TÉCNICO LISBOA

Development of the PathoYeastract database: aiming the study of transcriptional regulation in pathogenic yeasts

SAUVAGYA MANNA

Thesis to obtain the Master of Science Degree in

Biotechnology

Supervisors: Prof. Miguel Nobre Parreira Cacho Teixeira
Dr. Pedro Tiago Gonçalves Monteiro

Examination Committee

Chairperson: Prof. Isabel Maria de Sá Correia Leite de Almeida
Supervisor: Dr. Pedro Tiago Gonçalves Monteiro
Member of the Committee: Dr. Susana de Almeida Mendes Vinga Martins

July 2016

Acknowledgement

I would first like to thank my thesis advisors Prof. Miguel Nobre Parreira Cacho Teixeira of the Departamento de Bioengenharia (DBE) at Instituto Superior Técnico and Dr. Pedro Tiago Gonçalves Monteiro of the Departamento de Engenharia Informática (DEI) at Instituto Superior Técnico. The doors to Prof. Teixeira and Dr. Monteiro's offices were always open whenever I ran into a trouble spot or had a question about my research or writing. They consistently allowed this paper to be my own work, but steered me in the right direction whenever they thought I needed it.

I would also like to thank Prof. Erik Aurell of the Dept. of Computational Biology, KTH Royal Institute of Technology, Director of the Erasmus Mundus Master's Program in Systems Biology - euSYSBIO and Prof. Isabel Maria de Sá Correia Leite de Almeida at the Departamento de Bioengenharia (DBE) at Instituto Superior Técnico, the Chairperson of the dissertation committee, who had monitored my progress with the research work from the very beginning. Without their passionate participation and input, the research works could not have been successfully conducted.

I would like to thank the Erasmus Mundus euSYSBIO consortium for funding my research work carried out for the dissertation. Without their help it would not have been possible to conduct the work at ease.

I would also like to acknowledge Dr. Susana de Almeida Mendes Vinga Martins of IDMEC, Instituto Superior Técnico, Universidade de Lisboa as the member of the committee of this thesis, and I am gratefully indebted to her for her very valuable comments on this thesis.

Finally, I must express my very profound gratitude to my parents, my grandmother and Segolene Meeüs for providing me with unfailing support and continuous encouragement throughout my years of study and through the process of researching and writing this thesis. This accomplishment would not have been possible without them.

Thanking you,

Sauvagya Manna

Abstract

The PATHOgenic YEAst Search for Transcriptional Regulators And Consensus Tracking (PathoYeastract – http://pathoyeastract.org) database is a tool for the analysis and prediction of transcription regulatory associations at the gene and genomic levels in pathogenic yeasts. In the currently available version it enlists information focused on the two most prevalent of pathogenic yeasts of the *Candida* genus: *C. albicans* and *C. glabrata*.

This MSc thesis is a contribution to the construction of PathoYeastract. It includes the development of scripts for the automatic retrieval of data for each *C. albicans* and *C. glabrata* genes encountered in the Candida Genome Database, including coding and promoter sequences, description, associated Gene Ontology terms and orthologs in *Saccharomyces cerevisiae* and in other *Candida* species.

Given the overall lack of experimental data, the newly constructed PathoYeastract database has been developed to predict regulatory associations in 2 *Candida* species, based on the known transcription regulation of orthologous transcription factors and target genes in *Saccharomyces cerevisiae*, a model Yeast organism which has been widely studied for the last few decades. With the aid of this tool, a comprehensive comparison is expected to bring light into the mechanisms underlying the evolution of transcription regulatory networks among related yeasts.

The incorporation of regulatory data on other closely related *Candida* species to widen the scope of this database to the study of inter-species regulatory network is envisaged for the near future with expected impact in the understanding of the development of pathogenesis and antifungal drug resistance.

Keywords: *Candida glabrata, Candida albicans,* Regulatory associations, Evolution, PathoYeastract, Transcription regulation prediction, Yeast, Transcription regulatory network.

Resumo

A base de dados PathoYeastract (PATHOgenic YEAst Search for Transcriptional Regulators And Consensus Tracking – www.pathoyeastract.org) é uma ferramenta para a análise e previsão de associações de regulação transcricional, ao nível do gene e do genoma, em leveduras patogénicas. Na sua versão actual, inclui informação focada nas duas espécies mais prevalentes de leveduras patogénicas do género *Candida*: *C. albicans* e *C. glabrata*.

Esta tese de mestrado é uma contribuição para a construção da PathoYeastract. Inclui o desenvolvimento de *scripts* para a recuperação automática de dados para cada gene de *C. albicans* e *C. glabrata*, conforme encontrados na base de dados *Candida Genome Database*, incluindo sequências codificante e promotora, descrição, termos da *Gene Ontology* a ele associados e ortólogos em *Saccharomyces cerevisiae* e noutras espécies de *Candida*.

Dada a falta de dados experimentais, a base de dados recém-construída PathoYeastract foi desenvolvida para prever associações de regulação em duas espécies de *Candida*, com base na regulação transcricional de factores de transcrição e genes alvo ortólogos em *Saccharomyces cerevisiae*, uma levedura modelo que foi amplamente estudada nas últimas décadas. Com a ajuda desta ferramenta, é expectável que a comparação entre espécies possa trazer luz aos mecanismos subjacentes à evolução das redes de regulação transcricional entre leveduras aparentadas.

A incorporação de dados de regulação relativas a outras espécies próximas do género *Candida*, que alargará o espectro de acção desta base de dados ao estudo de redes de regulação em espécies diferentes, está prevista para o futuro próximo, com impacto espectável na compreensão do desenvolvimento da patogénese de da resistência a fármacos antifúngicos.

Keywords: *Candida glabrata, Candida albicans,* Associações regulatórias, Evolução, PathoYeastract, Previsão de regulação transcricional, Levedura, Redes de regulação transcricional.

Table of Contents

List of Figures

List of Tables

1. Introduction

1.1 Motivation

The total number of eukaryotic species on earth has been estimated nearly about 8.7 million (as of 2013), amongst which 7% (611,000 species) are fungi and only 600 of them had been identified as human pathogens till date [1]. Besides posing severe infections in human hosts, Candida species are the fourth most common cause of life-threatening systemic human infections, with mortality rates as high as 50% according to the United States hospital records of patients in the United States of America [2]. *Candida* infections in humans can be of two different types: 1) superficial, oral or vaginal candidiasis or 2) systemic infections which poses severe threat and prolonged effects. Colonization of *Candida sp.* in the oral cavity of 75% of the entire human population has been recorded [3]. Though it remains in a commensal state in healthy individuals, there is a fair chance of oral candidiasis to be developed even if the individual is mildly immunocompromised which affects the person's oropharynx and/or esophagus. Persons suffering from dysfunction of adaptive immune system e.g. HIV or cancer are at high risk to be exposed to *Candida* pathogenesis. Another common infection caused by *Candida sp.* in females is vulvovaginal candidiasis; 75% of women suffer from it at least once in their lifetime. Despite their abundancy, superficial *Candida* infections are non-lethal. In contrast, systemic infections caused by *Candida sp.* can be lethal, even after first line antifungal therapy. Damage of the gastrointestinal mucosa and neutropenia are the risk factors for developing experimental systemic candidiasis. Further risk factors include central venous catheters, which allow direct access of the fungus to the bloodstream, the application of broad-spectrum antibacterial, which enable fungal overgrowth, and trauma or gastrointestinal surgery, which disrupts mucosal barriers. In both cases the pathogenicity of *Candida* is triggered by several virulence factors. In a study done by Klempp-Selb, Rimek and Kappe [4] has shown that the two specific species *Candida glabrata* and *Candida albicans* resemble each other at a genomic level resulting in similar pattern of genetic regulation and drug resistance which makes the study of the gene regulatory network of the two species together.

Several virulence factors include morphological transition between yeast and hyphal form (polymormphism), in the case of *Candida albicans*, expression of adhesins and

invasins on the cell surface, thigmotropism, formation of biofilms, phenotypic switching and the secretion of hydrolytic enzymes, and fitness attributes such as adaptation to different pH helps the colonization of *Candida sp.* in a wide range of host niches.

