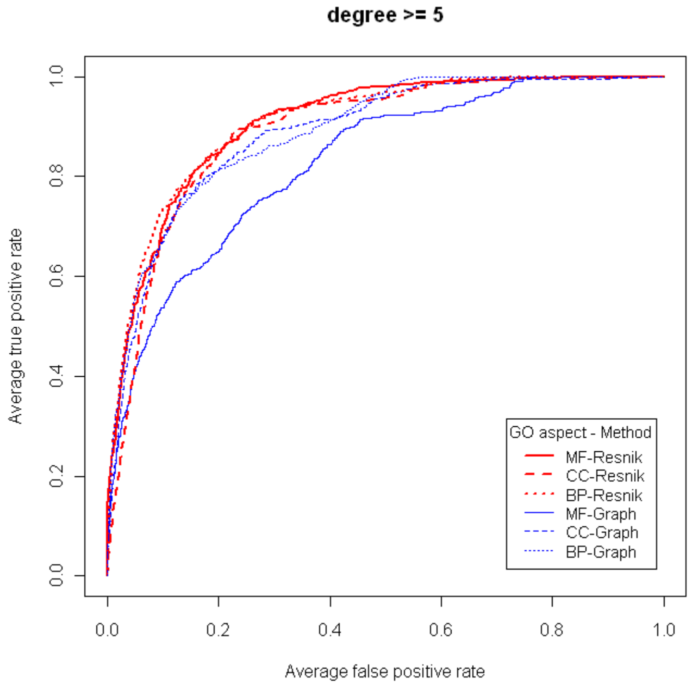


Figure 14: Average ROC curves, from the ranks of the target cancer genes having observed degree ≥5

4.2.3. Biological evaluation of the results

The first 15 rows in the resulting gene list, derived from the whole pancreatic cancer gene set, are shown in Table 6. In parentheses of the column labels are the parameters used in the execution of the algorithm. E.g. (R,0) means the Resnik-BubbleGenes procedure for agglomeration distance 0.

gene	MF (R,0)	BP (G,4)	CC (G,0)	TOTAL	*input*	GO BP Exp. Annotations
CTNNB1	1	5	5	11		07155 08285 10552 10909 16055 16337 16481 30997 32355 34333 34394 42493 43065 44334 44336 45768 45893 45941 48660 51149 60070 61154 70602 71363 71681 90279 2000008
TP53	3	14	6	23	Y	06289 06355 06461 06915 06974 06978 06983 07050 07265 07275 07569 08104 08635 10332 10552 10670 30308 30330 31571 42149 42771 42981 45941 45944 71479 90399 90403
EGFR	43	62	11	116	Y	07165 07166 07173 08284 16337 30335 31659 35413 42327 43406 45429 50679 50730 50999 51897 70141
ERBB2	10	113	21	144	Y	07165 07166 07169 14065 42060 43406 45785 46777 50679
INSR	5	59	89	153		00187 01934 03007 07186 08284 08286 18108 19087 30335 32147 32148 32583 32869 42593 43410 45429 45725 45740 45821 45840 45995 46326 46777 48639 51290 51897 60267
P2RX7	73	16	81	170		10524 33198 46931 50718 51899
SNCA	37	51	112	200		01921 06916 06919 10040 10517 10642 31115 31623 32026 32410 32496 32769 34341 35067 43154 45807 45920 51281 51585 51612 51622 55074 60732 70495 70555 71902
SMAD3	20	19	207	246		00122 01666 01933 06810 06917 06919 06955 07050 07179 07183 10718 17015 19049 30308 32909 35413 42177 42993 45216 45930 45941 45944 50821 90263
NOS3	143	80	47	270		03100 06527 06809 31284 34405 43542
CAV1	261	17	2	280		09267 16050 30193 30514 31295 31623 32091 32570 33137 33138 43627 51480 70836 90263
PRKCA	205	56	45	306	Y	35408 70555
TP63	68	55	192	315		06915 43193 45747 45892 45893 51289
AXIN1	61	91	170	322		01934 10800 30178 31398 32147 33138 43623 45732 45941 51443
VEGFA	127	22	184	333		01525 01666 01938 02575 08284 08360 10595 30335 30949 43066 43117 43536 45766 48008 50731 50927 50930 51272 60754 71456
JAK2	27	47	263	337		35409 46677

Table 6: Gene prioritization using the sum of the indexes for each GO aspect. The 'input' column marks genes present in the input gene set. The last column presents GO BP annotations for each gene, having experimental evidence codes (prefix ‘GO:00’ is removed)

Table 6 and Table 7 demonstrate the ability of GOrevenge to rank genes relative to the input set and to suggest new candidate hub genes. Although the number of the input genes differs significantly in the two cases, the two tables contain several common molecular players, with an established role in the molecular physiology of cancer. Among the relevant BP terms are these related to: regulation of apoptosis, cell proliferation growth and migration, DNA damage response, Ras and Wnt signalling pathways. 8 out of 9 genes of T-cell ALL are also present to the pancreatic cancer set. Nevertheless, none of the 8 common genes appear among the top rows of both tables.

gene	MF (G,3)	BP (G,3)	CC (G,3)	TOTAL	input
CTNNB1	1	4	3	8	-
TP53	2	14	16	32	-
ERBB2	12	88	49	149	-
EGFR	83	53	36	172	-
CAV1	166	39	5	210	-
INSR	11	75	127	213	-
AXIN1	33	184	20	237	-
BCL2	168	2	87	257	-
SOD1	188	65	24	277	-
P2RX7	139	30	113	282	-
SMAD3	8	17	289	314	-
CALR	32	266	52	350	-
GRIN2B	227	141	35	403	-
NOS3	212	73	120	405	-
SIRT1	99	128	179	406	-

Table 7: GOrevenge results having as input the ALL gene set and using the Graph-BubbleGenes algorithm

4.3. Cutaneous Melanoma Results

The results of the trials regarding the feature selection process for the melanoma datasets are depicted at Figure 15. Regarding the median value of optimum performances, in all cases an almost perfect score was achieved in the case of the unified datasets. Only-image dataset (oi) exhibited the lower performance and the higher subset numbers.

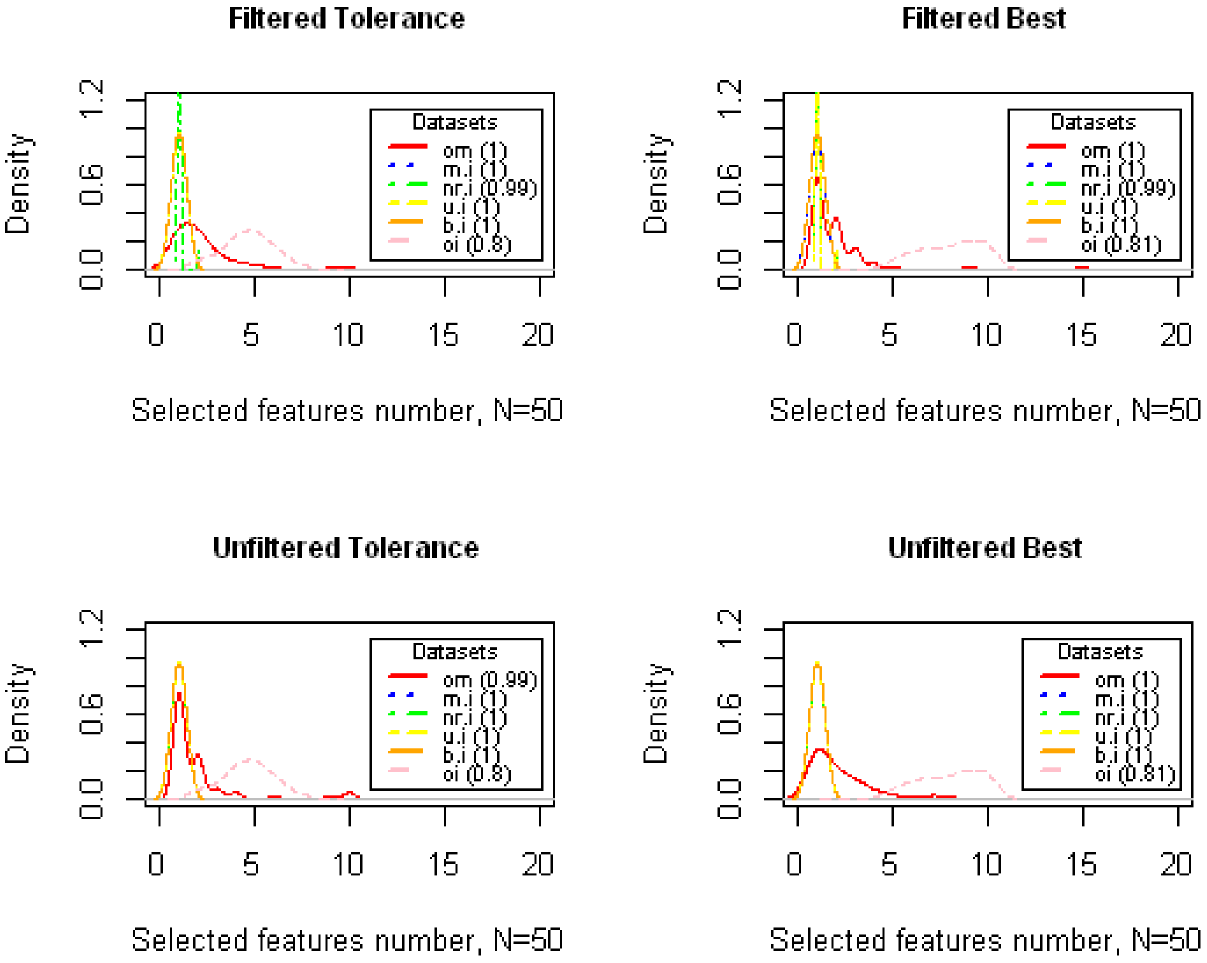


Figure 15: Density plots of the optimum features number from 50 repetitions. The six datasets are: only microarray (om), mean imputation (m.i), normal random imputation (nr.i), uniform imputation (u.i), bootstrap imputation (bi), and only image (oi). In parentheses are the medians of the obtained performances (auc) for each dataset.

The application of the co-linearity reduction filter reduced the dispersion of the optimum subset number. Furthermore, the execution time in the reduced dataset was 4 times faster, analogous with the remained feature number after the use of the filter (482 from the initial 1701 differentially expressed genes in the microarray dataset).

The results on the imputed datasets exhibited also a minimization of the dispersion of the subset numbers. In addition, the four imputed datasets presented almost the same distribution. Nevertheless this similarity ended when we compared the gene sets retrieved in each case. As shown Table 8 and Table 9, the normal random imputation dataset (nr.i) resulted in a considerably more stable selection of features comparing to the mean imputation unified dataset (m.i). The same pattern is observed for the unfiltered cases too. In the unfiltered cases (Table 10 and Table 11), the nr.i dataset exhibited far better stability in the predictors' selection even to the microarray-only dataset.

Feature **(om)**	*Freq.* **(om)**	*Feature* **(m.i)**	*Freq.* **(m.i)**	*Feature* **(nr.i)**	*Freq.* **(nr.i)**
CDC37L1	47	NEIL1	4	CDC37L1	49
RRAS2	34	IFI16	3	RRAS2	2
SLC7A8	18	CTDSPL	2		
HPCAL1.1	14	DLK2	2		
IFT81	8	NADK	2		
SSBP2	6	OR2A20P	2		
GIPC2	5	PIK3C2G	2		
CTDSPL	3	ABCD1	1		
NEIL1	1	ACADL	1		
PLCH2	1	ARMC9	1		

Table 8: Top Features (genes) selected after 50 repetitions of the 10-Fold Cross-Validation modelling for the Best-Filtered case in each of the three datasets

The features resulted from the mean imputation unified dataset presented high instability, and though proved as the least preferable option to the imputation procedure. The other two imputation methods (u.i and b.i) performed equally well to the nr.i method.

The deficiency of using only performance indicators for marker discovery has been noted in the literature [110] and this is consisting with the findings of this study. The measure of stability of feature selection results with respect to sampling top selected features at the tolerance-filtered case variations provides higher confidence in discovered biomarkers.

Feature **(om)**	*Freq.* **(om)**	*Feature* **(m.i)**	*Freq.* **(m.i)**	*Feature* **(nr.i)**	*Freq.* **(nr.i)**
CDC37L1	45	PARD3	5	CDC37L1	40
RRAS2	25	ACOT9	3	RRAS2	6
SLC7A8	17	CYP4F3	3	HPCAL1.1	2
HPCAL1.1	10	FZD10	3	SSBP2	2
IFT81	6	NEIL1	3		
GIPC2	5	ACADL	2		
CTDSPL	4	MTUS1	2		
NEIL1	4	PER3	2		
SSBP2	3	PPP2R3A	2		
SMAD5OS	2	SMAD5OS	2		

Table 9: Top Features selected at the Tolerance-Filtered case

But in this case, the imputations with the nr.i, u.i, and b.i methods, acted on the unified dataset as an additional variation factor. This additional variation resulted in the retrieval of smaller optimum subsets of features, consisting of fewer re-occurring genes as possible biomarkers.

Feature **(om)**	*Freq.* **(om)**	*Feature* **(m.i)**	*Freq.* **(m.i)**	*Feature* **(nr.i)**	*Freq.* **(nr.i)**
CNIH3	18	EPHX2	2	ABLIM1	23
KLHDC2	15	ABCD1	1	GDF15	12
MYO1D	15	ACOT7	1	LRRC59	4
PKM2	14	AHNAK	1	HLF	3
BAG1	12	AHNAK2	1	SPP1	3
ABLIM1	8	AMPD3	1	HEY1	2
PHACTR1	5	BAT2.1	1	MAOA.2	2
MAOA.2	4	CALU	1	NEBL	1
ECHDC2	3	CAPNS1	1		
GDF15	3	CDC37L1	1		

Table 10: Top Features selected at the Best-UnFiltered case

Feature **(om)**	*Freq.* **(om)**	*Feature* **(m.i)**	*Freq.* **(m.i)**	*Feature* **(nr.i)**	*Freq.* **(nr.i)**
PKM2	16	C1orf46	2	ABLIM1	30
MYO1D	15	CBX7	2	GDF15	8

CNIH3	13	LY6D	2	HLF	4
BAG1	11	ABLIM1	1	CNIH3	3
KLHDC2	11	ACOT7	1	HEY1	2
ABLIM1	9	AEBP1	1	CTSB	1
ECHDC2	3	AHNAK	1	HLF.1	1
GDF15	3	ALDOC	1	LRRC59	1
PPIB	3	ARIH2	1		
PHACTR1	2	ATP6V0C.1	1		

Table 11: Top Features selected at the Tolerance-UnFiltered case

Notably, none of the image-derived features were present to the top selected features of the unified datasets, as seen at the above. In order to assess the importance ranking of image-features, 50 repetitions of the random forest algorithm were run for the unified dataset imputed by the four methods (m.i, nr.i, u.i and b.i). Each of the resulted 50 lists of features was sorted by decreasing importance. Next, the positions of the image-features in the lists were collected and the density plots for the filtered/unfiltered cases are shown at Figure 16. Random forest avails four importance measures [111] and in this case the "MeanDecreaseGini" criterion was chosen. The results using the other three criteria were similar.

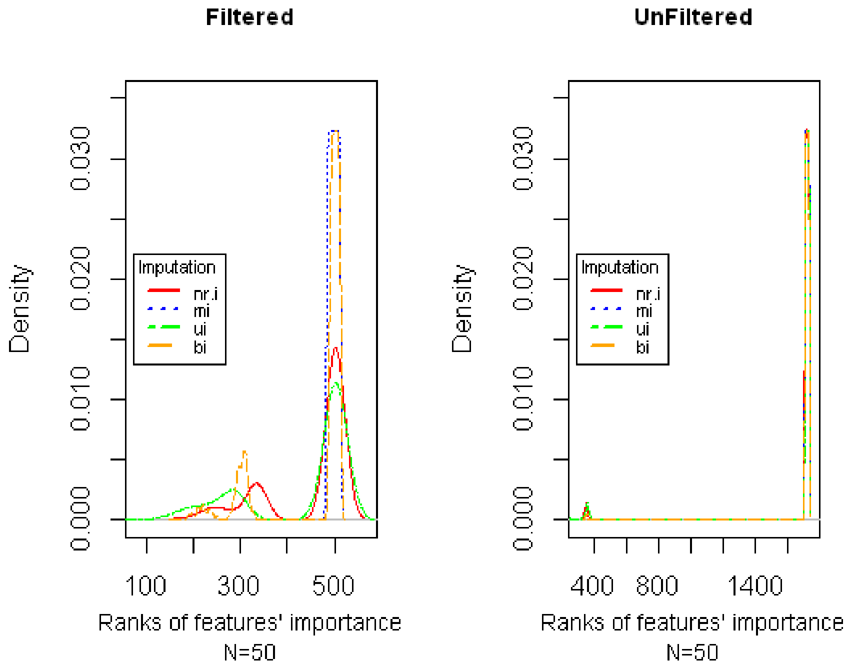


Figure 16: Density plots of importance ranks for image-derived features (ranking in the x-axis is in decreasing order of importance).

The majority of the image-features ranked as less important when compared to the microarray features. This implies their lower informative power with respect to the total observed variation in the integrated dataset, probably due to the fact that technical covariance but also size, leave their fingerprint in the integration process, despite the application of normalization techniques, thus inflicting their effect on the response vector of the disease. When using the nr.i, u.i, or b.i methods however, a better performance of the image features is observed, which is captured as their more frequent presence in higher positions of the classifier's vector, in discord with the results of the m.i method. Mean imputation process resulted in scoring all image features in the lowest positions of the complete feature set, considering them less informative compared to the microarray features. In this sense, it is obvious that the three imputation methods: nr.i, u.i and bi yield a more impartial effect, as can be surmised from the improved score of the image related features, providing practical value to its application in the integration process, as the simulated dataset thus derived, is a more realistic representation of the real one.

In order to assess whether the small size of the image features (31 variables) compared to the 482 microarray features (after the co-linearity filtering from the initial 1701 microarray features) is the reason that rf algorithm consistenly favors microarray predictors as best performers, we run again the procedure as described for the production of Figure 16. This time in the unified datasets we replicated the image features 10 times adding thus 310 replicated image features, in order to balance in size the microarray features. The results depicted again the same preference to the microarray features, excluding thus the case that the behaviour shown at Figure 16 was due to the feature size imbalance.