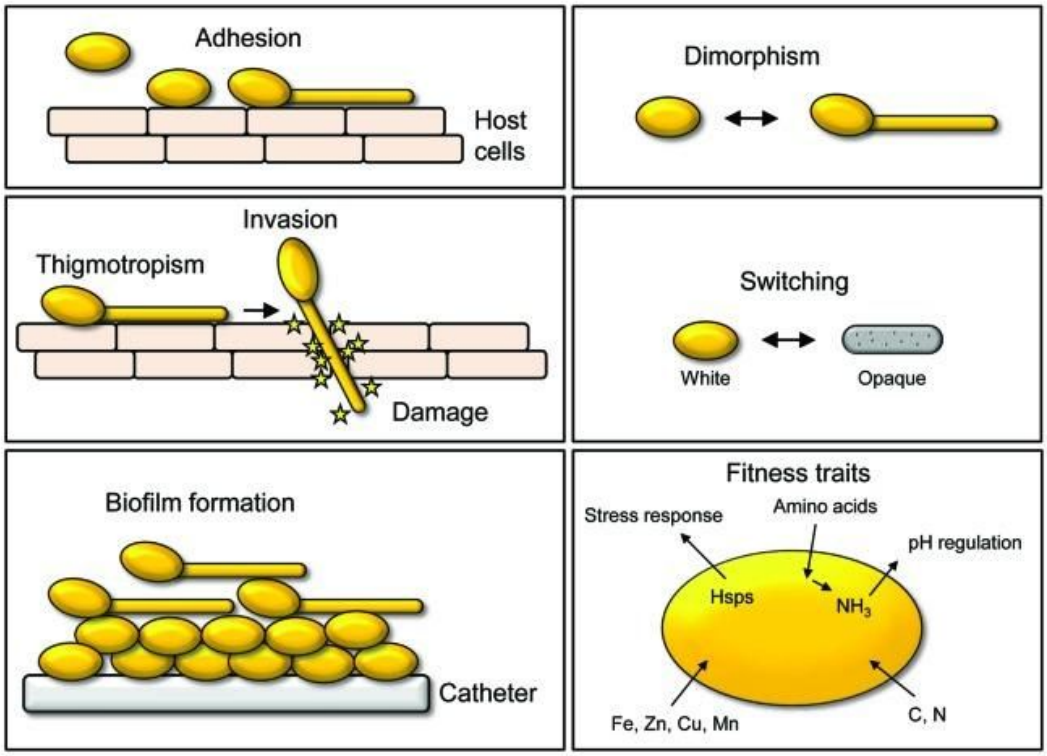

Figure 1: Different virulence factors displayed by *Candida albicans*

Some *Candida* species have the ability to undergo a morphogenetic switch from budding yeast to hyphal growth in response to stimulus and growth conditions. Recent virulence studies of filamentation regulatory mutants argue that both yeast and filamentous forms have roles in infection. The pathogenicity of *Candida* is linked to its capacity to switch among different growth forms [5]. Cph2 regulates hyphal development in *C. albicans*, as cph2/cph2 mutant strains show medium-specific impairment in hyphal development and in the induction of hypha-specific genes. However, many hyphae-specific genes do not have potential Cph2 binding sites in their upstream regions. Interestingly, upstream sequences of all known hyphae-specific genes are found to contain potential binding sites for Tec1, a regulator of hyphal development.

Multi-drug resistance (MDR), a phenomenon of acquiring non-susceptibility to a wide range of structurally and functionally distinct cytotoxic compounds, is ubiquitously presented in living organisms, from bacteria to mammals [6]. The emergence of MDR is becoming a challenging problem in the treatment of infectious diseases, food preservation and in crop protection [7]. The ABC transporter genes *CDR1* and *CDR2* are typically upregulated in *Candida albicans* when developing resistance to azoles or can be upregulated by exposing cells transiently to drugs such as fluphenazine [8]. The upregulation of *CDR1* and *CDR2* by fluphenazine in *C. albicans* is controlled by the TF Tac1, which is a major contributor to azole-resistance. Another transcription factor involved in this process in *C. albicans* is Mrr1 which controls the expression of the Mdr1 efflux pump and mediates multidrug resistance in *C. albicans* [9, 10]. In another study carried out at IST, Lisbon and University of Lisbon in recent days which has proved the Clotrimazole Drug Resistance in *Candida glabrata* Clinical Isolates Correlates with Increased Expression of the Drug:H^+ Antiporters CgAqr1, CgTpo1_1, CgTpo3, and CgQdr2 [11].

1.2 Approach

Given the importance of transcriptional control in pathogenesis-related phenotypes, as in all cellular processes, it is vital to understand the global mechanisms of transcriptional control in Candida species. With that aim, this work is focused on the development of the PATHOYEASTRACT (Pathogenic Yeast Search for Transcriptional Regulators And Consensus Tracking) database. Its goal is to: 1) make a publicly available online resource for Transcription Regulation Factors and target genes and the relation between TFs and DNA binding sites in *Candida* species, 2) to predict the gene regulatory network for *Candida* species, based on the data available for the model yeast *Saccharomyces cerevisiae* and 3) to study gene regulatory network evolution through the development of a cross-species comparison tool.

The new database was built using a similar architecture to that of the YEASTRACT database [12, 13, 14, 15]. Its structure was organized considering the basic functioning of transcription factors. TFs are proteins which are involved in the process of transcribing DNA into RNA (1). The presence of DNA-binding domains in TFs enables them to bind to specific sequences of DNA called promoter sequences (2). Some TFs bind to the DNA promoter sequences near the transcription initiation sites to form the transcription

initiation complex while other TFs can bind to regulatory sequences e.g. enhancer sequences to either stimulate or repress the regulation and expression of the related gene. Regulatory sequences can be hundreds of base pairs upstream from the transcription site. Once they are attached to the specific DNA, they trigger or suppress transcription, making genes up or downregulated.

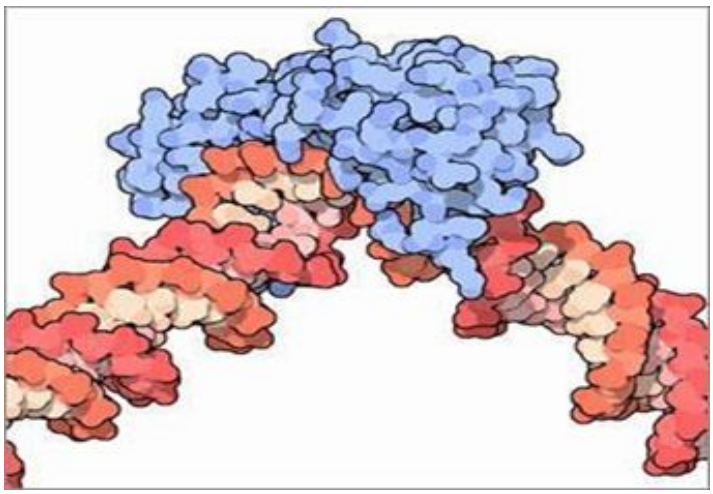

Figure 2: A transcription factor (blue) binds to DNA (red) and alters which genes are turned on, or expressed. (*Image courtesy of David S. Goodsell, RCSB Protein Data Bank*)

The YEASTRACT database had been introduced in 2006 for the first time. Since then it had been highly praised among the researchers in the field with around 170,000 queries (2011-2014) and 8500 scientists accessing around the world. There had been a huge development and expansion on the available data since the beginning with a 425% more transcription regulation data. It presently contains 206,299 (as of 2014) regulatory associations between genes and TFs. YEASTRACT contains 41.693 regulatory associations based on DNA binding evidence and 172.814 on expression evidence, with some overlap [12]. Other databases reporting information on the transcriptional control in *Saccharomyces* species include MYBS [16], TRANSFAC [17], RSAT [18], YPA [19] and YeTFaSCo [20]. However, their focus is mainly on the prediction and analysis of the promoter regions.

After this achievement with YEASTRACT, building PathoYeastract database was a challenge on its own. Besides providing the tools for promoter analysis in *Candida,* PathoYeastract will also provide a complete integration of all experimentally validated transcription regulation data ever published for *Candida.* Given that it is focused in multiple-species it will further enable the cross comparison between transcriptional networks from different species, providing interesting clues on the evolution of those

networks. So far the regulation data from the 2 most prevalent of *Candida* species; *Candida glabrata and Candida albicans* has been updated to the database.

Overall count		GO Ontology			Documented Regulations			Orthologs
Species	Total no. of ORF	No. of GO terms (Process)	No. of GO terms (Function)	No. of GO terms (Component)	No. of unique pairs	No. of unique TFs	No. of unique TGs	To *S. cerevisiae*
C. glabrata	5601	2984	1677	734	1818	34	1313	4509
C. albicans	6313	3053	1812	740	27223	105	5598	4012

Table 1: Documented regulation for *Candida glabrata* and *Candida albicans* for unique Transcription Factors against each other and *Saccharomyces cerevisiae*

Comparative transcriptomics has been used as a state of the art technology for the past few years to study gene regulation, enabling the comparison of transcriptome-wide responses to the same environmental cues in yeast species of the same family. *Saccharomyces cerevisiae* has been used as a model organism in that sense. As such, it became a key aspect of this work to develop a tool to use the huge amount of information gathered for *S. cerevisiae* to predict transcription regulation in the far less well studied *Candida* species.

2. Materials and Methods

2.1 Data Extraction

The information residing in different sources was retrieved from Gene Ontology [24,25] (http://www.geneontology.org/), Candida Genome Database [21] (http://candidagenome.org) and Saccharomyces Genome Database [22] (http://yeastgenome.org), post-processed and integrated in the PathoYeastract database.

For every ORF/gene existing in the database, information related to *C.glabrata* and *C.albicans*, regarding standard names, alias, gene sequence, promoter sequence, gene ontology annotations, etc, was retrieved from Candida Genome Database [21] (http://candidagenome.org), post-processed and integrated into the PathoYeastract database. Additionally, equivalent information related to *S.cerevisiae* was retrieved from Saccharomyces Genome Database [22] (http://yeastgenome.org) and integrated into the PathoYeastract database. All these data are the anchor points on which additional data from additional sources are mapped.

Information regarding the gene ontology terms and their hierarchy was retrieved from the Gene Ontology Consortium [24,25] (http://geneontology.org). Information regarding orthologs genes was retrieved from the Candida Genome Database between the following species:

- *Candida glabrata -> Saccharomyces cerevisiae*
- *Candida glabrata <-> Candida albicans*
- *Candida albicans -> Saccharomyces cerevisiae*

Finally, information regarding regulatory associations was collected by literature review and curated, by members of the Biological Sciences Research Group (BSRG) at Instituto Superior Técnico. This information was supplied in the form of an Excel file, containing for each regulatory association, the set of supporting references with the corresponding PubmedID, as well as the additional annotations (like environmental conditions, association type, etc).

To complete and correct information regarding references, a script was written to, given a PubmedID, obtain all information regarding a publication from Pubmed.