At Figure 17 the scores plot is depicted using the first two principal components for all datasets. The first principal component (which represents the direction of the larger data variation) can discriminate in most cases the melanoma/non-melanoma classes quite well. Comparing the graphs of the PCAs we note that although PCA on only-image data needs more than 2 PCs to discriminate the two classes, in all the other cases PC1 can separate the classes except for 2-3 samples, all coming from the microarray dataset (non-crossed symbols). The three imputation methods (nr.i, u.i, b.i) presents similar results regarding the degree of separation of the classes, as well as the percentage of variation covered by the two PCs.

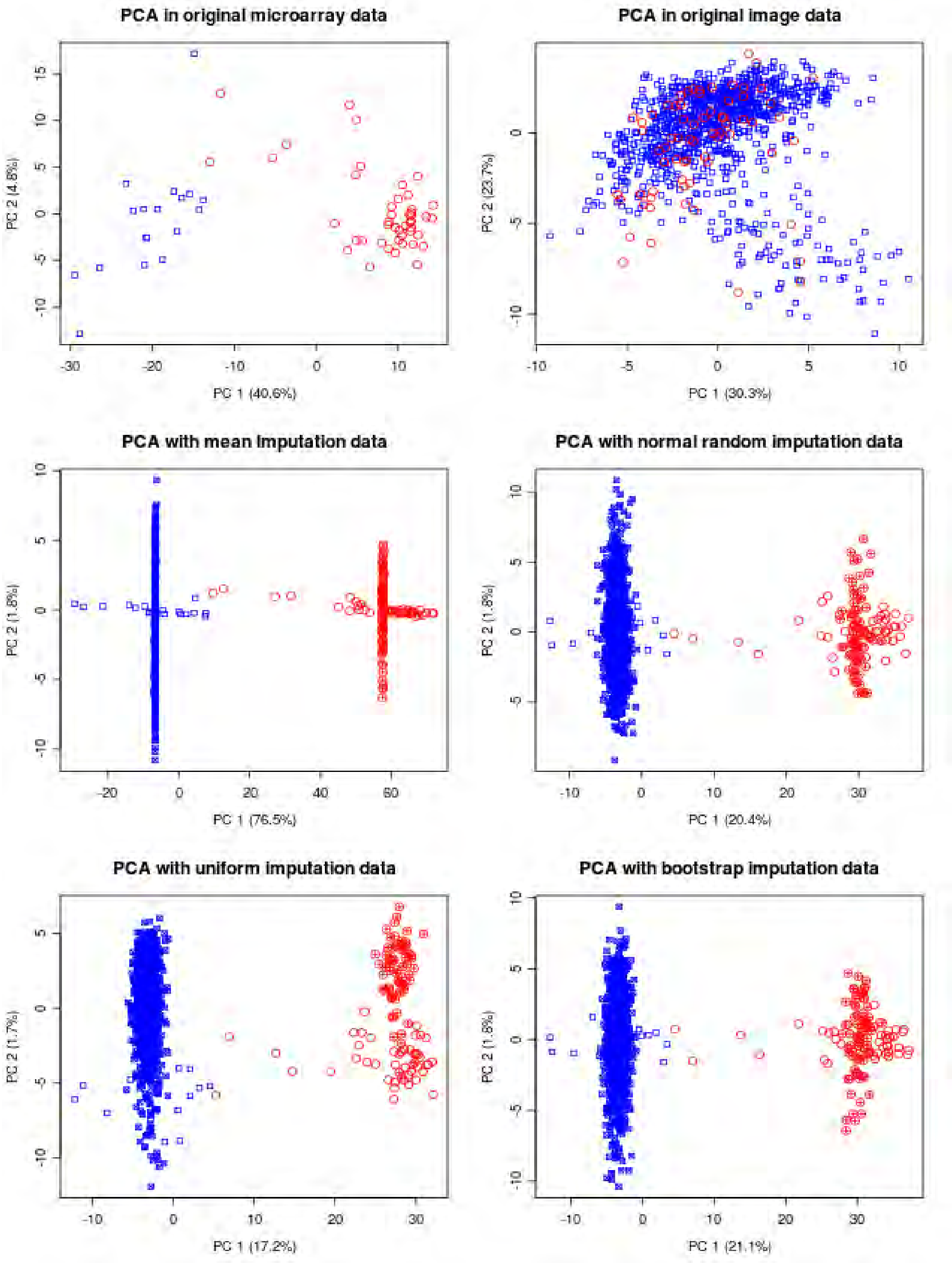


Figure 17: Scores plot, for all datasets. In parentheses is the percentage of variation covered by each principal component. With circles are the melanoma samples, and crossed (either circles or rectangles) are the image data points.

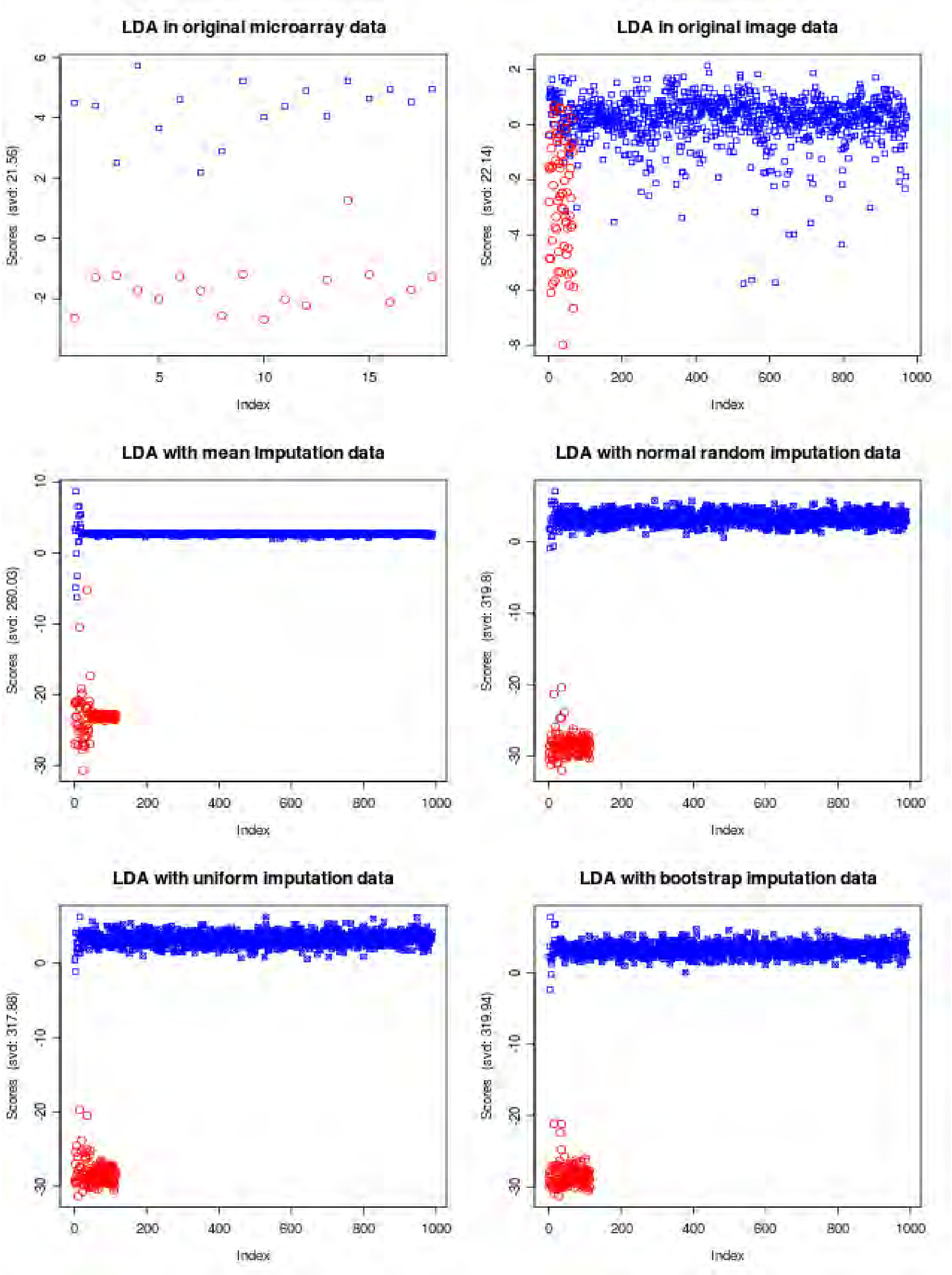


Figure 18: LDA scores plot for the nr.i dataset. With circles are the melanoma samples, and crossed (either circles or rectangles) are the image data points.

Comparing the graphs of the LDAs this time, for all the datasets (Figure 18) it is obvious that the imputation process increases significantly the discrimination between the two classes, as denoted by the svd values. The mean imputation process offers the smaller increase of the svd value relative to nr.i, u.i and b.i procedures.

The running of the `lda` function 50 times in LOO crossvalidation mode for each of the datasets, provided the accuracy distributions of a predictive model for the case of melanoma class. Accuracy is the percentage of correct guesses. As it is shown at Figure 19, the accuracy for the nr.i, u.i and b.i datasets is very high (over 95%).

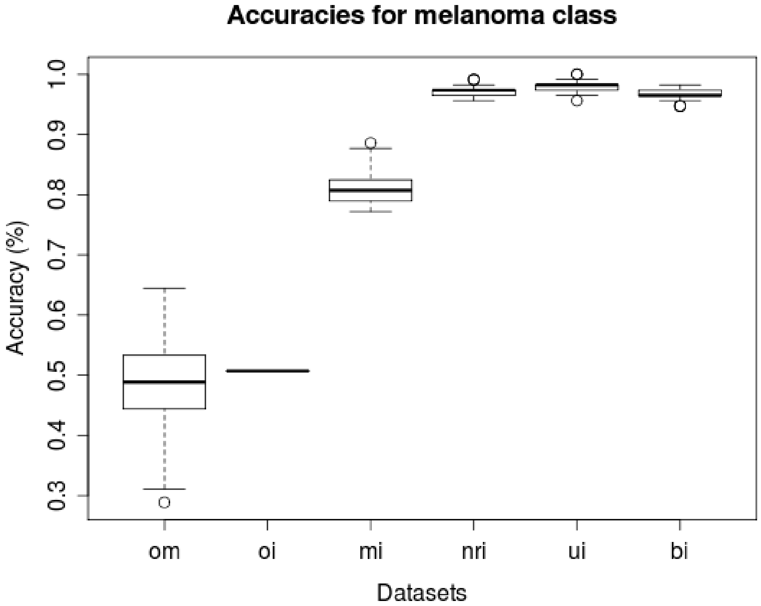


Figure 19: LDA CV accuracy for the melanoma class for each dataset (N=50)

The mean imputation unified dataset exhibits again the worst performance comparing to the other three imputation methods. There is a significant improvement in the attained LDA accuracy regarding the accuracy of the original only-microarray and only-image datasets.

As it was with the high perfomant features retrieved by the random forest procedure, stability of the features suggested by the LDA is of high importance. For this reason, we run the `lda` function 100 in CV=False mode for each unified dataset, and recorded the LDA coefficients (loadings). From these data we retrieved two ranking indicators: the top 20 features with the largest mean coefficient (topmeans) and the top 20 features which appeared most of the times having the top 20 largest coefficients (top20). The results are shown at Table 12.

om	oi	mi topmeans
ABCD1	I.mean	I.mean
CRIM1.1	mean.R	mean.R
UPP1	mean.G	mean.G
GSTT1	mean.B	mean.B
HPCAL1.1	A.mean	A.mean
ZSCAN18	L.std	L.std
IFI16	I.std	I.std
SHARPIN	S.mean	S.mean
TMEM80	std.R	std.R
HOMER3.1	S.std	S.std
PLAUR.1	A.std	A.std
SNTB1	std.B	std.B
SFRP1.1	std.G	std.G
MTUS1	B.mean	ABCD1
RPRM	L.mean	B.mean
FZD7	H.std	UPP1
SRC	H.mean	L.mean
MMP1	Distance.std	HPCAL1.1
LAMB4	PERIMETER	ZSCAN18
NEK2	COMPLEXITY	IFI16

nri topmeans	nri top20	nri top20 freq.	*ui topmeans*	*ui top20*	*ui top20 freq.*	*bi topmeans*	*bi top20*	*bi top20 freq.*
HPCAL1.1	L.mean	78	HPCAL1.1	HPCAL1.1	75	CDC37L1	L.std	77
MMP1	mean.G	75	Grad.std	mean.B	69	MMP1	I.std	74
CDC37L1	I.mean	71	CDC37L1	I.mean	63	HPCAL1.1	I.mean	70
C2orf68	I.std	70	Grad.mean	mean.G	61	ZSCAN18	mean.G	70
A.mean	L.std	68	MMP1	Grad.std	59	C2orf68	L.mean	69
ID4	mean.B	63	ID4	L.std	59	ALOX12	A.mean	68
ZSCAN18	std.B	61	SSBP2	std.B	58	PARD3	mean.B	67
MTUS1	std.G	61	NRP2	I.std	57	ID4	mean.R	66
SSBP2	mean.R	60	IFT81	L.mean	56	MTUS1	S.mean	61
ALOX12	A.mean	59	FOXO1	CDC37L1	55	IFT81	std.B	61
NCAPH	GMSM.mean	54	ZSCAN18	A.mean	53	NCAPH	std.G	59
ANG	PERIMETER	54	RRAS2	B.std	53	NRP2	std.R	59
std.B	A.std	53	GPRIN2	std.G	52	CYP3A5.1	PERIMETER	58
NRP2	HPCAL1.1	53	MTUS1	PERIMETER	51	SSBP2	DISSIMILARITY	54
PARD3	S.mean	52	ALOX12	Grad.mean	49	ANG	A.std	50
CYP3A5.1	B.mean	50	mean.B	std.R	48	FOXO1	CDC37L1	50
ABCD1	std.R	50	NCAPH	A.std	45	C12orf29	GMSM.mean	49
BAZ1B	COMPLEXITY	48	NEIL1	mean.R	45	BAZ1B	B.std	47
C12orf29	S.std	48	IFI16	DISSIMILARITY	44	ABCD1	B.mean	43
IFT81	B.std	45	SLC7A8	S.mean	44	MAOA.1	COMPLEXITY	43

Table 12 (a and b): Top 20 LDA features after 100 repetitions

At Table 12 (a) the three first columns (om, oi, and mi topmeans) are retrieved after only one run, since there is no variation in the LDA coefficients as the dataset is stable in these cases. The top 20 features for om, oi and mi have been included though as to have a full view of the best variables at each dataset, original (om, oi) or unified. As it is obvious from the frequencies columns, the three imputation methods (nri, ui, and bi) show similar distribution. In order to assess the method which presents the better stability, we used as *stability indicator* (si) the number of common features between topmeans and top20 column for each method. The si for nri, ui and bi was 2, 5 and 1 respectively.

As a last step we assess the predictive performance of the top features (ui topmeans), considering LDA and RF algorithms to all the datasets. For the cases of the original datasets (om, oi) only the relative part of the biomarkers was used as predictors. The results are shown at Figure 20. A perfect score is achieved by random forests for all the datasets apart to only-image original dataset. LDA reports an almost perfect score too, apart to oi dataset in which it cannot guess correctly any of the melanoma samples.

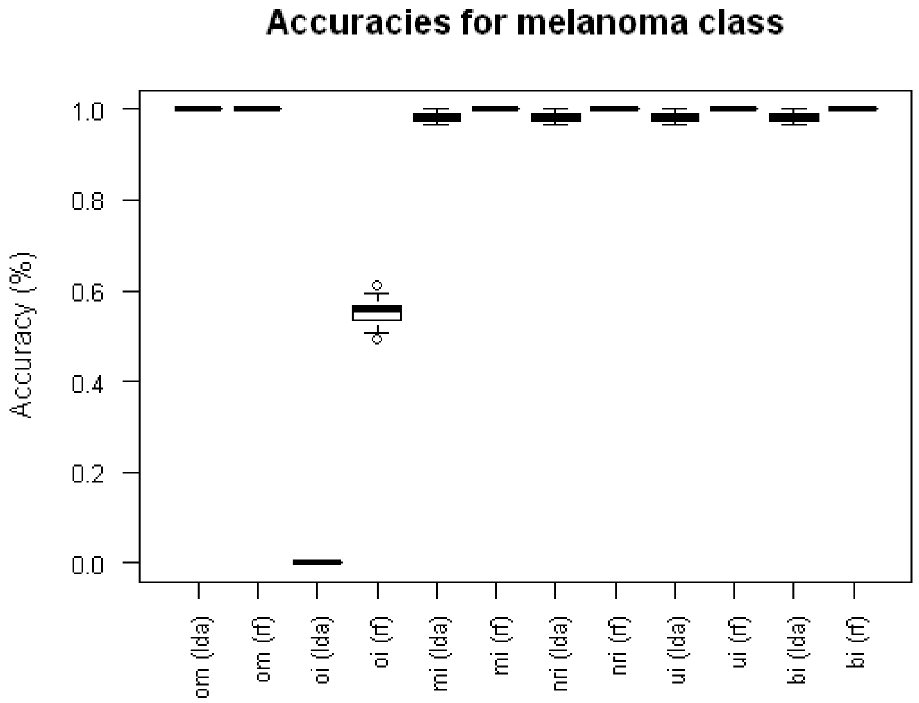


Figure 20: LDA LOO CV and RF OOB performance using ui topmeans biomarkers (N=50)

CHAPTER 5. DISCUSSION AND CONCLUSION

In response to the research questions of this study, three novel methodologies have been developed. Each targets a different aspect of biomedical modelling and, thus, contributes to the process of creating integrated physiological models. We will conclude this work with a synopsis of the outcomes, the novel contributions and proposed future work directions.

5.1. KEGGCONVERTER

KEGGconverter, a de novo developed software application implementing a novel algorithmic workflow for exploiting the valuable KEGG metabolic information related to many organisms, is capable of producing integrated metabolic pathway models ready for simulation purposes by taking just KEGG KGML files as input. KGML format cannot carry information regarding the stoichiometry and the kinetic properties of the reaction equations of the illustrated pathway, an essential prerequisite for constructing a simulated biochemical reaction model. Manually adding the kinetic laws in biochemical models represents a time exhaustive effort, depending on the size of the pathways to be modeled (i.e. encompassed reactions, substrates and modifiers). Moreover, the orientation of the KGML models residing in the KEGG pathway database mainly for visualization purposes has serious pitfalls regarding the biochemical accuracy of the described models for simulation purposes. In order to attain a nicer graphical layout, in terms of clarity of the graph, many compounds and reactions are included more than once in the same pathway. This is a problem with respect to the biochemical consistency of the model, given the fact that all reactions take place in the same cell compartment, unless it is a multi-compartmental model, which results in erroneous stoichiometries for the respective reaction network. Also, ambiguities in naming the compounds in a model, resulting from unresolved inconsistencies in nomenclature, complicate and incur additional flaws in quantifying metabolic pools (i.e. concentration of an enzyme).