2.2 Data Structure

The entity–relationship model (ER model) describes how the relation between the different data entities, allowing the information representing the real world to be implemented in a database. This section describes the identification of the main entities (Orf, Protein, Orthologs, Regulation, Consensus, Gene Ontology) and the mapping between them for the definition of the structure of the integrated database.

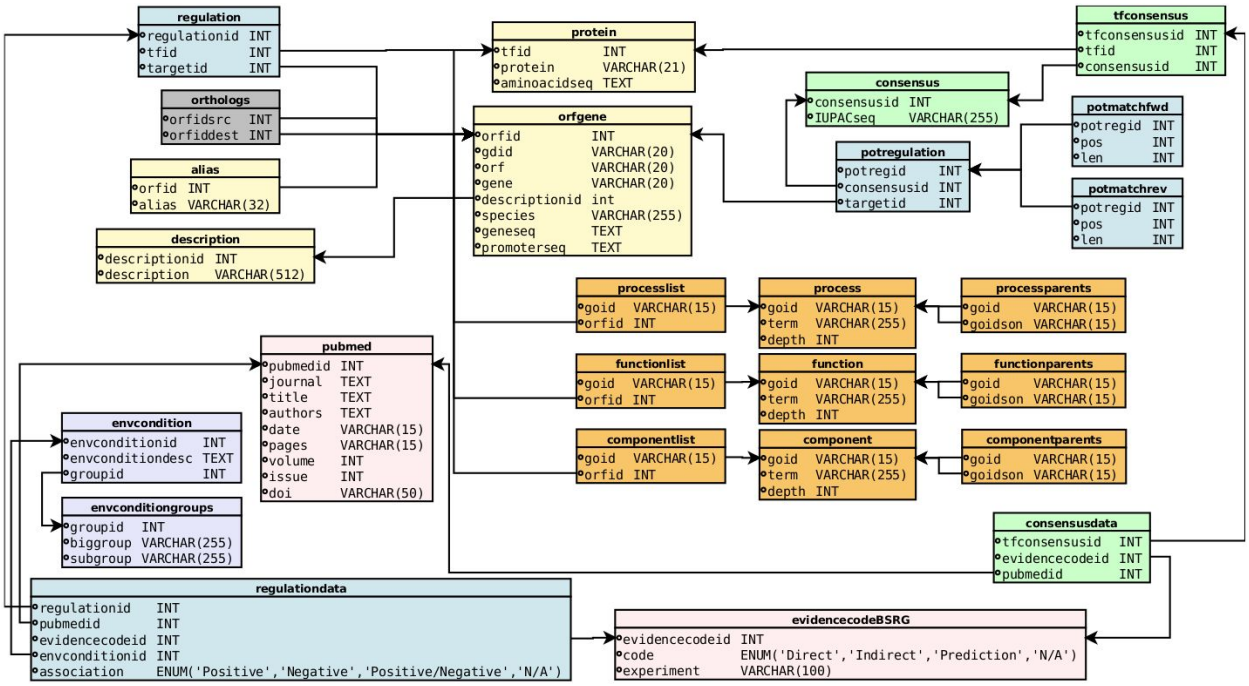

Figure 3: Physical model of the database

2.2.1 Concept of ORF

In molecular biology, an ORF orfresents an open reading frame which can potentially be translated into a protein or peptide or RNA. It is a continuous sequence of nucleotides beginning with a start codon and ending with a stop codon (Brown, 2010). Each gene has a corresponding ORF, but the reverse is not necessarily true, which means that the ORF is considered to be the unique identifier in the orfgene table.

Attributes:

For each ORF, an unique (internal to the database) identifier *orfid* is generated in order to relate a given ORF to the other PathoYeastract tables. Also, since all the data for this table

is retrieved from CGD and SGD, their corresponding internal IDs are also stored in the attribute *gdid*. The gene name is kept in the *gene* attribute. A description of the gene function is stored in the *description* table and the corresponding unique identifier *descriptionid* is associated to a given ORF. Also, the associated species is stored in the *species* attribute. Finally, the gene sequence and the promoter sequence, retrieved from the CGD and SGD, are associated to an ORF and kept in the *geneseq* and *promoterseq* fields, respectively.

Relations:

The ORF concept has a relation with the *protein* table, containing the corresponding protein name and sequence. In case of multiple gene names, all the alias are kept in the *alias* table identified by the *orfid* field. Also, an ORF has a relation with the three (function, process and component) gene ontologies where each ORF is mapped to the corresponding GO terms in tables *functionlist*, *processlist* and *componentlist*. It also relates to the *orthologs* table where an ORF from a given species is mapped to an homologous ORF in another species. Finally, since each ORF may be a target gene in a regulation, the *orfid* field is mapped into the *targetid* field in the *regulation* table.

2.2.2 Concept of Protein

Proteins are large biological molecules, composed of one or more chains of amino acids, and they are the fundamental components and functional units of the cell.

Attributes:

The Protein concept has an unique (internal to the database) identifier *tfid* which is mapped to the corresponding ORF on the attribute *orfid*. Also, it has the corresponding protein name, *proteinname* field, and the protein aminoacid sequence, *aminoacidseq* field.

Relations:

The Protein concept is related to the *orfgene* table, with the corresponding *orfid.* Also, since a protein may be involved in a regulatory association it has a relation with the *regulation* table, identified by the *tfid* field. Additionally, for the transcription factors with associated binding sites (*consensus* in the database), an association is created in the *tfconsensus* table.

2.2.3 Concept of Orthologs

Orthologs genes are the genes which belong to different species but share the same ancestral root and have the same function in species. They are an important component

to derive the functional relation between cross-species proteins.

Attributes:

In this system, the *orfid* field (internal to the database) is central to an ORF and is where the orthology mappings are done. Here, the *orthologs* table has two fields, *orfidsrc* and *orfiddest*, representing the ORF of source species being considered an ortholog of another ORF of a destination species. The *orthologs* table contains all the cross-species relations for all ORFs.

2.2.4 Concept of Regulation

For each transcription regulation, transcription factor - target gene pair, there is a unique regulation ID assigned, *regulationid* field, where all references related to that specific regulation are mapped.

Attributes:

The *regulation* table has three attributes: *regulationid*, *tfid* and *targetid*. The transcription factor ID (*tfid*) relates to an existing protein in the *protein* table, and the *targetid* relates to an existing ORF in the *orfgene* table. For each pair a unique regulation ID is created, the *regulationid* field.

Relations:

The *regulationid* field, is used to identify a particular regulation and to map the corresponding bibliographic references in the *regulationdata* table. This table contains additional information related with the bibliographic referece, like the environmental conditions, association type, among others. Finally, it contains the pubmedID for further consultation.

2.2.5 Concept of Consensus

Each protein can be associated to several binding sites, representing DNA binding sequences to the promoter of target genes. Each of these DNA binding sequences are here denominated *consensus* are represented in the compact IUPAC format [26].

Atributes:

A DNA binding sequences is represented in the *consensus* table in the *IUPACseq* field, and for each unique sequence there is an associated *consensusid* atribute. This attribute is then used to relate a given sequence with other concepts.

Relations:

A consensus is associated with at least one protein, through the table *tfconsensus*, where a protein *tfid* is associated with a DNA binding sequence *consensusid*. To each of these pairs, a *tfconsensusid* key is created to map all the bibliographic references related to it. The mapping of the bibliographic references is made in the *consensusdata* table.

2.2.5 Concept of Gene Ontology

After genome sequencing, there are much information needed to be interpreted. It is necessary to assign terms to represent the biological process and functions in which the objective proteins and genes are involved in a synthetic and standard form. Initially, there was no universal standard form for these terms, which means that the usage of terms may be different according to research areas or even research groups. This leads to increasing difficulties in communication and sharing data.

In order to overcome this problem, the Gene Ontology (GO) consortium [24,25] was founded in 1998 by three model organism databases, FlyBase (Drosophila) [27], the Saccharomyces Genome Database (SGD) [22] and the Mouse Genome Database (MGD) [28], aiming at defining consistent Ontology terms of gene product properties. Candida Genome Database (CGD) [23] also follows the convention of GO convention which had been followed in the PathoYeastract database. The GO project has developed three ontologies describing gene products in three different domains:

- Cellular component - a part of a cell or its extracellular environment;
- Molecular function - the “abilities” that the gene product has;
- Biological process - a set of molecular events with a defined beginning and end.

Each GO term has a term name, a unique alphanumeric identifier (*goid*), a definition with cited sources, a namespace indicating the domain which it belongs to, and several other elements describing its characteristics. These three ontologies are organized hierarchically through *part of* (composition) and *is_a* (inheritance) relations. The “son” terms are more specialized than their “parent” terms, but unlike a strict hierarchy, a GO term may have multiple parent terms. There is a root, the most general term and corresponding to the

top of the hierarchy, for each of the three ontologies cellular_component (GO:0005575), molecular_function (GO:0003674), biological_process (GO:0008150). All other terms in a domain can trace their parentage to the corresponding root term.

Attributes:

Each ontology as a table with mapping the *goid* field with the term *description*. Additionally, since the concepts are hierarchically organized, there is an attribute to indicate the distance between the objective GO term and the corresponding root term, called *depth*. The *parent* table relates a given GO term with its hierarchical parents, *goidson* and *goid* fields, respectively.

Relations:

Each of the three ontologies has a *list* table, relating a given GO term with a given ORF, through the mapping of the *goid* and *orfid* fields.

2.3 Database Architecture

The information system is developed in PHP and is supported by an Apache web server with a PHP module included, communicating with a MySQL database management system (DBMS).

In this section, we will describe the physical database model of the information system (Figure 3). The concepts in the conceptual model discussed in the previous section are represented as tables in the database, and the corresponding attributes represented as columns of those tables. Some auxiliary tables are used to represent the relations between different concepts (see Figure ??).

Each table may have some constraints where one or more fields are put together to create a primary key or unique key, meaning that entries in that table with repeated values on those fields are not allowed.

2.4 Functionalities

2.4.1 Search for TFs

This functionality permits the user to find all regulators of a given set of target genes. It also permits to filter the results based on the annotations associated with the regulatory associations, e.g. environmental conditions or association type.

```
function findRegulators($sOrfGenes, $regs, $ecgroup, $eviCode, $assocType) {
        list($aORFGenes, $aUnknown) = $this->orf2gene($sOrfGenes);

        $orflist = "''";
        foreach (array_keys($aORFGenes) as $orf) {
                $orflist .= ", '$orf'";
        }

        $docReg = array();
        if ($regs['doc'] || $regs['both']) {
                $q  = "select R.tfid,P.protein,R.targetid,O.orf,O.gene ";
                $q .= "from regulation as R, orfgene as O, protein as P where ";
                $q .= "O.species='$this->_species' and ";
                $q .= "P.tfid=R.tfid and O.orfid=R.targetid and O.orf in (";
                $q .= $orflist.") ";
                $q .= $this->getMysqlEviCode($eviCode, $assocType);
                $q .= $this->getMysqlEnvCond($ecgroup);
                if ($this->_dbAccess->query("docreg", $q)) {
                        while ($row = $this->_dbAccess->nextObject("docreg")) {
                            $docReg[$row['orf']][$row['protein']] = $row['protein'];
                        }
                }
                $this->_dbAccess->freeResult("docreg");
        }

        $potReg = array();
        if ($regs['pot'] || $regs['both']) {
                $q  = "select O.orf,O.gene,PR.potregid, C.IUPACseq, P.protein from ";
                $q .= "orfgene as O, potregulation as PR, consensus as C, tfconsensus as ";
                $q .= "TFC, protein as P where O.orf in ($orflist) and O.species='";
                $q .= $this->_species."' and PR.targetid=O.orfid and PR.consensusid=C.consensusid ";
                $q .= "and TFC.consensusid=C.consensusid and P.tfid=TFC.tfid";
                if ($this->_dbAccess->query("potreg", $q)) {
                        while ($row = $this->_dbAccess->nextObject("potreg")) {
                                $potReg[$row['orf']][$row['protein']][$row['potregid']] = $row['IUPACseq'];
                        }
                }
                $this->_dbAccess->freeResult("potreg");
        }

        return array($aORFGenes, $aUnknown, $docReg, $potReg);
}
```

Figure *5*: Script to search for TFs given a set of target genes

Additionally, the user can find for all regulators of a given set of target genes in a particular homologous species. This functionality searches for each target gene the homologous genes in the target species, then obtains the homologous regulators, and then maps back those homologous regulators to the potential regulators in the original species.

```
function findHomoRegulators($sOrfGenes, $toSpecies) {
  list($aORFGenes, $aUnknown) = $this->orf2gene($sOrfGenes);
  $homoORFs = $this->getHomoORFInfo($aORFGenes, $this->_species, $toSpecies);

  $orfidlist = "''";
  foreach (array_keys($homoORFs) as $orf) {
    $orfidlist .= ", " . $homoORFs[$orf]['homo']['orfid'];
  }

  $docReg = array();
  $docRegTo = array();
  $homoTFs = array();
  $q  = "select R.targetid, R.tfid,O.orf,O.gene,O.species ";
  $q .= "from regulation as R, orfgene as O where R.targetid in (";
  $q .= $orfidlist . ") and R.tfid=O.orfid";
  if ($this->_dbAccess->query("docreg", $q)) {
    while ($row = $this->_dbAccess->nextObject("docreg")) {
      $docRegTo[$row['targetid']][$row['orf']] = $row['gene'];
      $homoTFs[$row['orf']] = $row['gene'];
    }
  }
  $this->_dbAccess->freeResult("docreg");

  // Mapping back to the original species
  $homoRevORFs = $this->getHomoORFInfo($homoTFs, $toSpecies, $this->_species);

  // Get ORF names from the IDs
  $orfidlist = "''";
  foreach (array_keys($homoRevORFs) as $homoORF) {
    $orfidlist .= ", " . $homoRevORFs[$homoORF]['homo']['orfid'];
  }
  $id2ORF = array();
  $q  = "select O.orfid,O.orf,O.gene,P.protein from orfgene as O, ";
  $q .= "protein as P where O.species='{$this->_species}' and ";
  $q .= "P.tfid=O.orfid and orfid in ($orfidlist)";
  if ($this->_dbAccess->query("id2orf", $q)) {
    while ($row = $this->_dbAccess->nextObject("id2orf")) {
      $id2ORF[$row['orfid']]['orf']     = $row['orf'];
      $id2ORF[$row['orfid']]['gene']    = $row['gene'];
      $id2ORF[$row['orfid']]['protein'] = $row['protein'];
    }
  }
  $this->_dbAccess->freeResult("id2orf");

  $mapping = array();
  // Put the pieces together: Documented Homologous Regulations
  foreach (array_keys($homoORFs) as $orf) {
    $homoTGid = $homoORFs[$orf]['homo']['orfid'];
    $mapping[$orf]['orf']  = $homoORFs[$orf]['homo']['orf'];
    $mapping[$orf]['gene'] = $homoORFs[$orf]['homo']['gene'];
    if (isset($docRegTo[$homoTGid])) {
      foreach ($docRegTo[$homoTGid] as $homoTForf => $homoTFgene) {
        if (!isset($homoRevORFs[$homoTForf])) continue;
        $tfid = $homoRevORFs[$homoTForf]['homo']['orfid'];
        $prot = $id2ORF[$tfid]['protein'];
        $docReg[$orf][$prot] = $homoRevORFs[$homoTForf]['gene'];
      }
    }
  }
  return array($aORFGenes, $aUnknown, $docReg, $mapping);
}
```

Figure 6: Script to search for homologous TFs given a set of target genes

2.4.2 Search for target genes

Analogously to the search by TF functionality, the user can search for target genes given a set of transcription factors. It also permits to filter the results based on the annotations associated with the regulatory associations, e.g. environmental conditions or association type.

```
function findRegulated($sOrfTFs, $regs, $ecgroup, $eviCode, $assocType) {
        list($aOrfTFs, $unknown) = $this->orfgene2prot($sOrfTFs);

        $tflist = "''";
        foreach ($aOrfTFs as $p) {
                $tflist .= ", '".addslashes($p)."'";
        }

        $docReg = array();
        if ($regs['doc'] || $regs['both']) {
                $q  = "select R.tfid,P.protein,R.targetid,O.orf,O.gene from ";
                $q .= "regulation as R, orfgene as O, protein as P where ";
                $q .= "P.tfid=R.tfid and O.orfid=R.targetid and ";
                $q .= "P.protein in ($tflist) and O.species='{$this->_species}'";
                $q .= $this->getMysqlEviCode($eviCode, $assocType);
                $q .= $this->getMysqlEnvCond($ecgroup);
                if ($this->_dbAccess->query("docreg", $q)) {
                        while ($row = $this->_dbAccess->nextObject("docreg")) {
                                $docReg[$row['protein']][$row['orf']] = $row['gene'];
                        }
                }
                $this->_dbAccess->freeResult("docreg");
        }

        $potReg = array();
        if ($regs['pot'] || $regs['both']) {
                $q  = "select C.protein, C.consensus, PR.orf, PR.ID ";
                $q .= "from consensus as C, potentialregulation as PR where ";
                $q .= "C.proteinname in ($tflist) and C.consensus = PR.consensus";

                $q  = "select O.orf,PR.potregid, C.IUPACseq, P.protein from ";
                $q .= "orfgene as O, potregulation as PR, consensus as C, tfconsensus as ";
                $q .= "TFC, protein as P where P.protein in ($tflist) and O.species='";
                $q .= $this->_species."' and PR.targetid=O.orfid and PR.consensusid=C.consensusid ";
                $q .= "and TFC.consensusid=C.consensusid and P.tfid=TFC.tfid";
                if ($this->_dbAccess->query("potreg", $q)) {
                        while ($row = $this->_dbAccess->nextObject("potreg")) {
                                $potReg[$row['protein']][$row['IUPACseq']][$row['orf']] = $row['potregid'];
                        }
                }
                $this->_dbAccess->freeResult("potreg");
        }

        return array($aOrfTFs, $unknown, $docReg, $potReg);
}
```