KEGGconverter automatically transparently and seamlessly handles the fusion, conversion to SBML, proper renaming of the species, stoichiometric correction of the integrated pathway and insertion of reaction specific default kinetic laws, in a straight forward manner compared to other procedures mentioned in the case study. KEGGconverter integrates, for the first time, the aforementioned aspects, producing an SBML model as output that can be tested in a simulation environment and can be used as a foundation for further improvement and accurate tuning of the model. The addition of the kinetic equation layer is accomplished by invoking an in-house developed kinetic library, exploiting the reaction type (reversible or non-reversible) and the stoichiometric relationships of reactants, products and modifiers. The automated introduction of four case-specific default kinetic mechanisms in the models is directed by a rule-based algorithm that we implemented for the specific task. The final models derived from KEGGconverter do not include trivial metabolites (which result from reproducing inconsistencies of the KGML visualization-oriented simplified information pattern) but at the same time they contain all of the available information regarding the number of included reactions in each pathway. Furthermore, additional reactions to neighbouring pathways are constructed, indicating the direction of metabolic flows in the network and, thus, providing better stability in the boundary conditions of the models. This approach produces more accurate and dynamically reproducible models. Since the creation of the tool, the Institute for Chemical Research from the Kyoto University has contacted us. Recognizing the high conversion accuracy of the tool, the Institute wanted to improve its capabilities regarding the CellDesigner annotations in the created SBML file and to pinpoint specific cases where the conversion functionality could be further advanced. (CellDesigner is being developed by the Systems Biology Institute in Japan.)

5.2. GOREVENGE

The novel GOrevenge algorithm accepts a list of GO terms or of genes as input, and utilizes the duality of the mapping between genes and GO terms to highlight interesting hub genes. These genes, despite their products participating in several molecular functions and biological processes, may have been excluded from the initial significant gene list for numerous reasons, such as the lack of statistical power from a low number of replicates. This method also represents a sanity check, in order to trace back interesting targets from the differential expression single genes list, as it is capable of tracing back statistically significant genes, based not only to their statistical score, but also through the exploitation of the functional knowledge related to them, as captured by the GO. As a means of considering potential kinship among genes for discovery and prioritization purposes, GOrevenge's stepwise biphasic mechanism starts from the initial gene set considered and its corresponding GO terms. In the first (agglomeration) phase, genes are collected when linked not only to a given GO term but also to its neighbouring ones, i.e. its parents and children GO terms (according to a predefined distance). Then, for genes that are annotated by several terms, we consider an optional pruning phase, where GO terms are eliminated depending on the distance of the terms. The ranking of the discovered genes is performed by sorting the genes in descending order by the observed number of GO terms (e.g. the node degree or rank). GOrevenge incorporates Resnik semantic similarity metrics in addition to edge-based graph distances. The algorithm also has the ability to probe specific aspects of the GO (MF, BP, CC) for specific organisms, thus, enabling the application and evaluation of the algorithm in a wide range of experiments. Although in this work the algorithm has been showcased using the Gene Ontology for two cancer genes datasets (pancreatic cancer and T-cell acute lymphoblastic leukaemia), its applicability is generic and can accommodate other biological ontologies, genomes and versatile experimental designs. The selection of an appropriate distance metric in relation to the fitting agglomeration step determines the success regarding the rapid retrieval of target genes. Our method seems to act as a selective filter when considering the distribution of the degrees of the input genes. This could explain the variability of the results in the described test evaluation. Automating the selection of the optimal distance metric of the GOrevenge algorithm is a significant future task. Further examination of the fluctuation of the observed GO degrees regarding the initial input and examination of the study of the ‘saturation’ phenomenon, where the observed degrees of the resulting genes coincide with their original degrees from the GO annotations, are proposed. In addition, the application of the algorithm on different ontologies, e.g. disease ontologies such as the Medical Subject Headings (MeSH) (http://www.nlm.nih.gov/mesh/meshhome.html) or the Disease Ontology (http://www.disease-ontology.org), could provide new insights in the integration of phenological and molecular data. Students of the University of Central Greece have already made progress toward this goal, combining DO with the GOrevenge mechanism, which could further provide a way of inter-linking ontologies in the context of the integration of physiological models. Furthermore, the computation of the semantic similarities for all the GO terms in each aspect and for each organism is a heavy task, and, therefore, there are plans of transferring the GOrevenge web application to the Cloud as well as developing better interconnection with the GeneNetwork application from the University of Tennessee (http://www.genenetwork.org). The GOrevenge web application already provides a user-friendly shell, wrapping the inner algorithms of the application. This ensures easy access to and manipulations of its content as the whole process is transparently and seamlessly performed. XML-RPC web services have been implemented, offering easier integration with scripting procedures and pipelining with other web services.

5.3. BIOMARKERS FROM MULTI-MODAL, SEPARATE DATASETS

The fusion of multi-modal biomedical data is one of the key aspects in the context of VPH. The integration of separate datasets referring to the same disease that was developed in this study is an innovative approach, which can contribute significantly to the extraction of better biomarkers involved in various diseases and to the construction of more stable predictive models. The demonstrated improvement of the class discrimination between healthy and not-healthy samples, regarding the malignant cutaneous melanoma that was achieved for the unified datasets constructed by the imputation methods, involved dermatoscopy and microarray data originated from different patients for each modality and different experiments. Because of this, significant pre-processing steps were necessary, especially for the normalization of the raw microarray data. In addition, advanced machine-learning algorithms were used for the classification analysis endeavour, which involved linear and non-linear multivariate algorithms, such as LDA, PCA and Random Forests with stratified bootstrap sampling. Special care was given to the selection of matching experiments regarding the disease, so samples, preparation methods, and the subsequent analysis were in the same context. In the case of cutaneous melanoma in this study for instance, the microarray experiment selected was carefully screened among dozens residing in the GEO related to melanoma in order to match comparable state, progress and tissue taken for the disease (e.g. no artificial cell lines, not only metastatic tumours, etc.). The synthetic data created by the four imputation methods (‘mean value’, ‘random normal’, ‘uniform’ and ‘bootstrap’) were the integration mechanism for the assembly of the unified datasets, which were then further processed by AI methods. This innovative procedure allowed the study of the resulting unified dataset regarding the features’ contribution by each modality as well as the predictive performance assessment of the resulting top features. The biomarkers retrieved by this novel approach, employing the last three aforementioned imputation methods, achieved an almost perfect score for the prediction of the melanoma samples, both by LDA and Random Forests modelling. Although there is a plethora of freely accessible databases on various diseases, these datasets are mostly related to only one phenotypic dimension of each disease (e.g. only microarray data, only image data, etc.). The proposed methodology offers the aspiring prospect of repurposing existing data such as those residing in microarray expression, proteomic, and genomic data repositories, making it easy for researchers and physicians to ask new questions about existing data. Further study of the class discriminative effect of the imputation algorithms on unified datasets coming from other modalities and relating to other diseases as well as to multi-modal datasets retrieved though from the same patients (which is amenable to canonical correlation analysis-CCA [112, 113]), could solidify the method's capability and the potential performance improvement in predicting biomarkers.

APPENDICES

APPENDIX 1.   KEGGCONVERTER

1.1 MAIN.JAVA

```java
/*
 * Main.java
 */

package gr.eie.keggconvert;

/**
 *
 * @author kmouts
 */
import com.sun.org.apache.xerces.internal.parsers.DOMParser;
import gr.eie.utils.fileutils;
import gr.eie.utils.statistics;
import javax.xml.transform.dom.DOMSource;
import javax.xml.transform.Transformer;
import javax.xml.transform.TransformerFactory;
import javax.xml.transform.stream.StreamResult;
import org.w3c.dom.*;
import java.util.*;
import java.io.*;

public class Main {

    /** Creates a new instance of Main */
    public Main() {
    }

    public static boolean makeKinetics=false;
    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) {
        if (args.length==0 || args[0].equalsIgnoreCase("help")) {
            help();
        }

        for (String option:args){
            if (option.equalsIgnoreCase("makeKinetics")){
                makeKinetics=true;
                convert(false);
            }else
                if (option.equalsIgnoreCase("makeKineticsCD")){
                makeKinetics=true;
                convert(true);
                }else
                    if (option.equalsIgnoreCase("STATS")){
                makeKinetics=true;
                stats();
                    }else
                        if (option.equalsIgnoreCase("justConvert")){
                convert(false);
                        }else
                            if (option.equalsIgnoreCase("justConvertCD")){
                convert(true);
                            }else
                                if (option.equalsIgnoreCase("help")){
                help();
                                } else{
                System.out.println("No recognized option selected. Program termination.");
                System.exit(0);
                                }
        }

    }

    private static void help() {
        System.out.println("Usage: 'java -jar keggconvert.jar justConvert' to process kgml files from .\\in directory.");
        System.out.println("'justConvertCD' to process kgml files from .\\in directory to CellDesigner sbml.");
        System.out.println("'makeKinetics' to convert and introduce kinetics in the produced sbml files.");
        System.out.println("'makeKineticsCD' to convert and introduce kinetics in the produced CellDesigner sbml files.");
        System.out.println("'STATS' to produce reaction statistics for all sbml files in .\\out directory.");
        System.out.println("'help' to get these available options.");
        System.out.println("In case of being behind proxies you must use options before '-jar': '-Dhttp.proxyHost=xxx.xx.xxx.x -Dhttp.proxyPort=8080'" );
        System.exit(0);
    }

    private static void convert(boolean withCD) {
        fileutils.visitAllFiles(new File("in"));
        if (fileutils.in_files.isEmpty()) {
            System.out.println("Usage: ./in directory is empty. Please put there your kgml files for convertion");
            System.exit(1);
        }

        System.out.println("Input for convertion: " + fileutils.in_files.size() + " kegg files.");
        boolean sucess=fileutils.emptyDir(new File("out"));
        File outdir = new File("out");
        outdir.mkdir();

        System.out.println("Output directory \'out\' is empty for the new results: "+sucess);

        int counter=0;
        File file=null;
        for (File input_file:fileutils.in_files) {
            try {
                if (withCD){
                    file=gr.eie.kgml.converter2CD.converter_kgml(input_file);
                }else {
                    file = gr.eie.kgml.converter2SBML.converter_kgml(input_file);
                }
                System.out.println(++counter +" "+ input_file.toString() + " converted.");
                String dir=getDir(input_file);

                int numOfDirs=countOccurences(dir, "/"); //create dirs
                if (numOfDirs>0){
                    if (!(new File("out/"+dir)).exists()) {
                    (new File("out/"+dir)).mkdirs();
                    }
                }

                String newPath="out/"+dir+"sbml_"+input_file.getName();
                File newfile=new File(newPath);

                fileutils.copyFile(file,newfile);

                if (newfile.exists()) {
                    System.out.println(counter + " copied to output directory.");
                }

            } catch (Exception e) {
                System.out.println("Error at converting file "+input_file.toString()+ ", count:"+ counter);
                System.out.println(e.toString());
            }
        }
    }


// if the first argument is STATS, read the "out" dir, if it is not empty,
    // and make counting concerning num of reactions, reactants, products
    private static void stats() {
        System.out.println("Reaction statistics requested");
        fileutils.visitAllFiles(new File("out"));
        if (fileutils.in_files.isEmpty()) {
            System.out.println("./out directory is empty. Please put there your sbml files for reaction statistics");
            System.exit(1);
        }
        try {
            statistics.statReactions(fileutils.in_files) ;
        } catch (Exception e) {
            System.out.println("Error at statReactions");
            System.out.println(e.toString());
        }
        System.out.println("Statistics done. Processed:"+statistics.numReactions+ " reactions");
        statistics.saveToFileMatrices();
        System.out.println("Files revStatistics.txt and irrStatistics.txt created");
        System.exit(0);
    }
// taking the full path of the input file, to produce a String with only the intermediate dir path
// if available, else ""
     static String getDir(File file) {
         int nameLength=file.getName().length();
         String getPath=file.getPath();
         int pathLength=getPath.length();
         if (pathLength == 3+nameLength) { //there is no dir under "\in"
             return "";
         }else{
        String subDir= getPath.substring(3,pathLength-nameLength);
        return subDir.replaceAll("\\\\","/");

         }

    }
         public static int countOccurences(String string, String tosearchfor){
        int count = 0;
        for(int index = 0; (index = string.indexOf(tosearchfor, index)) != -1; count++, index += tosearchfor.length());
        return count;
    }

}
```

1.2 KINETICS.JAVA

```java
/*
 * kinetics.java
 *
 */

package gr.eie.keggconvert;

import gr.eie.sbmlEditor.km_KinInventory;
import gr.eie.sbmlEditor.km_KineticLaw;
import java.io.ByteArrayInputStream;
import java.io.UnsupportedEncodingException;
import java.util.ArrayList;
import javax.print.Doc;
import org.sbml.libsbml.ASTNode;
import org.sbml.libsbml.MathMLDocument;
import org.sbml.libsbml.libsbml;
import org.w3c.dom.Document;
import org.w3c.dom.Element;
import org.w3c.dom.NamedNodeMap;
import org.w3c.dom.Node;
import org.w3c.dom.NodeList;
import uk.ac.ebi.compneur.xml.ElementUtilities;
import uk.ac.ebi.compneur.xml.JAXPFacade;
import uk.ac.ebi.compneur.xml.ListErrorHandler;



public class kinetics {

  //  boolean reactionReversibleBoolean;
    static ArrayList<km_KineticLaw> arrayKinetics;


    /** Creates a new instance of kinetics */
    public kinetics() {
    }
//will create kineticLaw for each reaction. If it is 111 case will put Hyperbolic modifier
// else mass action, without removing the existing modifiers
    public static Document createKinetics(Document doc) {
              if (arrayKinetics==null || arrayKinetics.size()==0){ //read only once the file
           arrayKinetics=(new km_KinInventory("Kinetics.txt")).getKinetics();
        }
                  NodeList reactionList = doc.getElementsByTagName("reaction");
        String reversibleReaction=null; //if the reaction is reversible or not
        int reactants=0;

        for (int i=0; i<reactionList.getLength(); i++) { //for each reaction in file


            int numReactants=0;
            int numProducts=0;
            int numModifiers=0;
            String[] reactantsArray=null;
            String[] productsArray = null;
            String[] modifiersArray=null;

            Element reaction = (Element)reactionList.item(i);
            NamedNodeMap nnm = reaction.getAttributes();
            if (nnm != null) {
                Node   node = nnm.getNamedItem("reversible");
                if (node.equals("Null")) {reversibleReaction="true";} //default is reversible
                reversibleReaction = node.getNodeValue(); //true or false
            }
            Element kineticLaw = doc.createElement("kineticLaw");


            NodeList listOfReactants= reaction.getElementsByTagName("listOfReactants");
            if (listOfReactants.getLength()>0) {
                Element element = (Element)listOfReactants.item(0);
                NodeList ReactantSpeciesRef= element.getElementsByTagName("speciesReference");
                numReactants= ReactantSpeciesRef.getLength(); //update the num of reactants
                reactantsArray=new String[numReactants];
                for (int ii=0; ii<numReactants; ii++ ){
                Element reactant= (Element) ReactantSpeciesRef.item(ii);
                String reactantName=reactant.getAttribute("species");
                reactantsArray[ii]=reactantName;
                }
            }
            NodeList listOfProducts= reaction.getElementsByTagName("listOfProducts");
            if (listOfProducts.getLength()>0) {
                Element element = (Element)listOfProducts.item(0);
                NodeList ProductsSpeciesRef= element.getElementsByTagName("speciesReference");
                numProducts= ProductsSpeciesRef.getLength(); //update the num of products
                productsArray=new String[numProducts];
                for (int ii=0; ii<numProducts; ii++ ){
                Element product= (Element) ProductsSpeciesRef.item(ii);
                String productName=product.getAttribute("species");
                productsArray[ii]=productName;
                }
            }
            NodeList listOfModifiers= reaction.getElementsByTagName("listOfModifiers");
            if (listOfModifiers.getLength()>0) {
                Element element = (Element)listOfModifiers.item(0);
                NodeList modspeciesRef= element.getElementsByTagName("modifierSpeciesReference");
                numModifiers= modspeciesRef.getLength(); //update the num of modifiers
                 modifiersArray=new String[numModifiers];
                for (int ii=0; ii<numModifiers; ii++ ){
                Element modifier= (Element) modspeciesRef.item(ii);
                String modifierName=modifier.getAttribute("species");
                modifiersArray[ii]=modifierName;
                }
            }
            if (numReactants==1 && numProducts==1 && numModifiers==1){
                //add Hyperbolic Modifier;
                if (Boolean.parseBoolean(reversibleReaction)){
                    km_KineticLaw o =km_KinInventory.searchByTheName("Hypervolic_modifier");
                    fixKineticLawElement(kineticLaw,o,reactantsArray[0],productsArray[0],modifiersArray[0]);
                    reaction.appendChild(kineticLaw);

                }else {
                    km_KineticLaw o =km_KinInventory.searchByTheName("Hyperbolic modifier (irr)");
                    fixKineticLawElement(kineticLaw,o,reactantsArray[0],productsArray[0],modifiersArray[0]);
                    reaction.appendChild(kineticLaw);
                }
            }else {
                   //mass action;
                                if (Boolean.parseBoolean(reversibleReaction)){//"Mass_action"
                   String[] paramsArray={"k1","k2"};


fixKineticLawElementMA(kineticLaw,reversibleReaction,reactantsArray,productsArray,paramsArray);
                    reaction.appendChild(kineticLaw);

                }else {
                   String[] paramsArray={"k1"};

fixKineticLawElementMA(kineticLaw,reversibleReaction,reactantsArray,productsArray,paramsArray);
                    reaction.appendChild(kineticLaw);
                }
              //System.out.println("Mass action assumed. Existing Modifiers left intact.");
            }

        } //for all reactions of file


        return doc;
    }
    static
  {
    System.loadLibrary("sbmlj");
  }



    static String translateMathML(String xml)  //code form libSBML Java examples (translateMath.java)
    {
        /**
         * Prepend an XML header if not already present.
         */
        if (xml.charAt(0) == '<' && xml.charAt(1) != '?') {
            StringBuffer sb = new StringBuffer();

            sb.append("<?xml version='1.0' encoding='ascii'?>\n");
            sb.append(xml);

            xml = sb.toString();
        }

        MathMLDocument d = libsbml.readMathMLFromString(xml);
        ASTNode di=d.getMath();
        if (di != null) {
            return libsbml.formulaToString( di );
        } else return "";
    }
    static String translateInfix(String formula) //code form libSBML Java examples (translateMath.java)
    {
        MathMLDocument d = new MathMLDocument();

        d.setMath( libsbml.parseFormula(formula) );
        return libsbml.writeMathMLToString(d);
    }

       /**
     * Sets the mathML sub-tree from a String. Code from SBMLElementJDialog.java
     *
     * @param mathMLString
     */
    static protected void setMathMLContent(Element mathParent, String mathMLString) {

        if (mathMLString == null || mathMLString.length() == 0) {
            return;
        }
        // set all the mathMl Element
          final String MATHML_NAMESPACE = "http://www.w3.org/1998/Math/MathML";

        Element mathSubEl = ElementUtilities.appendChildElementIfNotExistentNS(
                mathParent, MATHML_NAMESPACE, "math");


        Document mathDoc =getNewDocument(mathMLString);

        Element mathEl = mathDoc.getDocumentElement();

        NodeList childs = mathEl.getChildNodes();
        for (int i = 0; i < childs.getLength(); i++) {
            Node node = childs.item(i);

        }

        Document document=mathParent.getOwnerDocument(); //kmouts
        Node mathNode = document.importNode((Node) mathEl, true);

        mathParent.replaceChild(mathNode, mathSubEl);
    }
       /**
     * Returns a new DOM document.
     *
     * @param documentString
     *           the new document as a String
     * @return the new DOM document.
     */
   static  protected Document getNewDocument(String documentString) {
        return getNewDocument(documentString, false);
    }

    /**
     * Returns a new DOM document.
     *
     * @param documentString
     *           the new document as a String
     * @param wantHTMLFormat
     * @return the new DOM document.
     */
    static protected Document getNewDocument(String documentString, boolean wantHTMLFormat) {

        StringBuffer documentHTLM = new StringBuffer();

        if (wantHTMLFormat) {
            char[] documentChars = documentString.toCharArray();


            for (int i = 0; i < documentChars.length; i++) {
                documentHTLM.append(documentChars[i]);
            }
        } else {
            documentHTLM.append(documentString);
        }
        final String MATHML_NAMESPACE = "http://www.w3.org/1998/Math/MathML";
        Object[] xmlns = {MATHML_NAMESPACE};
        String[] schemas = new String[1];
        schemas[0] = "http://www.ebi.ac.uk/compneur-srv/dopanet/MolecularPages/DopaNetMP.xsd";

        Document docu = null;
        try {
            docu = JAXPFacade.getInstance().create(new
ByteArrayInputStream(documentHTLM.toString().getBytes("US-ASCII")), null, new ListErrorHandler());
        } catch (UnsupportedEncodingException e) {
            e.printStackTrace();
            docu = JAXPFacade.getInstance().create(new
ByteArrayInputStream(documentHTLM.toString().getBytes()), xmlns, new ListErrorHandler());
```

```
        } catch (IllegalArgumentException e) {
            e.printStackTrace();
        } catch (RuntimeException e) {
            docu = JAXPFacade.getInstance().create(new ByteArrayInputStream(documentHTLM.toString().getBytes()), false);
        }

        return docu;
    }

    //given a kineticLaw element and a km_KineticLaw object for the case of 1-1 reaction for a given reactant, product and modifier
    // it populates the kineticLaw element with the appropriate <math> and <listOfParameters> elements
    //with parameter values "1"
    protected static void fixKineticLawElement(Element kineticLaw, km_KineticLaw o,String myreactant,String myproduct, String mymodifier) {
        String oFormula=o.getFormula();
        String newFormula=null;
        newFormula=oFormula.replaceAll("P","PP99PP"); //replace "P" as to not intervene with P chars is Product names
        newFormula=newFormula.replaceAll("S",myreactant);
        newFormula=newFormula.replaceAll("PP99PP",myproduct);
        String[] paramsArray;
        paramsArray = o.getParams().toArray(new String[o.getParamNum()]);
        newFormula=km_KineticLaw.substituteModifierString(o,newFormula,mymodifier);
        String mathEquation_xmlversion=translateInfix(newFormula); //get mathML text
        //              \<\?xml version\=\"1\.0\" encoding\=\"UTF-8\"\?\>[\r\n]  Remove the xml version tag
        mathEquation_xmlversion=mathEquation_xmlversion.replaceFirst(       "\\<\\?xml version\\=\\\"1\\.0\\\" encoding\\=\\\"UTF-8\\\"\\?\\>[\\r\\n]","");
        setMathMLContent(kineticLaw,mathEquation_xmlversion); //create math element under kinetic Law element
        Document doc=kineticLaw.getOwnerDocument();
        Element mylistOfParameters= doc.createElement("listOfParameters");
        for (int iii=0;iii<paramsArray.length; iii++ ){ //for each parameter
            Element myparam=doc.createElement("parameter");
            myparam.setAttribute("name",paramsArray[iii]); //set the name of the parameter
            myparam.setAttribute("value","1"); //set default value 1
            myparam.setAttribute("id",paramsArray[iii]); //create id for parameter also
            //  ((Element)mylistOfParameters).appendChild(document.createTextNode("\n"+"\t\t\t\t\t\t"));
            ((Element)mylistOfParameters).appendChild(myparam);
        }// end for
        kineticLaw.appendChild(mylistOfParameters);
    }

    protected static void fixKineticLawElementMA(Element kineticLaw,String reversibleReaction, String[] reactantsArray, String[] productsArray, String[] paramsArray) {
        String newFormula="k1";
        for (int k=0;k<reactantsArray.length;k++){
            newFormula=newFormula + "*" +reactantsArray[k];
        }
        if (Boolean.parseBoolean(reversibleReaction)){
            newFormula=newFormula + "-k2";
            for (int k=0;k<productsArray.length;k++){
                newFormula=newFormula + "*" +productsArray[k];
            }
        }
        String mathEquation_xmlversion=translateInfix(newFormula); //get mathML text
        //              \<\?xml version\=\"1\.0\" encoding\=\"UTF-8\"\?\>[\r\n]  Remove the xml version tag
        mathEquation_xmlversion=mathEquation_xmlversion.replaceFirst(       "\\<\\?xml version\\=\\\"1\\.0\\\" encoding\\=\\\"UTF-8\\\"\\?\\>[\\r\\n]","");
        setMathMLContent(kineticLaw,mathEquation_xmlversion); //create math element under kinetic Law element
        Document doc=kineticLaw.getOwnerDocument();
        Element mylistOfParameters= doc.createElement("listOfParameters");
        for (int iii=0;iii<paramsArray.length; iii++ ){ //for each parameter
            Element myparam=doc.createElement("parameter");
            myparam.setAttribute("name",paramsArray[iii]); //set the name of the parameter
            myparam.setAttribute("value","1"); //set default value 1
            myparam.setAttribute("id",paramsArray[iii]); //create id for parameter also
            //  ((Element)mylistOfParameters).appendChild(document.createTextNode("\n"+"\t\t\t\t\t\t"));
            ((Element)mylistOfParameters).appendChild(myparam);
        }// end for
        kineticLaw.appendChild(mylistOfParameters);
    }

}
```

1.3 KM_KININVENTORY.JAVA

```
/*
 * km_KinInventory.java
 */

package gr.eie.sbmlEditor;

import java.io.File;
import java.io.FileNotFoundException;
import java.util.ArrayList;
import java.util.Iterator;
import java.util.Scanner;
import java.util.regex.Matcher;
import java.util.regex.Pattern;

/**
 *
 * @author kmouts
 */
public class km_KinInventory {

    private static ArrayList<km_KineticLaw> kinetics=new ArrayList<km_KineticLaw>();

    /**
     * Creates a new instance of km_KinInventory
     * The 3 lines for each kinetics are:
     * 1:reversible,name, paramNum,params,modifierNum,modifierType,modifier
     * 2:formula
     * 3:pattern
     * The source is according to http://java.sun.com/developer/JDCTechTips/2004/tt1201.html
     * @param fileName The file name which has the kinetic definitions, and rests in the top directory, one dir up the source dir
     */
    public km_KinInventory(String fileName) {
        try {
            Scanner scanner =
                    new Scanner(new File(fileName));
            scanner.useDelimiter
                    (System.getProperty("line.separator"));
            while (scanner.hasNext()) { //3 more lines
                String par1=scanner.next();
                if (scanner.hasNext()){ String par2 = scanner.next();
                if (scanner.hasNext()){ String par3 = scanner.next();
                km_KineticLaw temp = new km_KineticLaw(par1,par2,par3);
                getKinetics().add(temp);
                }
                }
            }
            scanner.close();
        } catch (FileNotFoundException e) {
            e.printStackTrace();
        }
    }
    public static String[] convertToStringArrayKinetics(boolean reversibles) {
        //    String[] kiTemp = null;
        ArrayList<String> temp=new ArrayList<String>();
        Iterator it = getKinetics().iterator();
        while (it.hasNext()) {
            km_KineticLaw o = (km_KineticLaw) it.next();
            if (! o.isReversible() == reversibles) continue ;
            temp.add(o.toString());
        }
        String kiTemp[] = new String[temp.size()];
        kiTemp= temp.toArray(kiTemp);
        return kiTemp;
    }
    /**
     *
     * A function to match the ascii formulae with the inventory of rate laws.
     * It substitutes the symbol "&" from the kinetics.txt with the pattern group (par1|par2|..)
     * @param reversible If the reaction to be found is reversible or not
     * @param mathString The ascii math string of the reaction to compare with the inventory of kinetics
     * @param paramlabelsArray The names of the parameters
     * @return It returns a ArrayList with the first element the rate law found, and next the names of the modifiers if any exist. If there is no match it returns null.
     */
    protected static ArrayList<Object> matchKinetic(boolean reversible, String mathString,String[] paramlabelsArray) {
        Iterator it = getKinetics().iterator();
        while (it.hasNext()) {
            km_KineticLaw o = (km_KineticLaw) it.next();
            if (! reversible==o.isReversible()) continue ;
            int count_params=paramlabelsArray.length;
            if (count_params==o.getParamNum()){
                String pattern=o.getPattern();

                if (pattern.contains("&")){ // we use "&" to denote in the file the parms "or": (Kms|Kmp|..)
                    pattern=replaceModChar(pattern,paramlabelsArray);
                }
                if ( (!(reversible)) && count_params==1 && (! (paramlabelsArray[0].equalsIgnoreCase(mathString)))){
                    //    int findMassAction = kinetics.lastIndexOf("Mass_action");
                    ArrayList<Object> kinFoundResults = new ArrayList<Object>();
                    kinFoundResults.add(getKinetics().get(18));
                    return kinFoundResults;
                }
                boolean match;
                match=Pattern.matches(pattern,mathString);
                if (match){
                    ArrayList<Object> kinFoundResults = new ArrayList<Object>(); //the first item is km_KineticLaw found, and the rest the modifiers
                    kinFoundResults.add(o);
                    if (o.getModifierNum()>0){
                        Pattern kinPattern=Pattern.compile(pattern);
                        Matcher kinMatcher=kinPattern.matcher(mathString);
                        kinMatcher.find();
                        String modifier=kinMatcher.group(4); //Always we set the 4th pattern group as the Modifier
                        kinFoundResults.add(modifier);
                    }
                    return kinFoundResults;
                }
            }
        }
        return null;
    }
    public static  km_KineticLaw searchByTheName(String name){
        // Km_KineticLaw o;
        for (km_KineticLaw o : kinetics){
            String temp = o.getName();
            if (temp.equals(name)){
                return o;
            }
        }
        return null;
    }
    /**
     * It replaces the '&' char in a String with the given parameters
     * in order to make a substitute in the search pattern. For instance: & -> (Kms|Kmp|Vf|Vr|Ka)
     * @param toChange The given (pattern) String
     * @param paramlabelsArray An Array of String with the parameters chars
     * @return It returns the modified String
     */
    protected static String replaceModChar(String toChange,String[] paramlabelsArray){
        String paramsString="(" + paramlabelsArray[0];
        for (int i=1; i<paramlabelsArray.length ; i++){
            paramsString=paramsString + "|" + paramlabelsArray[i];
        }
        paramsString=paramsString + ")";
        String replaced = toChange.replaceAll("&",paramsString);
        return replaced;
    }

    public static ArrayList<km_KineticLaw> getKinetics() {
        return kinetics;
    }
}
```

1.4 KM_KINETICLAW.JAVA

```
/*
 * km_KineticLaw.java
 */

package gr.eie.sbmlEditor;

/**
 *
 * @author kmouts
 */
import java.util.ArrayList;
import java.util.Scanner;
import java.util.regex.Matcher;
import java.util.regex.Pattern;

public class km_KineticLaw {

    private String name;
    private String pattern;
    private boolean reversible;
    private int paramNum; //the number of parameters
    private ArrayList<String> params=new ArrayList<String>();
    private String formula;
    private int modifierNum;
    private int modifierType; //0=modif, 1=activator, 2=inhibitor
    private ArrayList<String> modifier=new ArrayList<String>();
    /**
     * Creates a new instance of km_KineticLaw
     */
    public km_KineticLaw(String par1, String par2, String par3) {
        parseLine1(par1);
        parseLine2(par2);
        parseLine3(par3);
    }


      private  void parseLine1(String line) {
            Scanner lineScanner = new Scanner(line);
            lineScanner.useDelimiter("\\s*,\\s*");

            this.reversible = lineScanner.nextBoolean();
            this.name=lineScanner.next();
            this.paramNum=lineScanner.nextInt();
            for (int i=0;i<getParamNum();i++){
                String par=lineScanner.next();
                this.getParams().add(par);
            }
            modifierNum=lineScanner.nextInt();
            for (int i=0;i<getModifierNum();i++){
                this.modifierType=lineScanner.nextInt();
                this.getModifier().add(lineScanner.next());
            }
        }
    private    void parseLine2(String line) {
            Scanner lineScanner = new Scanner(line);
            lineScanner.useDelimiter("\\s*,\\s*");
            this.formula = lineScanner.next();
        }
    private    void parseLine3(String line) {
            Scanner lineScanner = new Scanner(line);
        lineScanner.useDelimiter("\\s*,\\s*");
        if (! line.equals("")) {
        this.pattern = lineScanner.next();
        }
    }
public String toString() {
    return getName();
}


/**
 *
 * This routine finds the positions of a modifier string inside a equation string
 * @param o
 * @return An int[] array with the group index found starting counting at 0 or null otherwise
 */
   protected static int[] findModifierGroupPosition (km_KineticLaw o) {
       ArrayList<Integer> modNums=null;  //The positions of modifiers
       if (o.modifierNum != 0) {
           modNums=new ArrayList<Integer>();
           String regex="\\w+" ;
           Pattern p= Pattern.compile(regex);
           Matcher m=p.matcher(o.formula);

           boolean result = m.find(  ) ;
           int j=0; //group position counter
           while ( result )   {
               String stringFound=m.group();
               if (stringFound.equals(o.getModifier().get(0))){
                   modNums.add(j);
               }
               j++ ;
               result = m.find(  ) ;
           }
       }
       //convert ArrayList<Integer> to int[]
       int functionResult[] = new int[modNums.size()];
       for (int i=0; i<modNums.size();i++){
           functionResult[i]=modNums.get(i);
       }
       return functionResult;
   }


public int getModifierNum() {
    return modifierNum;
}

public int getModifierType() {
    return modifierType;
}

public ArrayList<String> getModifier() {
    return modifier;
}

public String getPattern() {
    return pattern;
}

public int getParamNum() {
    return paramNum;
}

public String getFormula() {
    return formula;
}

public String getName() {
    return name;
}

```

```
    public boolean isReversible() {
        return reversible;
    }

    public ArrayList<String> getParams() {
        return params;
    }


}
```