Figure 7: Script to search for target genes given a set of transcription factors

Also, as in the search for transcription factors, the search for target genes can be performed considering homologous species. Here, the user inputs a set of transcription factors which are mapped to the homologous transcription factors in the target species, then the regulatory associations are searched in that species, and the set of homologous target genes found are mapped back into the original species as a result to the user.

```
function findHomoRegulated($sOrfTFs, $toSpecies) {
        list($aOrfTFs, $unknown) = $this->orfgene2prot($sOrfTFs);
        $homoORFInfo = $this->getHomoORFInfo($aOrfTFs, $this->_species, $toSpecies);

        $tfidlist = "''";
        foreach (array_keys($homoORFInfo) as $tf) {
                $tfidlist .= ", " . $homoORFInfo[$tf]['homo']['orfid'];
        }

        $docReg = array();
        $docRegTo = array();
        $homoORFs = array();
        $q  = "select R.tfid, R.targetid,O.orf,O.gene,O.species ";
        $q .= "from regulation as R, orfgene as O where R.tfid in (";
        $q .= $tfidlist . ") and R.targetid=O.orfid";
        if ($this->_dbAccess->query("docreg", $q)) {
                while ($row = $this->_dbAccess->nextObject("docreg")) {
                        $docRegTo[$row['tfid']][$row['orf']] = $row['gene'];
                        $homoORFs[$row['orf']] = $row['gene']; // TGs
                }
        }
        $this->_dbAccess->freeResult("docreg");

        // Mapping back to the original species
        $homoRevTGs = $this->getHomoORFInfo($homoORFs, $toSpecies, $this->_species);

        // Get ORF names from the IDs
        $orfidlist = "''";
        foreach (array_keys($homoRevTGs) as $homoTG) {
                $orfidlist .= ", " . $homoRevTGs[$homoTG]['homo']['orfid'];
        }
        $id2ORF = array();
        $q  = "select O.orfid,O.orf,O.gene,P.protein from orfgene as O, ";
        $q .= "protein as P where O.species='{$this->_species}' and ";
        $q .= "P.tfid=O.orfid and orfid in ($orfidlist)";
        if ($this->_dbAccess->query("id2orf", $q)) {
                while ($row = $this->_dbAccess->nextObject("id2orf")) {
                        $id2ORF[$row['orfid']]['orf']     = $row['orf'];
                        $id2ORF[$row['orfid']]['gene']    = $row['gene'];
                        $id2ORF[$row['orfid']]['protein'] = $row['protein'];
                }
        }
        $this->_dbAccess->freeResult("id2orf");

        $mapping = array();
        // Put the pieces together: Documented Homologous Regulations
        foreach (array_keys($homoORFInfo) as $orf) {
                $tf = $homoORFInfo[$orf]['protein'];
                $homoTFid = $homoORFInfo[$orf]['homo']['orfid'];
                $mapping[$tf]['orf']  = $homoORFInfo[$orf]['homo']['orf'];
                $mapping[$tf]['gene'] = $homoORFInfo[$orf]['homo']['gene'];
                if (isset($docRegTo[$homoTFid])) {
                        foreach ($docRegTo[$homoTFid] as $homoTGorf => $homoTGgene) {
                                if (!isset($homoRevTGs[$homoTGorf])) continue;
                                $orfid = $homoRevTGs[$homoTGorf]['homo']['orfid'];
                                $gene = $id2ORF[$orfid]['gene'];
                                $orf  = $id2ORF[$orfid]['orf'];
                                $docReg[$tf][$orf]['gene'] = $gene;
                                $docReg[$tf][$orf]['homoGene'] = $homoRevTGs[$homoTGorf]['gene'];
                        }
                }
        }

        return array($aOrfTFs, $unknown, $docReg, $mapping);
}
```

Figure 8: Script for getting homologous regulators

2.4.3 Search for Regulatory associations

This functionality is used to find all regulatory associations among user input TFs and target genes. It also permits to filter the results based on the annotations associated with the regulatory associations, e.g. environmental conditions or association type.

```
function getRegAssociations($sTFs, $sORFGenes, $allTFs = false, $allGenes = false, $allToOne = false,
        $ecgroup, $eviCode, $assocType) {
                $regdoc = array();
                $regpot = array();
                $aORFGenes = array();
                $aTFs = array();
                $aPUnknown = array();
                $aGUnknown = array();

                $tflist = "";
                if ($allTFs) {
                        $q  = "select distinct O.orf, P.protein from protein as P, ";
                        $q .= "orfgene as O where O.species='{$this->_species}' and ";
                        $q .= "O.orfid=P.tfid and P.tfid in (select distinct tfid ";
                        $q .= "from regulation)";
                        if ($this->_dbAccess->query("docreg", $q)) {
                                while ($row = $this->_dbAccess->nextObject("docreg")) {
                                        $aTFs[$row['orf']] = $row['protein'];
                                }
                        }
                        $this->_dbAccess->freeResult("docreg");
                } else {
                        list($aTFs, $aPUnknown) = $this->orfgene2prot($sTFs);
                        foreach ($aTFs as $tf) {
                                $tflist .= ", '". addslashes($tf) . "'";
                        }
                }

                $orflist = "";
                if ($allGenes) {
                        // it is virtually all the genes
                        $q  = "select distinct orf,gene from orfgene ";
                        $q .= "where species='{$this->_species}'";
                        if ($this->_dbAccess->query("genes", $q)) {
                                while ($row = $this->_dbAccess->nextObject("genes")) {
                                        $g = $row['gene'];
                                        $o = $row['orf'];
                                        if ($g === "Uncharacterized") $aORFGenes[$o] = $o;
                                        else $aORFGenes[$o] = $g;
                                }
                        }
                        $this->_dbAccess->freeResult("genes");
                } else {
                        list($aORFGenes, $aGUnknown) = $this->orf2gene($sORFGenes);
                        foreach (array_keys($aORFGenes) as $o) {
                                $orflist .= ", '$o'";
                        }
                }
```

Figure 9: Script for finding regulatory associations among the target genes (1/3)

```
// Documented
$q = "";
if (strlen($orflist)>0) {
        $q .= "O.orf in (''$orflist) ";
}
if (strlen($tflist)>0) {
        if (strlen($q) > 0) {
                $q .= "and ";
        }
        $q .= "P.protein in (''$tflist) ";
}
if (strlen($q)>0) {
        $q = "where $q";
}
if ($allToOne) {
        $q  = "select O.orf, count(P.protein) as tfnumber from "
                        . "orfgene as O, regulation as R, protein as P " . $q
                        . "and P.tfid=R.tfid and R.targetid=O.orfid "
                        . " and O.species='{$this->_species}'";
        $q .= $this->getMysqlEviCode($eviCode, $assocType);
        $q .= $this->getMysqlEnvCond($ecgroup);
        $q .= " group by O.orf";
        if ($this->_dbAccess->query("docreg", $q)) {
                while ($row = $this->_dbAccess->nextObject("docreg")) {
                        $o = $row['orf'];
                        $g = $aORFGenes[$o];
                        $regdoc[$g]['orf'] = $o;
                        $regdoc[$g]['tfcount'] = $row['tfnumber'];
                }
                ksort($regdoc);
        }
} else {
        // One-to-one
        $q = "select O.orf, P.protein from orfgene as O, "
                . "regulation as R, protein as P "
                . $q . " and P.tfid=R.tfid and R.targetid=O.orfid"
                . " and O.species='{$this->_species}'";
        $q .= $this->getMysqlEviCode($eviCode, $assocType);
        $q .= $this->getMysqlEnvCond($ecgroup);
        if ($this->_dbAccess->query("regdoc", $q)) {
                while ($row = $this->_dbAccess->nextObject("regdoc")) {
                        $o = $row['orf'];
                        $regdoc[$row['protein']][$o] = $aORFGenes[$o];
                }
        }
}
```

Figure 10: Script for finding regulatory associations among the target genes (2/3)

```
        // Potential
        if (strlen($orflist)>0) {
                $orflist = "and O.orf in (''$orflist) ";
        }
        if (strlen($tflist)>0) {
                $tflist = "and P.protein in (''$tflist) ";
        }
        if ($allToOne) {
                $q  = "select O.orf, count(distinct P.protein) as tfnumber from ";
                $q .= "protein as P, tfconsensus as TFC, potregulation as PR, ";
                $q .= "orfgene as O where O.species='{$this->_species}' and ";
                $q .= "O.orfid=PR.targetid and PR.consensusid=TFC.consensusid ";
                $q .= "and TFC.tfid=P.tfid $orflist $tflist group by O.orf";
                if ($this->_dbAccess->query("regpot", $q)) {
                        while ($row = $this->_dbAccess->nextObject("regpot")) {
                                $o = $row['orf'];
                                $g = $aORFGenes[$o];
                                $regpot[$g]['orf'] = $o;
                                $regpot[$g]['tfcount'] = $row['tfnumber'];
                        }
                        ksort($regpot);
                }
                $this->_dbAccess->freeResult("regpot");
        } else {
                $q  = "select PR.potregid,O.orf,P.protein from potregulation as PR, ";
                $q .= "tfconsensus as TFC, protein as P, orfgene as O where ";
                $q .= "O.species='{$this->_species}' $orflist and O.orfid=PR.targetid ";
                $q .= "and PR.consensusid=TFC.consensusid and TFC.tfid=P.tfid ";
                $q .= $tflist;
                if ($this->_dbAccess->query("regpot", $q)) {
                        while ($row = $this->_dbAccess->nextObject("regpot")) {
                                $regpot[$row['protein']][$row['orf']] = $row['potregid'];
                        }
                }
                $this->_dbAccess->freeResult("regpot");
        }

        if (strlen($tflist)>0) {
                $aTFs = $this->getTFs(true, false, $tflist);
        } else {
                $aTFs = $this->getTFs(true, false);
        }
        return array($aTFs, $aPUnknown, $aORFGenes, $aGUnknown, $regdoc, $regpot);
}
```

Figure 11: Script for finding regulatory associations among the target genes (3/3)

The *allTFs* parameter considers all the transcription factors described in the PathoYeastract database and ignores the user input transcription factors, and the *allGenes* parameter considers all the genes described in the PathoYeastract database and ignores the user input target genes. The 'allToOne' parameter checks a specific gene against all the documented regulators based on homology. Furthermore, the results can be filtered based on direct and indirect regulation as well and DNA binding and/plus expression evidence.