APPENDIX 2. GOREVENGE

2.1 WEB2PY CONTROLLER FUNCTION GR

```
def gr():
    PROF = False #my profiling
    species = db(db.mydates.name != 'go').select(db.mydates.name)
    myspecies=[]
    [myspecies.append(i.name) for i in species]
    res=''
    is_email=eresult=False
    email_len=0

    form=SQLFORM.factory(
       Field('organism', requires=IS_IN_SET(myspecies,zero=T('choose one'),
           error_message=T('select one of the available organisms')) ),
       Field('aspect',requires=IS_IN_SET(['MF','BP','CC'],zero=T('choose one'),
           error_message=T('select one of the aspects'))),
       Field('distance',requires=IS_IN_SET(['graph','resnik'],zero=T('choose one'),
           error_message=T('select one of the distance methods')) ,comment=T("resnik is A LOT heavier to compute!") ),
       Field('algorithm',requires=IS_IN_SET(['Bubble GO','Bubble Genes'],zero=T('choose one'),
           error_message=T('select one of the algorithms'))),
       Field('relaxation',requires=IS_IN_SET(['-1','0','0.15','0.30','0.45','0.60','0.75','0.90'],
                              zero=T('choose one'),error_message=T('select one of the available levels'))),
       Field('input','text',requires=IS_NOT_EMPTY(error_message=T('paste GO terms for BubbleGO or genes for BubbleGenes')),
                        comment=T("Enter genes for 'Bubble Genes', or GO terms (without the 'GO:' prefix) for 'Bubble GO', separated with white spaces, comma, or semicolon (allthough not in combination!)")),
       Field('email', comment=T("Enter a valid email address if you'd like to receive all the results along with observed GO sets")),

    )

    if form.accepts(request.vars, session, formname='form', keepvalues=True):
        organism = form.vars.organism
        aspect = form.vars.aspect
        algorithm = form.vars.algorithm
        distance = form.vars.distance
        relaxation = float(form.vars.relaxation)
        input = form.vars.input
        email = form.vars.email
        if ',' in input: myinput = input.split(',')
        elif ';' in input: myinput = input.split(';')
        else: myinput = input.split()

        if PROF:
            print 'myinput',myinput
            t0=time.time()

        conn_str=os.path.join(request.folder,'databases','storage.sqlite')
        go_genes, gene_gos = reveng.make_go_genes_organism(conn_str, organism, aspect, fixed='')

        if distance == 'resnik':
#             data=reveng.resnik_reader_larry(conn_str, organism=organism, aspect = aspect, input = input)
#             print 'len(data)=',data.shape
            pass
        else:
            pickle_location = os.sep.join(conn_str.split(os.sep)[:-1]) + os.sep + 't2p_' + aspect + '.bin'
            t2p = cPickle.load(open(pickle_location)) # {'0019002': ['0001883', '0032561']}
            if relaxation<0: relaxation=0
            relax = int(relaxation * 10)
            if PROF:print 'relax=',relax

        if algorithm == 'Bubble GO':
            if distance == 'resnik':
#                 print 'Bubble GO'
```

```
                myinput.sort()
                data,mylabelsdict,f=reveng.resnik_reader_larry(conn_str, organism=organism, aspect = aspect, input = myinput)
                if PROF:print 'len(data)=',data.shape
                res = reveng.BubbleGO(myinput, go_genes, resnik=data, relax=relaxation)
#                 print 'Bubble GO finished'
            elif distance == 'graph':
                res = reveng.BubbleGO(myinput, go_genes, relax=relax, t2p=t2p)

        elif algorithm == 'Bubble Genes':
            if distance == 'resnik':
#                 print 'Bubble Genes'
                if PROF:
                    t1=time.time()
                    print 'in default BubbleGenes',time.time() - t0, "seconds"
#
#
                sem_gos = set()
                for gene in myinput:
                    sem_gos.update(gene_gos[gene])
                myGOSinput = list(sem_gos)
                myGOSinput.sort()
                data,mylabelsdict,f=reveng.resnik_reader_larry(conn_str, organism=organism, aspect = aspect, input = myGOSinput)
                if PROF:
                    t3=time.time()
                    print 'in default BubbleGenes after resnik_reader_larry',time.time() - t1, "seconds"
                res = reveng.BubbleGenes(myinput, go_genes, resnik=data, relax=relaxation)
                if PROF:print 'in default len(data)=',data.shape

#                 print 'Bubble Genes finished'
            elif distance == 'graph':
                res = reveng.BubbleGenes(myinput, go_genes, relax=relax, t2p=t2p)

        if distance == 'resnik':
            # load in larry 'data' the relaxed gos too
            # make a data 2 larry for the new relaxed gos
            relaxed_gos=set()
            [relaxed_gos.update(myres['gos']) for myres in res]
            extra_gos_set = relaxed_gos - set(data.label[0])
            extra_gos = list(extra_gos_set)
            extra_gos.sort()

            lgos = len(data.label[1])
            data2 = numpy.zeros( (len(extra_gos),lgos ), dtype=float)

            l_range =  range(lgos)
            for i,go in enumerate(extra_gos):
                f.seek(mylabelsdict[go]['p'])
                d = f.readline().split()
                for j in l_range:
                    data2[i][j]=d[1+j]  #float(d[1+j])


            label2 = [extra_gos, data.label[1]]
            myresnik2 = la.larry(data2, label2)

            if PROF:
                t5d=time.time()
                print 'in default BubbleGenes d',time.time() - t3, "seconds"

            data = data.merge(myresnik2)
    #         data,mylabelsdict,f=reveng.resnik_reader_larry(conn_str, organism=organism, aspect = aspect, input = relaxed_gos)
            if PROF:
                print 'in default after update, data has now=',data.shape #, globals()['data'].shape
                t5=time.time()
                print 'in default BubbleGenes',time.time() - t5d, "seconds"


        for myres in res:
#             print 'myres[gos]=',myres['gos']
```

```
            myres['count_orig'] = len(gene_gos[myres['gene']])
            if len(myres['gos'])>1:
                if distance == 'resnik':
                    clust = reveng.hcluster(list(myres['gos']),data)
                    myres['p1'] = reveng.hclust_cutter(clust, distance=0.3, res=set())
                    myres['p2'] = reveng.hclust_cutter(clust, distance=0.6, res=set())
                    myres['p3'] = reveng.hclust_cutter(clust, distance=0.9, res=set())
#                     print 'prune1', len(myres['gos']),'-->', len(myres['p1'])
                elif distance == 'graph':
                    mres = reveng.myprune_graph(list(myres['gos']), t2p, prune=6 )
                    myres['p1'] = mres[2]
                    myres['p2'] = mres[4]
                    myres['p3'] = mres[6]
            else:
                myres['p1'] = myres['p2']= myres['p3']= myres['gos']

        if PROF:t6=time.time()
        #print 'in default BubbleGenes after hcluster',time.time() - t5, "seconds"

        email_len=len(email)
        if email_len:
            email_test= IS_EMAIL()(email)
            if not email_test[1]: #an den einai None to verification
                is_email=True
                mail=Mail()
                mail.settings.server='smtp.gmail.com:587'
                mail.settings.sender='xxx@gmail.com'
                mail.settings.login= None or 'xxx:xxxxx'

                htmlmessage=""" <html><table id="results">
<TR><TH>gene</TH><TH>count</TH><TH>p1</TH><TH>p2</TH><TH>p3</TH><TH>count_orig</TH></TH>""" + \
str(TBODY(*[TR(*[rows['gene'],rows['count'],len(rows['p1']),len(rows['p2']),len(rows['p3']),rows['count_orig'] ])
                                         for rows in res[:1000]])) + \
                        """</table></html> """

                bodymessage='organism= '+organism+'\naspect= '+aspect+'\nalgorithm= '+algorithm+'\ndistance= '+distance+ \
                        '\nrelaxation= '+str(relaxation)+'\ninput= '+input+'\n\n' + \
                        'Thank you for using GOrevenge!'

                message="gene\tcount\tp1\tp2\tp3\tcount_orig\tcount_GOterms\n"
                message=message + "\n".join([ '\t'.join([rows['gene'],str(rows['count']),str(len(rows['p1'])),
                                                         str(len(rows['p2'])),str(len(rows['p3'])),str(rows['count_orig']),
                                                         ','.join((rows['gos']))
                                                        ]) for rows in res[:1000]])


                f = StringIO(message) #file-like object
                attachment = (Mail.Attachment(f,filename='GOrevenge_results.txt'))
                eresult = mail.send(to=email,subject='GOrevenge results',message=bodymessage, attachments = (attachment ) )
            else:
                is_email=False

    return dict(form=form,results=res,is_email=is_email,eresult=eresult, email_len=email_len, allgos=URL('static','gos/GOS.zip'))

```

2.2 Web2py XMLRPC controller function goRevenge

```

@service.xmlrpc
def goRevenge(organism='chicken',aspect='CC',distance='graph', algorithm='Bubble GO',relaxation=.1,input='0033178'):
    res=''
    if 'GO:' in input: input.replace('GO:','')

    if ',' in input: myinput = input.split(',')
    elif ';' in input: myinput = input.split(';')
    else: myinput = input.split()

    print 'myinput',myinput

    conn_str=os.path.join(request.folder,'databases','storage.sqlite')
    go_genes, gene_gos = reveng.make_go_genes_organism(conn_str, organism, aspect, fixed='')

    if distance == 'resnik':
        data=reveng.resnik_reader_larry(conn_str, organism=organism, aspect = aspect)
        print 'len(data)=',data.shape
    else:
        pickle_location = os.sep.join(conn_str.split(os.sep)[:-1]) + os.sep + 't2p_' + aspect + '.bin'
        t2p = cPickle.load(open(pickle_location)) # {'0019002': ['0001883', '0032561']}
        if relaxation<0: relaxation=0
        relax = int(relaxation * 10)
        print 'relax=',relax

    if algorithm == 'Bubble GO':
        if distance == 'resnik':
#                 print 'Bubble GO'
            res = reveng.BubbleGO(myinput, go_genes, resnik=data, relax=relaxation)
#                 print 'Bubble GO finished'
        elif distance == 'graph':
            res = reveng.BubbleGO(myinput, go_genes, relax=relax, t2p=t2p)

    elif algorithm == 'Bubble Genes':
        if distance == 'resnik':
#                 print 'Bubble Genes'
            res = reveng.BubbleGenes(myinput, go_genes, resnik=data, relax=relaxation)
#                 print 'Bubble Genes finished'
        elif distance == 'graph':
            res = reveng.BubbleGenes(myinput, go_genes, relax=relax, t2p=t2p)

    for myres in res:
#             print 'myres[gos]=',myres['gos']
        myres['count_orig'] = len(gene_gos[myres['gene']])
        if len(myres['gos'])>1:
            if distance == 'resnik':
                clust = reveng.hcluster(list(myres['gos']),data)
                myres['p1'] = reveng.hclust_cutter(clust, distance=0.3, res=set())
                myres['p2'] = reveng.hclust_cutter(clust, distance=0.6, res=set())
                myres['p3'] = reveng.hclust_cutter(clust, distance=0.9, res=set())
#                     print 'prune1', len(myres['gos']),'-->', len(myres['p1'])
            elif distance == 'graph':
                mres = reveng.myprune_graph(list(myres['gos']), t2p, prune=6 )
                myres['p1'] = mres[2]
                myres['p2'] = mres[4]
                myres['p3'] = mres[6]
        else:
            myres['p1'] = myres['p2']= myres['p3']= myres['gos']

    for dic in res: #cannot marshal set objects
        dic['gos']=list(dic['gos'])
        dic['p1']=list(dic['p1'])
        dic['p2']=list(dic['p2'])
        dic['p3']=list(dic['p3'])
    #print 'results=',res[:10]
    return res[:1000]

```

2.3 Library functions myrelax and myrelax_graph

```
def myrelax2(sem_gos, resnik, relax=-1 ): #line normalization and no_sort each time + larry update with relaxed gos...
    """
    relax: radius of go bubble
    relax = -1: no relax
    """
    relaxed_gos = set()
    for go in sem_gos:
        if go not in resnik.label[0]:
            print go, ' rejected.'
            continue
        if relax < 0:
            relaxed_gos.add(go)
        else:
            alarry = normalise_resnik_line(resnik.lix[[go]],go)
            relaxed_gos.update( set(alarry[alarry.x >= 1-relax].label[0]) )

    return relaxed_gos

def myrelax_graph(sem_gos, t2p, p2t, relax=0 ): #
    """
    relax: radius of go bubble, iterations of adding parents & childs
    relax = 0: no relax
    """
    for i,go in enumerate(sem_gos):
        if go not in t2p:
            print go, ' rejected.'
            del sem_gos[i]
    relaxed_gos = sem_gos[:]
    new_relaxed_gos = [] + relaxed_gos

    for i in range(relax): # for each circle take the childs and parents of each term
#         print i
        for go in relaxed_gos:
#             print i,go
            [new_relaxed_gos.append(go1) for go1 in t2p[go] if go1 not in new_relaxed_gos]
            [new_relaxed_gos.append(go2) for go2 in p2t[go] if go2 not in new_relaxed_gos]
        relaxed_gos=new_relaxed_gos[:]
    return relaxed_gos
```

2.4 LIBRARY FUNCTION HCLUSTER

```
def hcluster(gos,la_resnik):
    resnikeys=set(la_resnik.label[1])
    clear_gos=[]
    [clear_gos.append(go) for go in gos if go in resnikeys]
    rejected = [go for go in gos if go not in clear_gos]
    if len(rejected): print 'in hcluster, rejected:',rejected,'\
                                          len_clear_gos:',len(clear_gos)
    mylen = len(clear_gos)
    if mylen==1:
        return bicluster(la_resnik.lix[[clear_gos[0]],
                        [clear_gos[0]]],label=clear_gos[0],id=1)
    mymax = float(max([la_resnik.lix[[go],[go]] for go in clear_gos]))

    mymin = float(min([la_resnik.lix[[go],[go2]] for go in clear_gos \
                                                                                          for go2 in clear_gos]))
    myrange = mymax - mymin + 0.000000001

    full_normalised_resnik=defaultdict(dict)


    for go in clear_gos:
        for go2 in clear_gos: #resnik[go]:
            full_normalised_resnik[go][go2] = \
                                         (la_resnik.lix[[go],[go2]] - mymin) / myrange


    currentclustid=-1

    #Clusters are initially just the gos
    clust=[bicluster(la_resnik.lix[[clear_gos[i]],[clear_gos[i]]],
                        label=clear_gos[i],id=i) for i in range(len(clear_gos))]

    while len(clust)>1:
        lowestpair=(0,1)
        closest = 1 - full_normalised_resnik[clust[0].label][clust[1].label]

        #loop through every pair looking for the smaller distance
        for i in range(len(clust)):
            for j in range(i+1,len(clust)):
                d = 1 - full_normalised_resnik[clust[i].label][clust[j].label]

                if d < closest:
                    closest=d
                    lowestpair=(i,j)
        # keep the value of the max resnik of the two clusters
        mergeid=lowestpair[0] if clust[lowestpair[0]].val > \
                                        clust[lowestpair[1]].val else lowestpair[1]

        # create the new cluster with the label of the bigger resnik value
        newcluster=bicluster(clust[mergeid].val,left=clust[lowestpair[0]],
                                                        right=clust[lowestpair[1]],
                             label=clust[mergeid].label,
                                                         distance=closest,id=currentclustid)

        # cluster ids that weren't in the original set are negative
        currentclustid=-1
        del clust[lowestpair[1]]
        del clust[lowestpair[0]]
        clust.append(newcluster)
    return clust[0]
```

2.5 LIBRARY FUNCTION MYPRUNE_GRAPH

```
def myprune_graph(sem_gos, t2p, prune=0 ):
    """
    returns: res, a list with the remaining sem_gos after each prunning step
    """
    mygoset=set(sem_gos)
    parents=set()
    res=[sem_gos]

    if prune:
        for go in sem_gos:
            try: #if goterm is root continue
                parents.update(t2p[go])
            except KeyError: continue
        res.append(list(mygoset.difference(set(parents)))) # remove gos which were found to be parents
        prune -= 1

        while prune:
            newparents = parents.copy()
            for go in parents:
                try: #if goterm is root continue
                    newparents.update(t2p[go])
                except KeyError: continue
            parents = newparents.copy()
            res.append(list(mygoset.difference(set(parents)))) # remove gos which were found to be parents
            prune -= 1
        return res
```

2.5 TESTING FUNCTIONS

```
def test_myrelax_graph():
    t2p=term2parent #cPickle.load(open(os.path.join(gorevdir,'tests','t2p_MF.bin')))
    # {'0019002': ['0001883', '0032561']}
    p2t = resnik_module.fparent2term(t2p) # {'0019001': set(['0032561', '0032560'])}
    sem_gos = ['0046872']
    res = reveng.myrelax_graph(sem_gos, t2p, p2t, relax=0 ) # no adding
    assert res == ['0046872']
    res = reveng.myrelax_graph(sem_gos, t2p, p2t, relax=1 ) #
    assert len(res) == 3
    assert '0046872' in res
    res = reveng.myrelax_graph(sem_gos, t2p, p2t, relax=2 ) # get all the descendants
    assert len(res) == 4
    assert '0043167' in res
    sem_gos = ['0000497','0043565']
    res = reveng.myrelax_graph(sem_gos, t2p, p2t, relax=2 ) # get all the descendants
    assert len(res) == 7
    assert set(res) == set(['0000497', '0043565', '0000496', '0003677', '0000182',
                                          '0003676', '0043566'])
    res = reveng.myrelax_graph(sem_gos, t2p, p2t, relax=4 ) # get all the descendants
    assert len(res) == 20

def test_hclust():
    gos = ['0005525','0031406','0017076','0000287','0030145','0016301']
    # PEPCK Gene product
resnik= {'0005525': {'0005525': 6.0051900000000000000001, '0030145': 0.0077300000000000000000002, '0031406': 0.0077300000000000000000002, '0000287': 0.0077300000000000000000002, '0016301': 0.0, '0017076': 4.0002799009999900999996}, '0030145': {'0005525': 0.0077300000000000000000002, '0030145': 10.003800000000000000001, '0031406': 0.0077300000000000000000002, '0000287': 3.0029199009999900999998, '0016301': 0.0, '0017076': 0.0077300000000000000000002}, '0031406': {'0005525': 0.0077300000000000000000002, '0030145': 0.0077300000000000000000002, '0031406': 8.0058699009999900999997, '0000287': 0.0077300000000000000000002, '0016301': 0.0, '0017076': 0.0077300000000000000000002}, '0000287': {'0005525': 0.0077300000000000000000002, '0030145': 3.0029199009999900999998, '0031406': 0.0077300000000000000000002, '0000287': 8.141, '0016301': 0.0, '0017076': 0.0077300000000000000000002}, '0016301': {'0005525': 0.0, '0030145': 0.0, '0031406': 0.0, '0000287': 0.0, '0016301': 5.0007000000000000000000003, '0017076': 0.0}, '0017076': {'0005525': 4.0002799009999900999996, '0030145': 0.0077300000000000000000002, '0031406': 0.0077300000000000000000002, '0000287': 0.0077300000000000000000002, '0016301': 0.0, '0017076': 4.0002799009999900999996}}
clust=reveng.hcluster(gos,la_resnik)
    assert clust.label=='0030145'
    assert 1.0 - clust.distance < 0.001
#.printclust(clust, labels=True)
    print 'test_hclust: OK'

def test_myprune_graph():
    t2p=term2parent #cPickle.load(open(os.path.join(gorevdir,'tests','t2p_MF.bin')))
    # {'0019002': ['0001883', '0032561']}
    sem_gos = ['0003677','0043565'] # parent-child case
    res = reveng.myprune_graph(sem_gos, t2p, prune=0 ) # no prunning should occur
    assert res[0] == sem_gos
    res = reveng.myprune_graph(sem_gos, t2p, prune=1 ) # only child should remain
    assert res[1] == ['0043565']
    res = reveng.myprune_graph(sem_gos, t2p, prune=3 )
    # should be the same even with more iterations
    assert res[3] == ['0043565']
    sem_gos = ['0003676','0043565'] # grandparent case
    res = reveng.myprune_graph(sem_gos, t2p, prune=1 ) # nothing should change
    assert set(res[1]) == set(sem_gos)
    res = reveng.myprune_graph(sem_gos, t2p, prune=2 )
    # should be only the grand child
    assert res[2] == ['0043565']
    sem_gos = ['0005488','0000182','0046872']
    # 4 generations apart, and 3 gen apart from the 0005488
    res = reveng.myprune_graph(sem_gos, t2p, prune=3 ) #
    assert set(res[3]) == set(['0000182','0046872'])
    res = reveng.myprune_graph(sem_gos, t2p, prune=4 ) #
    assert set(res[4]) == set(['0000182','0046872'])

def test_hclust_with_real_data():
    gos=['0032142','0032137','0042803','0043531','0032181',
     '0003677','0030983','0019899','0008022','0000400','0016887','0003697','0019237',
        '0000287','0032143','0019901','0032357','0032405','0003684','0005524']
    #    resnik = test_resnik_reader_larry() #test_resnik_reader()
    resnik = test_resnik_reader_larry_with_input() #test_resnik_reader()

    clust=reveng.hcluster(gos[:10],resnik)
    res = reveng.hclust_cutter(clust, distance=0.9, res=set())
    assert len(gos) > len(res)
    clust2=reveng.hcluster(gos[:3],resnik)
    res2 = reveng.hclust_cutter(clust2, distance=0.9, res=set())
    assert len(gos[:3]) > len(res2)
    clust3=reveng.hcluster(['0046872','0000827'],resnik)
    # 0000827 not in resnik, so remove it
    res3 = reveng.hclust_cutter(clust3, distance=0.5, res=set())
    assert len(res3) > 0
    print 'test_hclust_with_real_data: OK'

def test_BubbleGO():
    sem_gos = ['0000049','0046872']
    res = reveng.BubbleGO(sem_gos, go_genes, resnik=la_resnik, relax=-1) # no relax
    assert res[0]['gene'] == 'CARS'
    assert res[0]['gos'] == set(['0000049','0046872'])
    assert len(res) == 7
    assert len(res[1]['gos']) == 1
    assert res[1]['count'] == 1

    res = reveng.BubbleGO(sem_gos, go_genes, resnik=la_resnik, relax=0) # resnik gets all the descendants of each term
    assert len(res) == 11
    gos=set()
    for g in res:
        gos.update(g['gos'])
    assert '0000287' in gos

    res = reveng.BubbleGO(sem_gos, go_genes, resnik=la_resnik, relax=0.5) #
    #spans to: 0000287-0046872-0043169-0043167 & 0000049-0003723
    assert len(res) == 20
    gos=set()
    for g in res:
        gos.update(g['gos'])
    assert '0003723' in gos
    assert '0043167' in gos


def test_BubbleGO_graph():
    t2p=term2parent #cPickle.load(open(os.path.join(gorevdir,'tests','t2p_MF.bin')))
    # {'0019002': ['0001883', '0032561']}
    sem_gos = ['0000049','0046872']
    res = reveng.BubbleGO(sem_gos, go_genes, t2p=t2p, relax=0 ) # no adding
    assert res[0]['gene'] == 'CARS'
    assert res[0]['gos'] == set(['0000049','0046872'])
    assert len(res) == 7
    assert len(res[1]['gos']) == 1
    assert res[1]['count'] == 1
    print 'test_BubbleGO_graph: OK'

def test_BubbleGenes():
    sem_genes = ['YARS','FARSA','CARS','ALB']

    res = reveng.BubbleGenes(sem_genes, go_genes, resnik=la_resnik, relax=-1)
    # no relax
    assert res[0]['gene'] == 'CARS'
    assert res[0]['gos'] == set(['0000049','0046872'])
    assert len(res) == 8
    assert len(res[1]['gos']) == 1
    assert res[1]['count'] == 1

    res = reveng.BubbleGenes(sem_genes, go_genes, resnik=la_resnik, relax=0)
    # resnik gets all the descendants of each term
    assert len(res) == 16
    gos=set()
    for g in res:
        gos.update(g['gos'])
    assert '0000287' in gos
    assert '0000062' in gos

def test_CreateGOAdb():
    specie = 'chicken'
    adresse = 'ftp://ftp.ebi.ac.uk/pub/databases/GO/goa/' + specie.upper() \
         +'/gene_association.goa_' + specie + '.gz'
    organism = specie
    conn_str=os.path.join(gorevdir,'tests','testdb')
    resnik.CreateGOAdb(adresse, organism, conn_str=conn_str)
    conn = sqlite3.connect(conn_str)
    curs = conn.cursor()
    curs.execute('SELECT * FROM annotation_chicken')
```

```
assert len(curs.fetchone()) == 17
assert len(curs.fetchall()) > 80000
```

APPENDIX 3. HETEROGENOUS DATA FUSION

3.1 MYVARIMPORTANCE.R

```
myVarImportance <- function(filter=TRUE, myname='ImageFeatImp',
                            myrepetitions=50, makeplots=F,
                            microdata_file="myDATA/DSm_ready.RData",
                            unifiedData_file="myDATA/DS.RData",
                            saveresultsfile=F,
                            timeoutput=F)
  {
  plotFilterName <- ifelse(filter,'Filtered','Unfiltered')
  saveName <- paste(myname,plotFilterName, sep='')


  # Study Importance of Image Features

  require(randomForest)
  require(caret)
  library(doMC)
  registerDoMC(2)

  source(file="my_imputations.r")

  applyCorFilter = filter
  results <- list()

  load(microdata_file) #DSm2: a dframe only with microar. data in proper form
  y <- factor(DSm2$class)
  ym  <- factor(DSm2$class,levels=c(1,0)) #melanoma class fist level in factor
  x <- subset(DSm2, select=c(-myOrigin,-class)) #only gene cols

  if (filter) {
    # nzv <- nearZeroVar(x) # zero vars
    xCor <- cor(x)
    highlyCorX <- findCorrelation(xCor, cutoff = .75)
    filteredX <- x[,-highlyCorX] # stay 482 from 1703 vars...
    x <- filteredX
  }

  if (! 'DS' %in% ls()) {load(unifiedData_file);print('DS loaded')}
  image_features <- setdiff(names(DS),names(DSm2))
  image_features <- setdiff(image_features,c("Grad.min"))
  cat("Tail of image_features:",tail(image_features))

  #-----------------------------------------------------------------------------
  #---------  varImportance with mean Imputation data --------------------------
  #-----------------------------------------------------------------------------
  DSmi  <- keeping.order(DS, ddply, .(class), colwise(f2))

  y2 <- factor(DSmi$class)
  y2m  <- factor(DSmi$class,levels=c(1,0)) #melanoma class fist level in factor
  x2 <- subset(DSmi, select=c(-class, -Grad.min)) # Grad.min is 0 everywhere

  if (filter) {
    filteredX <- x2[,-highlyCorX] # stay 513 from 1734 vars...  USE THE SAME highlyCorX as M.O. DATA...
    x2 <- filteredX
  }

  normalization <- preProcess(x2)
  x2 <- predict(normalization, x2)
  x2 <- as.data.frame(x2)



  if(timeoutput)cat('\nmean imp:',' ')
```

```
 
 results <- myImploop(x=x2, y=y2m, mycounts=myrepetitions, myrandom=F,
          timeoutput=timeoutput, res=results,
          label="mi", image_features=image_features)


 if(saveresultsfile)
 save(results, file = paste("myDATA/",saveName,".RData",sep=''))


 #----------------------------------------------------------------------
 #---------  varImportance with normal random Imputation data ------------------
 #----------------------------------------------------------------------
 if(timeoutput)cat('\nnormal random imp:',' ')

 results <- myImploop(x=NULL, y=NULL, mycounts=myrepetitions,
          timeoutput=timeoutput, res=results,
          label="ri", filter=filter,
          myrandom=T, unifiedDS=DS, imputfun=f2r,
          highlyCorX=highlyCorX,
          image_features=image_features
          )

 if(saveresultsfile)
 save(results, file = paste("myDATA/",saveName,".RData",sep=''))
 #----------------------------------------------------------------------
 #---------  varImportance with uniform sampling imputation --------------------
 #----------------------------------------------------------------------
 if(timeoutput)cat('\nuniform sampling imp:',' ')

 results <- myImploop(x=NULL, y=NULL, mycounts=myrepetitions,
            timeoutput=timeoutput, res=results,
            label="ui", filter=filter,
            myrandom=T, unifiedDS=DS, imputfun=f2u,
            highlyCorX=highlyCorX,
            image_features=image_features
            )

 if(saveresultsfile)
  save(results, file = paste("myDATA/",saveName,".RData",sep=''))
 #----------------------------------------------------------------------
 #---------  varImportance with bootstrap imputation ---------------------------
 #----------------------------------------------------------------------
 if(timeoutput)cat('\nbootstrap imp:',' ')

 results <- myImploop(x=NULL, y=NULL, mycounts=myrepetitions,
            timeoutput=timeoutput, res=results,
            label="bi", filter=filter,
            myrandom=T, unifiedDS=DS, imputfun=f2b,
            highlyCorX=highlyCorX,
            image_features=image_features
            )

 if(saveresultsfile)
  save(results, file = paste("myDATA/",saveName,".RData",sep=''))

 #Histogram plots
#   breaks = seq(from=0,to=30,by=10)
#   hist(results$omnum, xlim=c(0,30), ylim=c(0,11), col="red", breaks=20,
#        main="Histograms of var num",xlab='vars')
#   hist(results$minum, add=T, col=rgb(0, 1, 0, 0.5))
#   hist(results$rinum, add=T, col=rgb(0, 0, 1, 0.5))
#   legend("right",inset=.05, title="Dataset", c("only micro","mean imp","random imp"),
#          lty=c(1,1,1), lwd=2, col=c("red","blue","green"))

 # Density plots
#   plot(density(unlist(results$ri$I1)))
 if (makeplots){ # to update IS PREVIOUS FUNCTIONS from VarSelection
  jpeg(file = paste("myDATA/",plotFilterName,plotBestName,".jpeg",sep=""))

  plot(density(results$omnum), col="red", lwd=2, lty=2,ylim=c(0,1.2),
     main=paste("Densities of features numbers per dataset\n",
          plotFilterName, plotBestName),
     xlab='Selected features number, N=50',
     )
  lines(density(results$minum), col="blue", lwd=2, lty=3)
  lines(density(results$rinum), col="green", lwd=2, lty=4)
  lines(density(results$uinum), col="orange", lwd=2, lty=5)
  lines(density(results$binum), col="gray", lwd=2, lty=6)
  legend("right",inset=.05, title="Dataset", c("only micro","mean imp","random imp",
                        "uniform imp","bootstrap imp"),
      lty=c(2,3,4,5,6), lwd=2, col=c("red","blue","green","orange","gray"))
  dev.off()
 }
 return(results)
}

myImploop <- function(x=x, y=y, mycounts=mycounts,
          timeoutput=timeoutput, res=list(),
          label="", filter=filter,
          myrandom=F, unifiedDS=NULL, imputfun=NULL,
          highlyCorX=NULL, image_features=NULL
          )
 {
 ptm <- proc.time()[3]
 I1_features  <- list()
 I1 <- list() #just the positions of features
 I2 <- list()
 I2_features  <- list()
 I3 <- list()
 I3_features  <- list()
 I4 <- list()
 I4_features  <- list()

 for (i in 1:mycounts) {

  if(myrandom){
   DSri <- keeping.order(unifiedDS, ddply, .(class), colwise(imputfun)) #normal random imputation

   y <- factor(DSri$class)
   ym  <- factor(DSri$class,levels=c(1,0)) #melanoma class fist level in factor
   x <- subset(DSri, select=c(-class, -Grad.min)) # Grad.min is 0 everywhere

   if (filter) {
    filteredX <- x[,-highlyCorX] # stay .. from .. vars...
    x <- filteredX
   }

   normalization <- preProcess(x)
   x <- predict(normalization, x)
   x <- as.data.frame(x)
  }

  rf <- randomForest(x=x,y=y,strata=y,sampsize=c(18,18),importance=T)
  imp  <- rf$importance

  f_series <- names(sort(imp[,1],decreasing=T))
  I1[[i]] <- which(f_series %in% image_features)
  I1_features[[i]] <- f_series[I1[[i]]]

  f_series <- names(sort(imp[,2],decreasing=T))
  I2[[i]] <- which(f_series %in% image_features)
  I2_features[[i]] <- f_series[I2[[i]]]

  f_series <- names(sort(imp[,3],decreasing=T))
  I3[[i]] <- which(f_series %in% image_features)
  I3_features[[i]] <- f_series[I3[[i]]]

  f_series <- names(sort(imp[,4],decreasing=T))
  I4[[i]] <- which(f_series %in% image_features)
  I4_features[[i]] <- f_series[I4[[i]]]



```

```
    if(timeoutput)cat(i,' ')
  }
  if(timeoutput)cat('\n',label,' time:',round((proc.time()[3] - ptm)/60,2),' mins\n')

  #plot(density(only_micro_num))
  #plot(1:nrow(only_micro_res),only_micro_res[,2])

  res[[label]] <- list('I1'=I1, 'I2'=I2, 'I3'=I3, 'I4'=I4,
                       'I1_features'=I1_features,'I2_features'=I2_features,
                       'I3_features'=I3_features,'I4_features'=I4_features)

    return(res)

}

```

3.2 MY_IMPUTATIONS.R

```
library(plyr)

keeping.order <- function(data, fn, ...) {
  col <- 'MYCOLS'
  rn <- row.names(data)
  data[,col] <- 1:nrow(data)
  out <- fn(data, ...)
  if (!col %in% colnames(out)) stop("Ordering column not preserved by function")
  out <- out[order(out[,col]),]
  rownames(out) <- rn
  out[,col] <- NULL
  out
}
# mean imputation
f2 <- function(X)replace(X,is.na(X),mean(X, na.rm=T))

# random normal imputation
f2r <- function(X){
  cm <- mean(X, na.rm=T)
  csd <- sd(X, na.rm=T)
  isna <- is.na(X)
  c <- sum(isna)
  X[isna] <- rnorm(c,cm,csd)
  X
}

#my uniform sampling imputation
f2u <- function(X){
  ranges <- range(X, na.rm=T)
  isna <- is.na(X)
  c <- sum(isna)
  X[isna] <- runif(c,min=ranges[1],max=ranges[2])
  X
}

#my bootstrap imputation
f2b <- function(X){
  isna <- is.na(X)
  c <- sum(isna)
  X[isna] <- sample(x=X[!isna],size=c,replace=T)
  X
}
```

3.3 WORKFLOW_IMAGE_IMPORTANCE.R

```
# draw the 2 plots (Filtered, Unfiltered). Are density plots of importance ranks
# for image-derived features (ranking in the x-axis is in decreasing order of
# importance)

source(file="myVarImportance.r")

```

```
screen <- T

filtered <- myVarImportance(filter=T,timeoutput=T)
unfiltered <- myVarImportance(filter=F,timeoutput=T)

imp_results <- list('filtered'=filtered,'unfiltered'=unfiltered)

save(imp_results, file = paste("myDATA/varImageImportance.RData",sep=''))

plot(density(unlist(imp_results$unfiltered$ri$I4)))
# Importance names:"0","1","MeanDecreaseAccuracy","MeanDecreaseGini"

print("In position lower than 200,")
less200=lapply(imp_results$filtered$bi$I4,function(x)ifelse(x<200,T,F))
sort(table(unlist(imp_results$filtered$bi$I4_features)[unlist(less200)]),decreasing=T)

if (! 'imp_results' %in% ls()) {load("myDATA/varImageImportance.RData")}
#Densities of ranks for image-derived features
opar <- par(no.readonly=T)


if (!screen)
  jpeg(file = "myDATA/imageFeaturesImportance.jpeg")
par(mfrow=c(1,2))

plot(density(unlist(imp_results$filtered$ri$I4)), col="red",
     lwd=1, lty=1,ylim=c(0,0.035),
     main="",
     xlab=""
     )
title(main = paste("Filtered"),
      cex.main=0.9, cex.sub=0.75,
      xlab="Ranks of features' importance\nN=50", cex.lab=0.9)
lines(density(unlist(imp_results$filtered$mi$I4)), col="blue", lwd=1, lty=3)
lines(density(unlist(imp_results$filtered$ui$I4)), col="green", lwd=1, lty=6)
lines(density(unlist(imp_results$filtered$bi$I4)), col="orange", lwd=1, lty=5)

legend("left",inset=.03, title="Imputation",
       c("nr.i","mi","ui","bi"),
       lty=c(1,3,6,5), lwd=2, col=c("red","blue","green","orange"),cex=0.7)

plot(density(unlist(imp_results$unfiltered$ri$I4)), col="red",
     lwd=1, lty=1,ylim=c(0,0.035),
     main="",
     xlab=""
     )
title(main = "UnFiltered",
      cex.main=0.9, cex.sub=0.75,
      xlab="Ranks of features' importance\nN=50", cex.lab=0.9)
lines(density(unlist(imp_results$unfiltered$mi$I4)), col="blue", lwd=1, lty=3)
lines(density(unlist(imp_results$unfiltered$ui$I4)), col="green", lwd=1, lty=6)
lines(density(unlist(imp_results$unfiltered$bi$I4)), col="orange", lwd=1, lty=5)

legend("left",inset=.03, title="Imputation",
       c("nr.i","mi","ui","bi"),
       lty=c(1,3,6,5), lwd=2, col=c("red","blue","green","orange"),cex=0.7)

if (!screen)  dev.off()

par(opar)

## Find image features Importance only in original image data
if (! 'DSi' %in% ls()) {load("myDATA/DSi.RData");print('DSi loaded')}

rownames(DSi) <- DSi$NAME
y2i  <- factor(DSi$class,levels=c(1,0)) #melanoma class fist level in factor
DSi2 <- subset(DSi, select=c(-origin,-NAME,-class)) #only image attr cols
rf <- randomForest(x=DSi2,y=y2i,strata=y2i,sampsize=c(18,18),importance=T)
imp  <- rf$importance
#f_series <- names(sort(imp[,4],decreasing=T))
head(data.frame("features"=names(sort(imp[,4],decreasing=T)),"imp"=sort(imp[,4],decreasing=T)))

```

3.4 COMPARE_LOOPS_WITH_REPETITIONS.R

```
# 2ND EDITION. Random Forest oob resampling.
# iterated models

library(caret)
# library(doMC)
# registerDoMC(2)

source(file="pickST.r")
source(file="my_imputations.r")

applyCorFilter = T
subsets <- c(1:10,15,20,25,30,35,40,45,50)

load("myDATA/DSm_ready.RData") #DSm2: a dframe only with microar. data in proper form
y <- factor(DSm2$class)
ym  <- factor(DSm2$class,levels=c(1,0)) #melanoma class fist level in factor
x <- subset(DSm2, select=c(-myOrigin,-class)) #only gene cols

if (applyCorFilter) {
  # nzv <- nearZeroVar(x) # zero vars
  xCor <- cor(x)
  highlyCorX <- findCorrelation(xCor, cutoff = .75)
  filteredX <- x[,-highlyCorX] # stay 482 from 1703 vars...
  x <- filteredX
}

normalization <- preProcess(x)
x <- predict(normalization, x)
x <- as.data.frame(x)

# ----Caret Parameters------
rfFuncs$summary <- twoClassSummary
rfFuncs$selectSize <- pickST # allow 5% tolerance #pickSizeTolerance   #pickSizeBest
ctrl <- rfeControl(functions = rfFuncs, method='repeatedcv',verbose=F,returnResamp='none',repeats=1)
mycounts <- 2
# -------------------------

myresnum <- rep(0,mycounts)
myperf <- rep(0,mycounts)
myresvars <- list()
cat('only micro:',' ')
ptm <- proc.time()

set.seed(1)

for (i in 1:mycounts) {
  rfmodel <- rfe(x,ym,sizes=subsets,rfeControl=ctrl,metric="ROC",maximize=T,strata=ym,sampsize=c(18,18))
  myresnum[i] <- rfmodel$bestSubset
  myresvars[[i]] <- rfmodel$optVariables
  myperf[i] <- subset(rfmodel$results,
                    subset=Variables==rfmodel$bestSubset,
                    select=ROC)
  cat(i,' ')
}
cat('only micro time:',proc.time() - ptm)

only_micro_num <- myresnum
#plot(density(only_micro_num))
only_micro_res <- as.data.frame(table(unlist(myresvars)),stringsAsFactors=F)
only_micro_res <- only_micro_res[order(only_micro_res[2],decreasing=T),]
#plot(1:nrow(only_micro_res),only_micro_res[,2])

results = list(omres = only_micro_res,omnum = only_micro_num,
               omperf = unlist(myperf))