2.4.4 Ranking genes by Gene Ontology

With the help of this function ranking of genes can be done by the three gene ontologies: Biological Process, Molecular Function or Cellular Component. This query returns a table organized by GO terms, ordered either by: a) the percentage of the genes (from the input list) associated with each GO term; b) or by the percentage of genes in the PathoYeastract database associated with each GO term; c) or by the p-value calculated through an Hypergeometric distribution considering the set of genes from the user input associated to a given GO term relative to the PathoYeastract described.

```
    function rankByGO($sOrfGenes, $goDB) {
            list($aOrfGenes, $unknown) = $this->orf2gene($sOrfGenes);

            $hyperN = $this->getTotalNumDBGenes();
            $hypern = count($aOrfGenes);

            $assocs = array();
            $csORFs = "''";
foreach (array_keys($aOrfGenes) as $orf) {
                    $csORFs .= ",'$orf'";
            }
            if ($goDB != "process" && $goDB != "function" && $goDB != "component") {
                    $goDB = "process";
            }
            $q  = "select T.goid, O.orf from $goDB"."list as T, orfgene as O ";
            $q .= "where O.orf in ($csORFs) and O.orfid=T.orfid";
            $q .= " and O.species='{$this->_species}'";
            $this->_dbAccess->query("goids", $q);
            while ($row = $this->_dbAccess->nextObject("goids")) {
                    $assocs[$row['goid']][$row['orf']] = $aOrfGenes[$row['orf']];
            }
            $this->_dbAccess->freeResult("goids");

            $terms = array();
            foreach (array_keys($assocs) as $id) {
                    $terms[$id]['setNum'] = count($assocs[$id]);
                    $terms[$id]['setPer'] = $terms[$id]['setNum'] / $hypern;

                    $q  = "select count(distinct orfid) as assoc from $goDB"."list where ";
                    $q .= "goid='$id' and orfid in (select distinct orfid from orfgene ";
                    $q .= "where species='{$this->_species}')";
                    $terms[$id]['dbNum'] = $this->_dbAccess->getObject("assoc", $q);
                    $terms[$id]['dbPer'] = $terms[$id]['setNum'] / $terms[$id]['dbNum'];

                    $q = "select * from $goDB where goid='$id'";
                    $this->_dbAccess->query("idinfo", $q);
                    if ($row = $this->_dbAccess->nextObject("idinfo")) {
                            $terms[$id]['term'] = $row['term'];
                            $terms[$id]['depth'] = $row['depth'];
                    }
                    $this->_dbAccess->freeResult("idinfo");
            }

            # Hypergeometric call to R
            $content = "";
            foreach (array_keys($terms) as $id) {
                    $content .= "print(phyper(".$terms[$id]['setNum'].",".$terms[$id]['dbNum'].",".
                            ($hyperN - $terms[$id]['dbNum']) . ",".$hypern."), digits=22)\n";
            }
            $Rfile = $this->_properties->get("yeastract.sync.tmp.local")."pvalue_".rand().".R";
            if (file_put_contents($Rfile, $content) === false) {
                    throw new Exception("Error writing R hypergeometric functions to disk!");
            }
            $Rpath = $this->_properties->get("yeastract.R.path");
            $Rbin  = $this->_properties->get("yeastract.R.bin");
            exec("$Rpath/$Rbin $Rfile", $aLines);
            $i = 0;
            foreach (array_keys($terms) as $id) {
                    $val = substr($aLines[$i],4);
                    $terms[$id]['pvalue'] = 1 - substr($aLines[$i],4);
                    $i++;
            }
```

Figure 12: Script for ranking genes by gene ontology

2.4.5 Ranking genes by Transcription Factors

This function allows the user to group a given list of genes (e.g., a set of co-activated genes from a microarray experiment) according to the TFs which are either documented or potential regulators. It then ranks those TFs according to two possible criteria: 1) % of genes associated to each TF; 2) or the p-value calculated through an Hypergeometric distribution considering the set of genes from the user input regulated by a given TF relative to the PathoYeastract described.

The TFs considered in this query can be either those in the input list, or all the TFs in the PathoYeastract database (by selecting the option *Check for all TFs*). Either documented or potential regulations can be considered (by selecting *Documented Regulations* or *Potential Regulations*, respectively).

Furthermore, the query enables the user to restrict its' search to the regulatory associations identified based on direct or indirect evidences. Depending on the options selected, the output is a table containing the input genes, grouped by TF and ordered by the percentage of genes regulated by the respective TF or a downloadable image of the regulatory network.

```
function rankByTF($sTFs, $sOrfGenes, $regs, $dbTFs=false, $ecgroup, $eviCode, $assocType, $simple=false) {
        list($aOrfGenes, $unknownGenes) = $this->orf2gene($sOrfGenes);

        $unknownTFs = array();
        if ($dbTFs) {
                $aTFs = array_keys($this->getTFs());
        } else {
                list($aTFs, $unknownTFs) = $this->orfgene2prot($sTFs);
        }

        $csTFs = "";
        foreach ($aTFs as $tf) {
                if (!empty($csTFs)) $csTFs .= ", ";
                $csTFs .= "'".addslashes($tf)."'";
        }
        $csGenes = "";
        foreach (array_keys($aOrfGenes) as $orf) {
                if (!empty($csGenes)) $csGenes .= ", ";
                $csGenes .= "'$orf'";
        }
        $regulations = array();
        if (empty($csGenes) || empty($csTFs)) {
                return $regulations;
        }

        if ($regs['doc']) {
                $q  = "select P.protein, O.orf from regulation as R, orfgene as ";
                $q .= "O, protein as P where P.tfid=R.tfid and R.targetid=O.orfid ";
                $q .= "and O.orf in ($csGenes) and P.protein in ($csTFs) and ";
                $q .= "O.species='$this->_species'";
                $q .= $this->getMysqlEviCode($eviCode, $assocType);
                $q .= $this->getMysqlEnvCond($ecgroup);
        } else {
                // FIX potential
                $q  = "select O.orf,P.protein from potregulation as PR, tfconsensus ";
                $q .= "as TFC, protein as P, orfgene as O where O.species='";
                $q .= $this->_species."' and O.orf in ($csGenes) and ";
                $q .= "O.orfid=PR.targetid and PR.consensusid=TFC.consensusid and ";
                $q .= "TFC.tfid=P.tfid and P.protein in ($csTFs)";
        }
        if ($this->_dbAccess->query("reg", $q)) {
                while ($row = $this->_dbAccess->nextObject("reg")) {
                        $regulations[$row['protein']][$row['orf']] = $aOrfGenes[$row['orf']];
                }
        }
        $this->_dbAccess->freeResult("reg");
```

Figure 13: Script for ranking genes by TFs (1/3)

```
// Non-regulated genes
$nonRegORFs = array();
$q  = "select distinct orf from orfgene where orf in ($csGenes) ";
$q .= "and orf not in (select distinct O.orf from orfgene as O, ";
$q .= "regulation as R, protein as P where O.orfid=R.targetid ";
$q .= "and O.orf in ($csGenes) and P.tfid=R.tfid and ";
$q .= "P.protein in ($csTFs))";
if ($this->_dbAccess->query("nonRegOrfs", $q)) {
        while ($row = $this->_dbAccess->nextObject("nonRegOrfs")) {
                $nonRegORFs[$row['orf']] = $aOrfGenes[$row['orf']];
        }
}
$this->_dbAccess->freeResult("nonRegOrfs");

// TODO: do a function to get the regulations
if ($simple) {
        return array($regulations, $unknownGenes, $unknownTFs, $aOrfGenes);
}

$regCount = array();
if (count($regulations) > 0) {
        $hyperN = $this->getTotalNumDBGenes();
        $hypern = count($aOrfGenes);

        foreach (array_keys($regulations) as $tf) {
                $regCount[$tf]['SetReg'] = count($regulations[$tf]);

                # % regs in Set
                $regCount[$tf]['SetPer'] = $regCount[$tf]['SetReg'] / $hypern;
        }

        # % regs in DB
        if ($regs['doc']) {
                $q2  = "select count(distinct R.targetid) as numGenes from ";
                $q2 .= "regulation as R, protein as P, orfgene as O where ";
                $q2 .= "P.tfid=R.tfid and P.tfid=O.orfid and O.species='";
                $q2 .= $this->_species."' and P.protein='";
        } else {
                $q2  = "select count(distinct PR.targetid) as numGenes from ";
                $q2 .= "potregulation as PR, protein as P, tfconsensus as TFC ";
                $q2 .= "where PR.consensusid=TFC.consensusid and ";
                $q2 .= "TFC.tfid=P.tfid and PR.targetid in (select orfid from ";
                $q2 .= "orfgene where species='{$this->_species}') and P.protein='";
        }
        foreach (array_keys($regCount) as $tf) {
                $tmp = $q2 . addslashes($tf)."'";
                $regCount[$tf]['DBReg'] = $this->_dbAccess->getObject("numGenes", $tmp);
                $regCount[$tf]['DBPer'] = $regCount[$tf]['SetReg'] / $regCount[$tf]['DBReg'];
        }
```