```

```
#REPETITIONS
set.seed(1)
ctrl2 <- rfeControl(functions = rfFuncs, method='repeatedcv',verbose=F,repeats=2) #returnResamp='none'
rfmodel2 <- rfe(x,ym,sizes=subsets,rfeControl=ctrl2,metric="ROC",maximize=T,strata=ym,sampsize=c(18,18))



#Histogram plots
breaks = seq(from=0,to=30,by=10)
hist(results$omnum, xlim=c(0,30), ylim=c(0,11), col="red", breaks=20,
     main="Histograms of var num",xlab='vars')
hist(results$minum, add=T, col=rgb(0, 1, 0, 0.5))
hist(results$rinum, add=T, col=rgb(0, 0, 1, 0.5))
legend("right",inset=.05, title="Dataset", c("only micro","mean imp","random imp"),
       lty=c(1,1,1), lwd=2, col=c("red","blue","green"))

# Density plots
jpeg(file = "myDATA/FilteredTolerance.jpeg")
plot(density(results$omnum), col="red", lwd=2,ylim=c(0,1.2),
     main="Densities of features numbers per dataset\nFiltered Tolerance",
     xlab='Selected features number, N=50',
     )
lines(density(results$minum), col="blue", lwd=2)
lines(density(results$rinum), col="green", lwd=2)
legend("right",inset=.05, title="Dataset", c("only micro","mean imp","random imp"),
       lty=c(1,1,1), lwd=2, col=c("red","blue","green"))
dev.off()
```

3.5 BIOMARKERS_PERFORMANCE.R

```
require(randomForest)
require(caret)
require(MASS)

source(file="pickST.r")
source(file="my_imputations.r")
source(file="confusion.r")

if (! 'DS' %in% ls()) {load("myDATA/DS.RData");print('DS loaded')}
if (! 'DSm2' %in% ls()) load("myDATA/DSm_ready.RData") #DSm2: a dframe only with microar. data in proper form
if (! 'DSi' %in% ls()) {load("myDATA/DSi.RData");print('DSi loaded (image data)')}
if (! 'ldatopfeatures_results' %in% ls()) {load("myDATA/ldatopfeatures_results.RData");print('ldatopfeatures_results loaded')}

biomarkers <- list("names"=ldatopfeatures_results$top_ui$topmeans)

biomarkersPerformance <- function(DS, biomarkers=list('names'=c(),'loadings'=c()),
                                  mytimes=50){

  #run LDA CV and RF (OOB) only with biom features
  res_om <- perfeval(myfun="",mytimes=mytimes,original=T)
  cat("om median accuracy: LDA=", median(res_om$total_lda), " RF=", median(res_om$total_rf))
  res_oi <- perfeval(myfun="",mytimes=mytimes,original=T, image=T)
  cat("oi median accuracy: LDA=", median(res_oi$total_lda), " RF=", median(res_oi$total_rf))
  res_mi <- perfeval(myfun=f2,mytimes=mytimes)
  cat("mi median accuracy: LDA=", median(res_mi$total_lda), " RF=", median(res_mi$total_rf))
  res_nri <- perfeval(myfun=f2r,mytimes=mytimes)
  cat("nri median accuracy: LDA=", median(res_nri$total_lda), " RF=", median(res_nri$total_rf))
  res_ui <- perfeval(myfun=f2u,mytimes=mytimes)
  cat("ui median accuracy: LDA=", median(res_ui$total_lda), " RF=", median(res_ui$total_rf))
  res_bi <- perfeval(myfun=f2b,mytimes=mytimes)
  cat("bi median accuracy: LDA=", median(res_bi$total_lda), " RF=", median(res_bi$total_rf))

  boxplot(cbind("om (lda)"=res_om$total_lda, "om (rf)"=res_om$total_rf,
                "oi (lda)"=res_oi$total_lda, "oi (rf)"=res_oi$total_rf,
                "mi (lda)"=res_mi$total_lda, "mi (rf)"=res_mi$total_rf,
                "nri (lda)"=res_mi$total_lda, "nri (rf)"=res_mi$total_rf,
                "ui (lda)"=res_mi$total_lda, "ui (rf)"=res_mi$total_rf,
                "bi (lda)"=res_mi$total_lda, "bi (rf)"=res_mi$total_rf),
          las=2, cex=0.8, adj=0, cex.axis=0.8)

  title(main="Accuracies for melanoma class",ylab="Accuracy (%)")

#   #run LDA with previous biom. loadings
#   x_lda <- lda(x2,y2m)
#   x_lda$scaling[!row.names(x_lda$scaling) %in% biomarkers$names]  <- 0

}

perfeval <- function(myfun, mytimes=50, original_data=F, image=F){
  res=list("total_lda"=rep(0,times=mytimes),"total_rf"=rep(0,times=mytimes))
  for (i in 1:mytimes){
    if (original_data){
        mybiomarkers <- biomarkers$names
        if (image){
          mybiomarkers <- mybiomarkers[mybiomarkers %in% colnames(DSi)]
          selrows <- !grepl("[G*]",row.names(DS))
        } else{
          mybiomarkers <- mybiomarkers[mybiomarkers %in% colnames(DSm2)]
          selrows <- grepl("[G*]",row.names(DS))
        }

        DSmi <-  DS[selrows, colnames(DS) %in% mybiomarkers]
        y2 <- factor(DS$class[selrows])
        y2m  <- factor(DS$class[selrows],levels=c(1,0)) #melanoma class fist level in factor
        x2 <- DSmi
    } else {
      DSmi  <- keeping.order(DS[,c(biomarkers$names, 'class', 'Grad.min')],
                             ddply, .(class), colwise(myfun))
      y2 <- factor(DSmi$class)
      y2m  <- factor(DSmi$class,levels=c(1,0)) #melanoma class fist level in factor
      x2 <- subset(DSmi, select=c(-class, -Grad.min)) # Grad.min is 0 everywhere
    }

    normalization <- preProcess(x2)
    x2 <- predict(normalization, x2)
    x2 <- as.data.frame(x2)

    xmi_ldacv <- lda(x2,y2m,CV=T)
    res$total_lda[i] <- confusion(y2m,xmi_ldacv$class,printit=F)[1,1]

    rf <- randomForest(x=x2,y=y2m,strata=y2m,sampsize=c(18,18),importance=F)
    res$total_rf[i] <- 1 - rf$con[1,3]
  }
  return(res)
}
```

BIBLIOGRAPHY

[1] S. Cerutti, D. Hoyer, and A. Voss, "Multiscale, multiorgan and multivariate complexity analyses of cardiovascular regulation," *Philosophical transactions. Series A, Mathematical, physical, and engineering sciences,* vol. 367, no. 1892, pp. 1337-58, Apr 13, 2009.

[2] Q. Tian, N. D. Price, and L. Hood, "Systems cancer medicine: towards realization of predictive, preventive, personalized and participatory (P4) medicine," *Journal of internal medicine,* vol. 271, no. 2, pp. 111-21, Feb, 2012.

[3] J. B. Bassingthwaighte, "Strategies for the physiome project," *Annals of Biomedical Engineering*, vol. 28, no. 8, pp. 1043-1058, Aug, 2000.

[4] P. J. Hunter, "Modeling human physiology: The IUPS/EMBS physiome project," *Proceedings of the Ieee,* vol. 94, no. 4, pp. 678-691, Apr, 2006.

[5] J. W. Fenner, B. Brook, G. Clapworthy, P. V. Coveney, V. Feipel, H. Gregersen, D. R. Hose, P. Kohl, P. Lawford, K. M. McCormack, D. Pinney, S. R. Thomas, S. Van Sint Jan, S. Waters, and M. Viceconti, "The EuroPhysiome, STEP and a roadmap for the virtual physiological human," *Philosophical transactions. Series A, Mathematical, physical, and engineering sciences,* vol. 366, no. 1878, pp. 2979-99, Sep 13, 2008.

[6] STEPConsortium. "Seeding the EuroPhysiome: a roadmap to the virtual physiological human," http://www.europhysiome.org/roadmap.

[7] A. C. Guyton, T. G. Coleman, and H. J. Granger, "Circulation: overall regulation," *Annu Rev Physiol,* vol. 34, pp. 13-46, 1972.

[8] E. Welsh, M. Jirotka, and D. Gavaghan, "Post-genomic science: cross-disciplinary and large-scale collaborative research and its organizational and technological challenges for the scientific research process," *Philos Transact A Math Phys Eng Sci,* vol. 364, no. 1843, pp. 1533-49, Jun 15, 2006.

[9] P. Hunter, P. V. Coveney, B. de Bono, V. Diaz, J. Fenner, A. F. Frangi, P. Harris, R. Hose, P. Kohl, P. Lawford, K. McCormack, M. Mendes, S. Omholt, A. Quarteroni, J. Skar, J. Tegner, S. Randall Thomas, I. Tollis, I. Tsamardinos, J. H. van Beek, and M. Viceconti, "A vision and strategy for the virtual physiological human in 2010 and beyond," *Philos Transact A Math Phys Eng Sci,* vol. 368, no. 1920, pp. 2595-614, Jun 13, 2010.

[10] M. Kanehisa, S. Goto, S. Kawashima, and A. Nakaya, "The KEGG databases at GenomeNet," *Nucleic Acids Res,* vol. 30, no. 1, pp. 42-6, Jan 1, 2002.

[11] M. Kanehisa, S. Goto, S. Kawashima, Y. Okuno, and M. Hattori, "The KEGG resource for deciphering the genome," *Nucleic Acids Res,* vol. 32, no. Database issue, pp. D277-80, Jan 1, 2004.

[12] M. Kanehisa, S. Goto, M. Hattori, K. F. Aoki-Kinoshita, M. Itoh, S. Kawashima, T. Katayama, M. Araki, and M. Hirakawa, "From genomics to chemical genomics: new developments in KEGG," *Nucleic Acids Res,* vol. 34, no. Database issue, pp. D354-7, Jan 1, 2006.

[13] M. Kanehisa, S. Goto, M. Furumichi, M. Tanabe, and M. Hirakawa, “KEGG for representation and analysis of molecular networks involving diseases and drugs,” *Nucleic Acids Res*, vol. 38, no. Database issue, pp. D355-60, Jan, 2010.

[14] S. Okuda, T. Yamada, M. Hamajima, M. Itoh, T. Katayama, P. Bork, S. Goto, and M. Kanehisa, “KEGG Atlas mapping for global analysis of metabolic pathways,” *Nucleic Acids Res*, vol. 36, no. Web Server issue, pp. W423-6, Jul 1, 2008.

[15] M. Kanehisa, and S. Goto, “KEGG: kyoto encyclopedia of genes and genomes,” *Nucleic Acids Res*, vol. 28, no. 1, pp. 27-30, Jan 1, 2000.

[16] www. "KEGG FTP," http://www.genome.jp/kegg/download/ftp.html.

[17] A. Funahashi, Jouraku, A., and Kitano, H., "Converting KEGG pathway database to SBML." p. 8th Annual International Conference on Research in Computational Molecular Biology (RECOMB 2004).

[18] www. "systems-biology.org - Model Repository," http://systems-biology.org/001/001.html.

[19] www. "KEGG API," http://www.genome.jp/kegg/soap/.

[20] www. "KGML," http://www.genome.jp/kegg/xml/.

[21] M. Hucka, A. Finney, H. M. Sauro, H. Bolouri, J. C. Doyle, H. Kitano, A. P. Arkin, B. J. Bornstein, D. Bray, A. Cornish-Bowden, A. A. Cuellar, S. Dronov, E. D. Gilles, M. Ginkel, V. Gor, Goryanin, II, W. J. Hedley, T. C. Hodgman, J. H. Hofmeyr, P. J. Hunter, N. S. Juty, J. L. Kasberger, A. Kremling, U. Kummer, N. Le Novere, L. M. Loew, D. Lucio, P. Mendes, E. Minch, E. D. Mjolsness, Y. Nakayama, M. R. Nelson, P. F. Nielsen, T. Sakurada, J. C. Schaff, B. E. Shapiro, T. S. Shimizu, H. D. Spence, J. Stelling, K. Takahashi, M. Tomita, J. Wagner, and J. Wang, “The systems biology markup language (SBML): a medium for representation and exchange of biochemical network models,” *Bioinformatics*, vol. 19, no. 4, pp. 524-31, Mar 1, 2003.

[22] www. "ERATO-SORST Kitano Symbiotic Systems Project," http://sbi.jp/symbio/symbio2/objectives.html.

[23] A. Funahashi, Jouraku, A., and Kitano, H., "Converting KEGG pathway database to SBML." pp. 5th International Conference on Systems Biology (ICSB 2004), October, 2004.

[24] A. Funahashi, M. Morohashi, H. Kitano, and N. Tanimura, “CellDesigner: a process diagram editor for gene-regulatory and biochemical networks,” *BIOSILICO*, vol. 1, no. 5, pp. 159-162, 5 November 2003, 2003.

[25] M. Hucka, A. Finney, H. M. Sauro, H. Bolouri, J. Doyle, and H. Kitano, “The ERATO Systems Biology Workbench: enabling interaction and exchange between software tools for computational biology,” *Pac Symp Biocomput*, pp. 450-61, 2002.

[26] C. Klukas, and F. Schreiber, “Dynamic exploration and editing of KEGG pathway diagrams,” *Bioinformatics*, vol. 23, no. 3, pp. 344-50, Feb 1, 2007.

[27] J. D. Zhang, and S. Wiemann, “KEGGgraph: a graph approach to KEGG PATHWAY in R and bioconductor,” *Bioinformatics*, vol. 25, no. 11, pp. 1470-1, Jun 1, 2009.

[28] V. Lacroix, L. Cottret, P. Thébault, and M.-F. Sagot, “An introduction to metabolic networks and their structural analysis.,” *IEEE/ACM Transactions*

on Computational Bioloy and Bioinformatics, vol. 5, no. 4, pp. 594-617., October-December 2008, 2008.

[29] L. du Plessis, N. Skunca, and C. Dessimoz, "The what, where, how and why of gene ontology--a primer for bioinformaticians," *Brief Bioinform,* vol. 12, no. 6, pp. 723-35, Nov, 2011.

[30] M. Ashburner, C. A. Ball, J. A. Blake, D. Botstein, H. Butler, J. M. Cherry, A. P. Davis, K. Dolinski, S. S. Dwight, J. T. Eppig, M. A. Harris, D. P. Hill, L. Issel-Tarver, A. Kasarskis, S. Lewis, J. C. Matese, J. E. Richardson, M. Ringwald, G. M. Rubin, and G. Sherlock, "Gene ontology: tool for the unification of biology. The Gene Ontology Consortium," *Nat Genet,* vol. 25, no. 1, pp. 25-9, May, 2000.

[31] "Creating the gene ontology resource: design and implementation," *Genome Res,* vol. 11, no. 8, pp. 1425-33, Aug, 2001.

[32] "The Gene Ontology (GO) project in 2006," *Nucleic Acids Res,* vol. 34, no. Database issue, pp. D322-6, Jan 1, 2006.

[33] "The Gene Ontology project in 2008," *Nucleic Acids Res,* vol. 36, no. Database issue, pp. D440-4, Jan, 2008.

[34] "The Gene Ontology in 2010: extensions and refinements," *Nucleic Acids Res,* vol. 38, no. Database issue, pp. D331-5, Jan, 2010.

[35] "The Gene Ontology: enhancements for 2011," *Nucleic Acids Res,* vol. 40, no. Database issue, pp. D559-64, Jan, 2012.

[36] M. A. Harris, J. Clark, A. Ireland, J. Lomax, M. Ashburner, R. Foulger, K. Eilbeck, S. Lewis, B. Marshall, C. Mungall, J. Richter, G. M. Rubin, J. A. Blake, C. Bult, M. Dolan, H. Drabkin, J. T. Eppig, D. P. Hill, L. Ni, M. Ringwald, R. Balakrishnan, J. M. Cherry, K. R. Christie, M. C. Costanzo, S. S. Dwight, S. Engel, D. G. Fisk, J. E. Hirschman, E. L. Hong, R. S. Nash, A. Sethuraman, C. L. Theesfeld, D. Botstein, K. Dolinski, B. Feierbach, T. Berardini, S. Mundodi, S. Y. Rhee, R. Apweiler, D. Barrell, E. Camon, E. Dimmer, V. Lee, R. Chisholm, P. Gaudet, W. Kibbe, R. Kishore, E. M. Schwarz, P. Sternberg, M. Gwinn, L. Hannick, J. Wortman, M. Berriman, V. Wood, N. de la Cruz, P. Tonellato, P. Jaiswal, T. Seigfried, and R. White, "The Gene Ontology (GO) database and informatics resource," *Nucleic Acids Res,* vol. 32, no. Database issue, pp. D258-61, Jan 1, 2004.

[37] R. Rentzsch, and C. A. Orengo, "Protein function prediction--the power of multiplicity," *Trends Biotechnol,* vol. 27, no. 4, pp. 210-9, Apr, 2009.

[38] A. J. Richards, B. Muller, M. Shotwell, L. A. Cowart, B. Rohrer, and X. Lu, "Assessing the functional coherence of gene sets with metrics based on the Gene Ontology graph," *Bioinformatics,* vol. 26, no. 12, pp. i79-87, Jun 15, 2010.

[39] E. Schadt, and P. Lum, "Reverse engineering gene networks to identify key drivers of complex disease phenotypes," *Journal of Lipid Research,* vol. 47, pp. 2601-2613, Dec 2006, 2006.

[40] A. L. Barabasi, "Scale-free networks: a decade and beyond," *Science,* vol. 325, no. 5939, pp. 412-3, Jul 24, 2009.

[41] M. Suderman, and M. Hallett, "Tools for visually exploring biological networks," *Bioinformatics,* vol. 23, no. 20, pp. 2651-9, Oct 15, 2007.