Figure 14: Script for ranking genes by TFs (2/3)

```
                # Hypergeometric call to R
                $content = "";
                foreach (array_keys($regCount) as $tf) {
                        $content .= "print(phyper(".$regCount[$tf]['SetReg'].",".$regCount[$tf]['DBReg'].",".
                                ($hyperN - $regCount[$tf]['DBReg']) . ",".$hypern."), digits=22)\n";
                }
                $Rfile = $this->_properties->get("yeastract.sync.tmp.local")."pvalue_".rand().".R";
                if (file_put_contents($Rfile, $content) === false) {
                        throw new Exception("Error writing R hypergeometric functions to disk!");
                }
                $Rpath = $this->_properties->get("yeastract.R.path");
                $Rbin  = $this->_properties->get("yeastract.R.bin");
                exec($Rpath . $Rbin . " $Rfile", $aLines);
                $i = 0;
                foreach (array_keys($regCount) as $tf) {
                        $val = substr($aLines[$i],4);
                        $regCount[$tf]['pvalue'] = 1 - substr($aLines[$i],4);
                        $i++;
                }
        }
        return array($regulations, $regCount, $unknownGenes, $unknownTFs, $aOrfGenes, $nonRegORFs);
}
```

Figure 15: Script for ranking genes by TFs (3/3)

2.4.6 View Regulation

This function enables the user to view information relative to a given regulatory association in a tabular format. It presents all the bibliographic references and annotated information associated with it.

```
function viewRegulation($sTFs, $sORFGenes) {
        $retObj = array();

        $tflist = "";
        list($aTFs, $aPUnknown) = $this->orfgene2prot($sTFs);
        foreach ($aTFs as $tf) {
                if (strlen($tflist)>0) $tflist .= ", ";
                $tflist .= "'". addslashes($tf) . "'";
        }
        $orflist = "";
        list($aORFGenes, $aGUnknown) = $this->orf2gene($sORFGenes);
        foreach (array_keys($aORFGenes) as $o) {
                if (strlen($orflist)>0) $orflist .= ", ";
                $orflist .= "'$o'";
        }
        if (empty($orflist) || empty($tflist)) {
                return $retObj;
        }

        $q  = "select RD.regulationid, P.protein, O.orf,O.gene, RD.association, ";
        $q .= "EV.code,EV.experiment, Pub.* ";
        $q .= "from pubmed as Pub, regulation as R, orfgene as O, ";
        $q .= "protein as P, regulationdata as RD, evidencecodeBSRG ";
        $q .= "as EV where O.species='".$this->_species;
        $q .= "' and P.tfid=R.tfid and R.targetid=O.orfid and O.orf ";
        $q .= "in ($orflist) and P.protein in ($tflist) and ";
        $q .= "RD.evidencecodeid=EV.evidencecodeid ";
        $q .= "and R.regulationid=RD.regulationid ";
        $q .= "and RD.pubmedid=Pub.pubmedid";
        if ($this->_dbAccess->query("regs", $q)) {
                while ($row = $this->_dbAccess->nextObject("regs")) {
                        $p  = $row['protein'];
                        $o  = $row['orf'];
                        $id = $row['pubmedid'];
                        $retObj[$p][$o][$id]['authors'] = $row['authors'];
                        $retObj[$p][$o][$id]['journal'] = $row['journal'];
                        $retObj[$p][$o][$id]['date']    = $row['date'];
                        $retObj[$p][$o][$id]['volume']  = $row['volume'];
                        $retObj[$p][$o][$id]['issue']   = $row['issue'];
                        $retObj[$p][$o][$id]['pages']   = $row['pages'];
                        $retObj[$p][$o][$id]['code']    = $row['code'];
                        $retObj[$p][$o][$id]['experiment'] = $row['experiment'];
                        $retObj[$p][$o][$id]['association'] = $row['association'];
                        $q  = "select EV.envconditiondesc from envcondition as EV, ";
                        $q .= "regulationdata as RD where EV.envconditionid=RD.envconditionid ";
                        $q .= "and RD.regulationid={$row['regulationid']} and RD.pubmedid=";
                        $q .= $row['pubmedid'];
                        $retObj[$p][$o][$id]['envcond'] =
                                $this->_dbAccess->getObject("envconditiondesc", $q);

                }
        }
        $this->_dbAccess->freeResult("regs");

        return $retObj;
}
```

Figure 16: Script for viewing a given regulatory association

2.4.7 View Protein

This function enables the user to obtain all the information related to a given protein to be visualized on the web interface.

```
function viewProtein($protein) {
    list($aTFs, $aPUnknown) = $this->orfgene2prot($protein);
    $tf = addslashes(array_pop($aTFs));
    $retObj = array();

                $q  = "select P.*,O.orfid,O.orf from protein as P, orfgene ";
                $q .= "as O where P.tfid=O.orfid and P.protein='$tf' ";
                $q .= "and O.species='$this->_species'";
        if ($this->_dbAccess->query("protein", $q)) {
            while ($row = $this->_dbAccess->nextObject("protein")) {
                $retObj['proteinname'] = $row['protein'];
                $retObj['aminoacidsequence'] = $row['aminoacidseq'];
                $retObj['orfname'] = $row['orf'];

                                    $qq  = "select D.description from description as D, ";
                                    $qq .= "orfgene as O where O.descriptionid=D.descriptionid ";
                                    $qq .= "and O.orfid='".$row['orfid']."'";
                $retObj['description'] =
                        $this->_dbAccess->getObject("description", $qq);
            }
        }
        $this->_dbAccess->freeResult("protein");

        // FIXME
        $q  = "select C.IUPACseq from tfconsensus as T, consensus as C, ";
        $q .= "protein as P, orfgene as O where P.tfid=T.tfid and ";
        $q .= "T.consensusid=C.consensusid and P.tfid=O.orfid and ";
        $q .= "P.protein='$tf' and O.species='".$this->_species."'";
        if ($this->_dbAccess->query("consensus", $q)) {
            while ($row = $this->_dbAccess->nextObject("consensus")) {
                $retObj['consensus'][] = $row['IUPACseq'];
            }
        }
        $this->_dbAccess->freeResult("consensus");

        $q = "select distinct R.tfid from regulation as R, protein as P, ";
        $q .= "orfgene as O where P.tfid=R.tfid and P.protein='$tf' and ";
        $q .= "P.tfid=O.orfid and O.species='".$this->_species."'";
        if ($this->_dbAccess->getObject("tfid", $q)) {
            $retObj['has_regdoc'] = true;
        }

        // FIXME
        $q = "select distinct C.IUPACseq from consensus as C, potentialregulation as PR where C.protein = '$tf' and C.consensus = PR.consensus";
        if ($this->_dbAccess->getObject("protein", $q)) {
            $retObj['has_regpot'] = true;
        }

        return $retObj;

}
```

Figure 17: Script for viewing the information relative to a given protein

2.4.8 View GO

This function enables the user to obtain all information relative to Gene Ontology annotations associated to a given ORF.

```
function viewGO($orfname) {
  list($aYeastInfo, $aPUnknown) = $this->getInfoOrfProt($orfname);
  $retObj = array();
        $aOrf = array_pop($aYeastInfo);
        $retObj['orf'] = $aOrf['orf'];
        $retObj['protein'] = $aOrf['prot'];
        $retObj['description'] = $aOrf['desc'];

        $q  = "select P.goid,P.term from processlist as PL, process as P, ";
        $q .= "orfgene as O where PL.orfid=O.orfid and O.orf='".$aOrf['orf'];
        $q .= "' and PL.goid=P.goid and O.species='$this->_species'";
        if ($this->_dbAccess->query("process", $q)) {
            while ($row = $this->_dbAccess->nextObject("process")) {
                    $p = ($row['term'] == "biological_process")? "biological_process unknown": $row['term'];
                $retObj['process'][$row['goid']] = $p;
            }
        }
        $this->_dbAccess->freeResult("process");

        $q  = "select P.goid,P.term from functionlist as PL, function as P, ";
        $q .= "orfgene as O where PL.orfid=O.orfid and O.orf='".$aOrf['orf'];
        $q .= "' and PL.goid=P.goid and O.species='$this->_species'";
        if ($this->_dbAccess->query("function", $q)) {
            while ($row = $this->_dbAccess->nextObject("function")) {
                    $f = ($row['term'] == "molecular_function")? "molecular_function unknown": $row['term'];
                $retObj['function'][$row['goid']] = $f;
            }
        }
        $this->_dbAccess->freeResult("function");

        $q  = "select P.goid,P.term from componentlist as PL, component as P, ";
        $q .= "orfgene as O where PL.orfid=O.orfid and O.orf='".$aOrf['orf'];
        $q .= "' and PL.goid=P.goid and O.species='$this->_species'";
        if ($this->_dbAccess->query("component", $q)) {
            while ($row = $this->_dbAccess->nextObject("component")) {
                    $c = ($row['term'] == "cellular_component")? "cellular_component unknown": $row['term'];
                $retObj['component'][$row['goid']] = $c;
            }
        }
        $this->_dbAccess->freeResult("component");

        return $retObj;
}
```