[42] W. Huang da, B. T. Sherman, and R. A. Lempicki, “Bioinformatics enrichment tools: paths toward the comprehensive functional analysis of large gene lists,” *Nucleic Acids Res,* vol. 37, no. 1, pp. 1-13, Jan, 2009.

[43] Y. A. Chen, L. P. Tripathi, and K. Mizuguchi, “TargetMine, an integrated data warehouse for candidate gene prioritisation and target discovery,” *PLoS One,* vol. 6, no. 3, pp. e17844, 2011.

[44] J. E. Hutz, A. T. Kraja, H. L. McLeod, and M. A. Province, “CANDID: a flexible method for prioritizing candidate genes for complex human traits,” *Genet Epidemiol,* vol. 32, no. 8, pp. 779-90, Dec, 2008.

[45] N. Tiffin, E. Adie, F. Turner, H. G. Brunner, M. A. van Driel, M. Oti, N. Lopez-Bigas, C. Ouzounis, C. Perez-Iratxeta, M. A. Andrade-Navarro, A. Adeyemo, M. E. Patti, C. A. Semple, and W. Hide, “Computational disease gene identification: a concert of methods prioritizes type 2 diabetes and obesity candidate genes,” *Nucleic Acids Res,* vol. 34, no. 10, pp. 3067-81, 2006.

[46] L. C. Tranchevent, F. B. Capdevila, D. Nitsch, B. De Moor, P. De Causmaecker, and Y. Moreau, “A guide to web tools to prioritize candidate genes,” *Brief Bioinform,* vol. 12, no. 1, pp. 22-32, Jan, 2011.

[47] P. Maji, “f-Information measures for efficient selection of discriminative genes from microarray data,” *IEEE Trans Biomed Eng,* vol. 56, no. 4, pp. 1063-9, Apr, 2009.

[48] C. Gao, X. Dang, Y. Chen, and D. Wilkins, “Graph ranking for exploratory gene data analysis,” *BMC Bioinformatics,* vol. 10 Suppl 11, pp. S19, 2009.

[49] P. C. Ma, and K. C. Chan, “A novel approach for discovering overlapping clusters in gene expression data,” *IEEE Trans Biomed Eng,* vol. 56, no. 7, pp. 1803-9, Jul, 2009.

[50] T. Barrett, and R. Edgar, “Mining microarray data at NCBI's Gene Expression Omnibus (GEO),” *Methods Mol Biol,* vol. 338, pp. 175-90, 2006.

[51] T. Barrett, D. B. Troup, S. E. Wilhite, P. Ledoux, C. Evangelista, I. F. Kim, M. Tomashevsky, K. A. Marshall, K. H. Phillippy, P. M. Sherman, R. N. Muertter, M. Holko, O. Ayanbule, A. Yefanov, and A. Soboleva, “NCBI GEO: archive for functional genomics data sets--10 years on,” *Nucleic Acids Res,* vol. 39, no. Database issue, pp. D1005-10, Jan, 2011.

[52] T. Barrett, D. B. Troup, S. E. Wilhite, P. Ledoux, D. Rudnev, C. Evangelista, I. F. Kim, A. Soboleva, M. Tomashevsky, and R. Edgar, “NCBI GEO: mining tens of millions of expression profiles--database and tools update,” *Nucleic Acids Res,* vol. 35, no. Database issue, pp. D760-5, Jan, 2007.

[53] R. Edgar, M. Domrachev, and A. E. Lash, “Gene Expression Omnibus: NCBI gene expression and hybridization array data repository,” *Nucleic Acids Res,* vol. 30, no. 1, pp. 207-10, Jan 1, 2002.

[54] A. Brazma, P. Hingamp, J. Quackenbush, G. Sherlock, P. Spellman, C. Stoeckert, J. Aach, W. Ansorge, C. A. Ball, H. C. Causton, T. Gaasterland, P. Glenisson, F. C. Holstege, I. F. Kim, V. Markowitz, J. C. Matese, H. Parkinson, A. Robinson, U. Sarkans, S. Schulze-Kremer, J. Stewart, R. Taylor, J. Vilo, and M. Vingron, “Minimum information about a microarray experiment (MIAME)-toward standards for microarray data,” *Nat Genet,* vol. 29, no. 4, pp. 365-71, Dec, 2001.

[55] J. W. Fenner, B. Brook, G. Clapworthy, P. V. Coveney, V. Feipel, H. Gregersen, D. R. Hose, P. Kohl, P. Lawford, K. M. McCormack, D. Pinney, S. R. Thomas, S. Van Sint Jan, S. Waters, and M. Viceconti, “The EuroPhysiome, STEP and a roadmap for the virtual physiological human,” *Philos Transact A Math Phys Eng Sci*, vol. 366, no. 1878, pp. 2979-99, Sep 13, 2008.

[56] T. Rohlfing, A. Pfefferbaum, E. V. Sullivan, and C. R. Maurer, “Information fusion in biomedical image analysis: combination of data vs. combination of interpretations,” *Inf Process Med Imaging,* vol. 19, pp. 150-61, 2005.

[57] R. Haapanen, and S. Tuominen, “Data combination and feature selection for multi-source forest inventory,” *Photogrammetric Engineering and Remote Sensing,* vol. 74, no. 7, pp. 869-880, 2008.

[58] J. L. Jesneck, L. W. Nolte, J. A. Baker, C. E. Floyd, and J. Y. Lo, “Optimized approach to decision fusion of heterogeneous data for breast cancer diagnosis,” *Med Phys*, vol. 33, no. 8, pp. 2945-54, Aug, 2006.

[59] G. Lee, S. Doyle, J. Monaco, A. Madabhushi, M. D. Feldman, S. R. Master, and J. E. Tomaszewski, “A knowledge representation framework for integration, classification of multi-scale imaging and non-imaging data: preliminary results in predicting prostate cancer recurrence by fusing mass spectrometry and histology,” in Proceedings of the Sixth IEEE international conference on Symposium on Biomedical Imaging: From Nano to Macro, Boston, Massachusetts, USA, 2009, pp. 77-80.

[60] P. Tiwari, S. Viswanath, G. Lee, and A. Madabhushi, “Multi-Modal Data Fusion Schemes For Integrated Classification Of Imaging And Non-Imaging Biomedical Data,” in IEEE International Symposium on Biomedical Imaging (ISBI), 2011, pp. 165-168.

[61] M. Balázs, S. Ecsedi, L. Vízkeleti, and A. Bégány, "Genomics of Human Malignant Melanoma " *Breakthroughs in Melanoma Research*, Breakthroughs in Melanoma Research Y. Tanaka, ed., InTech, 2011.

[62] J. Timar, B. Gyorffy, and E. Raso, “Gene signature of the metastatic potential of cutaneous melanoma: too much for too little?,” *Clin Exp Metastasis,* vol. 27, no. 6, pp. 371-87, Aug, 2010.

[63] W. K. Martins, G. H. Esteves, O. M. Almeida, G. G. Rezze, G. Landman, S. M. Marques, A. F. Carvalho, L. R. LF, J. P. Duprat, and B. S. Stolf, “Gene network analyses point to the importance of human tissue kallikreins in melanoma progression,” *BMC Med Genomics,* vol. 4, pp. 76, 2011.

[64] M. Ogorzałek, L. Nowak, G. Surówka, and A. Alekseenko, "Modern Techniques for Computer-Aided Melanoma Diagnosis," *Melanoma in the Clinic - Diagnosis, Management and Complications of Malignancy*, Melanoma in the Clinic - Diagnosis, Management and Complications of Malignancy M. Murph, ed., InTech, 2011.

[65] I. Maglogiannis, and C. N. Doukas, “Overview of advanced computer vision systems for skin lesions characterization,” *IEEE Trans Inf Technol Biomed,* vol. 13, no. 5, pp. 721-33, Sep, 2009.

[66] Y. Saeys, I. Inza, and P. Larranaga, “A review of feature selection techniques in bioinformatics,” *Bioinformatics,* vol. 23, no. 19, pp. 2507-17, Oct 1, 2007.

[67] L. Breiman, "Random Forests," *Machine Learning*, 2001, pp. 5-32.

[68] Y. Sun, A. K. C. Wong, and M. S. Kamel, “Classification of imbalanced data: A review,” *International Journal of Pattern Recognition and Artificial Intelligence,* vol. 23, no. 4, pp. 687-719, 2009.

[69] A. Liaw, and M. Wiener, “Classification and Regression by randomForest,” *R News,* vol. 2, no. 3, pp. 18-22, 2002.

[70] C. Chen, A. Liaw, and L. Breiman, "Using Random Forest to Learn Imbalanced Data," http://www.stat.berkeley.edu/users/chenchao/666.pdf, 2004].

[71] I. Kanaris, K. Moutselos, A. Chatziioannou, I. Maglogiannis, and F. N. Kolisis, "Building in-silico pathway SBML models from heterogeneous sources," *8th IEEE International Conference on BioInformatics and BioEngineering (BIBE 2008).* , 2008, pp. 1-6.

[72] N. Rodriguez, M. Donizelli, and N. Le Novere, “SBMLeditor: effective creation of models in the Systems Biology Markup language (SBML),” *BMC Bioinformatics,* vol. 8, pp. 79, 2007.

[73] K. Moutselos, I. Maglogiannis, and A. Chatziioannou, “Delineation and interpretation of gene networks towards their effect in cellular physiology- a reverse engineering approach for the identification of critical molecular players, through the use of ontologies,” *Conf Proc IEEE Eng Med Biol Soc,* vol. 2010, pp. 6709-12, 2010.

[74] P. Resnik, "Using information content to evaluate semantic similarity in a taxonomy," *Proc. of the 14th International Joint Conference on Artificial Intelligence.* pp. 448-453.

[75] "R: A Language and Environment for Statistical Computing," http://www.R-project.org.

[76] "Python Programming Language," www.python.org.

[77] M. F. Sanner, “Python: a programming language for software integration and development,” *J Mol Graph Model,* vol. 17, no. 1, pp. 57-61, Feb, 1999.

[78] M. DiPierro, “web2py for Scientific Applications,” *Computing in Science and Engineering,* vol. 13, no. 2, pp. 64-69, 2011.

[79] C. Pesquita, D. Faria, A. O. Falcao, P. Lord, and F. M. Couto, “Semantic similarity in biomedical ontologies,” *PLoS Comput Biol,* vol. 5, no. 7, pp. e1000443, Jul, 2009.

[80] G. J. Myatt, *Making Sense of Data: A Practical Guide to Exploratory Data Analysis and Data Mining,* p.^pp. 111-120: Wiley-Interscience, 2006.

[81] T. Segaran, *Programming Collective Intelligence: Building Smart Web 2.0 Applications*, p.^pp. 33: O'Reilly Media, 2007.

[82] C. Bettembourg. "go2Sim," https://sourceforge.net/projects/go2sim/.

[83] "MySQL," www.mysql.com.

[84] X. Guo. "SemSim: Gene Ontology-based Semantic Similarity Measures," http://www.bioconductor.org/packages/2.2/bioc/html/SemSim.html.

[85] A. Schlicker, and M. Albrecht, “FunSimMat update: new features for exploring functional similarity,” *Nucleic Acids Res,* vol. 38, no. Database issue, pp. D244-8, Jan, 2010.

[86] R. Balaji, R. Rajesh, R. David, and N. Sreevani. "Pancreatic Cancer Gene Database," http://www.bioinformatics.org/pcgdb/index.htm.

[87] R. Balaji, R. Rajesh, R. David, and N. Sreevani. "Leukemia gene database," http://www.bioinformatics.org/legend/leuk_db.htm.

[88] M. Maragoudakis, and I. Maglogiannis, “Skin lesion diagnosis from images using novel ensemble classification techniques,” in 10th IEEE EMBS International Conference on Information Technology Applications in Biomedicine, Corfu, Greece, 2010.

[89] D. Talantov, A. Mazumder, J. X. Yu, T. Briggs, Y. Jiang, J. Backus, D. Atkins, and Y. Wang, “Novel genes associated with malignant melanoma but not benign melanocytic lesions,” *Clin Cancer Res,* vol. 11, no. 20, pp. 7234-42, Oct 15, 2005.

[90] S. Davis, and P. Meltzer, “GEOquery: a bridge between the Gene Expression Omnibus (GEO) and BioConductor,” *Bioinformatics,* vol. 14, pp. 1846-1847, 2007.

[91] G. K. Smyth, "Limma: linear models for microarray data," *Bioinformatics and Computational Biology Solutions using R and Bioconductor*, pp. 397-420, New York: Springer, 2005.

[92] R. C. Gentleman, V. J. Carey, D. M. Bates, and others, “Bioconductor: Open software development for computational biology and bioinformatics,” *Genome Biology,* vol. 5, pp. R80, 2004.

[93] N. GEO. "GEO2R "; http://www.ncbi.nlm.nih.gov/geo/info/geo2r.html.

[94] N. J. Horton, and K. P. Kleinman, “Much ado about nothing: A comparison of missing data methods and software to fit incomplete data regression models,” *Am Stat*, vol. 61, no. 1, pp. 79-90, Feb, 2007.

[95] H. Wickham, “The Split-Apply-Combine Strategy for Data Analysis,” *Journal of Statistical Software,* vol. 40, no. 1, pp. 1-29, 2011.

[96] M. Kuhn, Contributions:, S. Weston, A. Williams, C. Keefer, and A. Engelhardt, "caret: Classification and Regression Training," http://CRAN.R-project.org/package=caret, 2012].

[97] T. Fawcett, “An introduction to ROC analysis,” *Pattern Recogn. Lett.,* vol. 27, no. 8, pp. 861-874, 2006.

[98] W. N. Venables, and B. D. Ripley, *Modern Applied Statistics with S,* New York: Springer, 2002.

[99] R. Wehrens, *Chemometrics with R: multivariate data analysis in the natural sciences and life sciences,* New York, NY: Springer, 2011.

[100] Z. Díaz, M. J. Segovia, J. Fernández , and E. M. d. Pozo, “Machine learning and statistical techniques. An application to the prediction of insolvency in spanish non-life insurance companies,” *The International Journal of Digital Accounting Research,* vol. 5, no. 6, pp. 1-45, 2005.

[101] Z. Hu, E. S. Snitkin, and C. DeLisi, “VisANT: an integrative framework for networks in systems biology,” *Brief Bioinform,* vol. 9, no. 4, pp. 317-25, Jul, 2008.

[102] B. H. Junker, C. Klukas, and F. Schreiber, “VANTED: a system for advanced data analysis and visualization in the context of biological networks,” *BMC Bioinformatics,* vol. 7, pp. 109, 2006.

[103] www. "SemanticSBML: a tool for annotating, checking, and merging of biochemical models in SBML format," http://sysbio.molgen.mpg.de/semanticsbml/index.html]

[104] B. J. Bornstein, S. M. Keating, A. Jouraku, and M. Hucka, “LibSBML: an API library for SBML,” *Bioinformatics*, vol. 24, no. 6, pp. 880-1, Mar 15, 2008.

[105] K. Raman, and N. R. Chandra, “Merging and Visualisation of SBML Models,” in The 9th SBML Forum Meeting, Heidelberg, Germany, 2004.

[106] A. Drager, N. Hassis, J. Supper, A. Schroder, and A. Zell, “SBMLsqueezer: a CellDesigner plug-in to generate kinetic rate equations for biochemical networks,” *BMC Syst Biol*, vol. 2, pp. 39, 2008.

[107] N. Le Novere, A. Finney, M. Hucka, U. S. Bhalla, F. Campagne, J. Collado-Vides, E. J. Crampin, M. Halstead, E. Klipp, P. Mendes, P. Nielsen, H. Sauro, B. Shapiro, J. L. Snoep, H. D. Spence, and B. L. Wanner, “Minimum information requested in the annotation of biochemical models (MIRIAM),” *Nat Biotechnol*, vol. 23, no. 12, pp. 1509-15, Dec, 2005.

[108] www. "SBO:Systems Biology Ontology," http://www.ebi.ac.uk/sbo.

[109] T. Sing, O. Sander, N. Beerenwinkel, and T. Lengauer, “ROCR: visualizing classifier performance in R,” *Bioinformatics*, vol. 21, no. 20, pp. 3940-1, Oct 15, 2005.

[110] Z. He, and W. Yu, “Stable feature selection for biomarker discovery,” *Comput Biol Chem*, vol. 34, no. 4, pp. 215-25, Aug, 2010.

[111] L. Breiman, "Manual on setting up, using, and understanding random forests v3.1," http://oz.berkeley.edu/users/breiman/Using_random_forests_V3.1.pdf, 2002].

[112] S. Waaijenborg, and A. H. Zwinderman, “Sparse canonical correlation analysis for identifying, connecting and completing gene-expression networks,” *BMC bioinformatics*, vol. 10, pp. 315, 2009.

[113] D. M. Witten, and R. J. Tibshirani, “Extensions of sparse canonical correlation analysis with applications to genomic data,” *Statistical applications in genetics and molecular biology*, vol. 8, no. 1, pp. Article28, 2009.

PUBLICATIONS

Journals

GOrevenge: A novel generic reverse engineering method for the identification of critical molecular players, through the use of ontologies. K.Moutselos, I. Maglogiannis, A.Chatziioannou. IEEE Trans Biomed Eng. 2011 (TBME) Dec;58(12):3522-7. Epub 2011 Aug 15.

KEGGconverter: a tool for the in-silico modelling of metabolic networks of the KEGG Pathways database. K.Moutselos, I.Kanaris, A.Chatziioannou, I.Maglogiannis, FN.Kolisis. BMC Bioinformatics, vol. 10, pp.324,2009.

Data Integration and Feature Selection for Robust Composite Biomarker Discovery from High-volume Molecular and Imaging Data. K.Moutselos, I. Maglogiannis, A.Chatziioannou. Submitted to IEEE Transactions on Computational Biology and Bioinformatics (TCBB).

Conferences

Heterogeneous data fusion and selection in high-volume molecular and imaging datasets. K.Moutselos, I. Maglogiannis, A.Chatziioannou. IEEE BIBE 2012: 407-412.

Feature Selection Study on Separate Multi-modal Datasets: Application on Cutaneous Melanoma. K.Moutselos, A.Chatziioannou, I. Maglogiannis. AIAI (2) 2012: 36-45.

Delineation and interpretation of gene networks towards their effect in cellular physiology- a reverse engineering approach for the identification of critical molecular players, through the use of ontologies. K.Moutselos, I. Maglogiannis, A.Chatziioannou. Conf Proc IEEE Eng Med Biol Soc. 2010;2010:6709-12.

Building in-silico pathway SBML models from heterogeneous sources. I.Kanaris, K.Moutselos, A.Chatziioannou, I.Maglogiannis, F.Kolisis. IEEE BIBE 2008: 1-6.