Figure 18: Script for viewing the Gene Ontology annotations of a given ORF

2.4.9 View LocusInfo

This function enables the user to retrieve information relative to a given ORF, like alias, description, gene sequence, promoter sequence, etc.

```
function viewOrfgene($orfgene, $potentialID = false, $regulator = "") {
        list($aORFGenes, $aGUnknown) = $this->orf2gene($orfgene);
        $aORFs = array_keys($aORFGenes);
        $orf = array_pop($aORFs);
        if (!isset($orf)) {
                if (strlen($orfgene)>0) {
                        throw new Exception("The ORF '$orfgene' does not exist, in species '{$this->_species}'!");
                } else {
                        throw new Exception("No ORF/Gene was inserted!");
                }
        }
        $retObj = array();
        $q  = "select * from orfgene where orf='$orf' ";
        $q .= "and species='$this->_species'";
        if ($this->_dbAccess->query("orfgene", $q)) {
                while ($row = $this->_dbAccess->nextObject("orfgene")) {
                        $retObj['orfname'] = $row['orf'];
                        $retObj['genename'] = $row['gene'];
                        $retObj['promotersequence'] = $row['promoterseq'];
                        $retObj['genesequence'] = $row['geneseq'];

                        $qq  = "select P.protein,D.description from protein as P, ";
                        $qq .= "orfgene as O, description as D where P.tfid=O.orfid ";
                        $qq .= "and D.descriptionid=O.descriptionid and tfid='";
                        $qq .= $row['orfid']."'";
                        if ($this->_dbAccess->query("protdesc", $qq)) {
                                $r = $this->_dbAccess->nextObject("protdesc");
                                $retObj['proteinname'] = $r['protein'];
                                $retObj['description'] = $r['description'];
                        }
                        $this->_dbAccess->freeResult("protdesc");

                        $qq = "select alias from alias where orfid='".$row['orfid']."'";
                        if ($this->_dbAccess->query("alias", $qq)) {
                                while ($r = $this->_dbAccess->nextObject("alias")) {
                                        $retObj['altname'][] = $r['alias'];
                                }
                        }
                        $this->_dbAccess->freeResult("alias");

                }

        }
        $this->_dbAccess->freeResult("orfgene");

        $q = "select distinct orfname from regulation where orfname = '$orf'";
        if ($this->_dbAccess->getObject("orfname", $q)) {
                $retObj['has_regdoc'] = true;
        }
        $q = "select distinct orfname from potentialregulation where orfname = '$orf'";
        if ($this->_dbAccess->getObject("orfname", $q)) {
                $retObj['has_regpot'] = true;
        }
```

Figure 19: Script for viewing the information relative to a given ORF

3. Results

3.1. Predicting gene and genomic regulation

The query “Rank by TF” enables the user to automatically select and rank the transcription factors potentially involved in the regulation of the genes of interest. The TFs are represented in a descending order of relevance score calculated for each transcription factor which are cross checked against the regulatory data available on the PathoYeastract database. There are several filters available for the user to choose from so they can narrow down the search depending on the environmental conditions or species to check the regulation against.

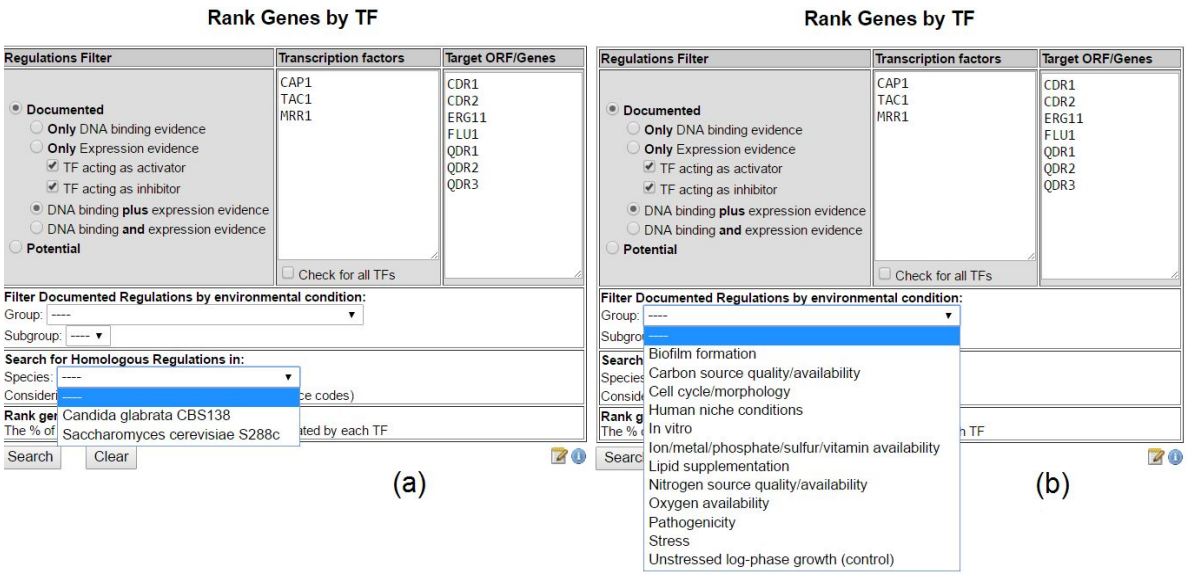

Figure 20: (a) Rank by TF filtered by species; (b): Rank by TF filtered by environmental conditions

To explain the functionalities, the example of the *TPO3* gene has been used, encoding a polyamine transporter in *Candida glabrata*. Using the tool quick search on the homepage the user can gather all the Locus, Protein and GO information for the gene of interest.

Locus Info | Protein Info | **GO Info**

Protein Name	Tpo3p
Description	Verified ORF; Predicted polyamine transporter of the major facilitator superfamily; required for azole resistance
GO Terms: Biological Process	. spermine transport . transmembrane transport
GO Terms: Molecular Function	. spermine transmembrane transporter activity
GO Terms: Cellular Component	. fungal-type vacuole membrane . plasma membrane . integral component of membrane
Open in CGD	

Go Back

(a)

Locus Info | Protein Info | GO Info

Standard Name	TPO3
Systematic Name	CAGL0I10384g
Alias	CAG60636.1, CAGL-CDS1349.1, CAGL-IPF6440
Description	Verified ORF; Predicted polyamine transporter of the major facilitator superfamily; required for azole resistance
Gene Sequence Download FASTA View Seq	*Sequence hidden*
Promoter Sequence Download FASTA	>CAGL0I10384g TPO3 GTAGCCTCCACGTAAGCACGCGATATCGGTTGTTTGGGGGGGGGGGGGAGCCAAGGGGTT TCACAGAGAGAGAGGCACTACACGTACGGGTATTCCTCTGCAGAGCATGCCATTCCCATT TCCCACACACAGTAACGATGCAAGAGATCCAATTTCACTATTTTCCCCCCAGTTCTTCAG ATGTTTATGCATAGAACCCGGAACAACTCCCCCAGAACACCACGCACAACTGGGTTTAAA AAAAACCATCCCTGTGCTTCTCTCTTCTCTACAGGCATCTTTTGTCTCTTTGAAGAAGTG ATTGGTAGAAACTCGGCGAGCCCGGAAGTAGAGCATTCACTCGCACTGTAAACACCGTAT AGATCAACGATAAATTTCCAAACAATGAATAGATTATACAGAAGCTTCGATTTTCCATCA ATTATATGATTTCGTGCGGTACTTTCTTAGTAGGAAGTGAAAAAAAAACACTACACAAAAT CAAATCAGAATTTGAACCTTCGGTGATATAATCCTGGTCAAAAAAAATTGACTTGAAAAT TTAATTGAAAAAAAAATCTTGTAGGGCATCATACAATAAGGATAACAATAGTGGGATGAA CAGCAATAGCAAGCAACTTTTGAGTTGTTTTTTAGTGCAACGCCAATAGTACCAGAGTTT AAACGTATATCTTATAGCTAAGTCGCACAAAAAGAAATACTAGTCTTGAAAATAGAAGAA AAAATAAAAAAAGCCACCGTTCTATTCCGCGATCTTTTATTTTAAAAAACACATATATAT AAGTACTAGACTTGACTTGCTTTTGACTCACTGCAGATTTGTTACACAGAAAGCAAATCA TACAAGAGTATCTGCTCCCACAAAAGTATCTGTTCACACAAAGGCATCTGTTTTTATACA AAACAGCACTACACTTTTTGCACTTTTCATTAATATAATAATATAACACAGATAAAAAGG AAAAAGACCCCCAACAGATAATCTTAACAAAAAATTACAA
Open in CGD	

Go Back

(b)

Locus Info | **Protein Info** | GO Info

Protein Name	Tpo3p
Description	Verified ORF; Predicted polyamine transporter of the major facilitator superfamily; required for azole resistance
Aminoacid Sequence Download FASTA	>CAGL0I10384g TPO3 MVDQESLVSFSSETSQSINSDIDIESQQQPRQYIPSNEKDGNKERLHLTRTETVKSLQEM GMTQDAPIPDVNAPQTTTKNAIFPEEYTMETPTGLVPVATLQSLGRTSTAISKSRTRQIE RSVSRRSQNIAASSNSSNKEELEDEEEVSSDMSNQQPELDPEIEFVTFVTGDPTNPHNWP LWIRWAYTVILSCLVICVAYGSACITGGLFTVQEQYHVGLEAAILSCSLMVIGFSLGPLI WSPVSDLYGRRLAYFISMGLYTIFNIPCALSPNLGGLLVCRFLCGVFSSSGLCLVGGSIA DMFPSETRGRAIAFFAFAPYTGPIIGPLVNGFVSVCTRRMDLIFWINMAFAGVMWIIVAF IPETYAPVILKWRAAKLRKETGNPKIMTEQEAQGVSVNEMMKACLIRPLYFAVTEPVLDL TCFYVCLIYSLLYAFFFAFPVVFGELYGYKDNLIGLMFIPILIGATMALATTFYCENEYL KLVKKRKPTPEDRLFGAMIGAPFAAAALWILGATSYKHIIWVGPASSGLAFGYGMVLIYY SLNNYIIDCYVQYASSALATKVFLRSAGGAAFPLFTNQMYHKLGLQWASWLLAFISTAMI LLPFGFYYYGKTLRHKLSKKDYSIDTIEGY*
Open in CGD	

Go Back

(c)

Figure 21: The gene information of TPO3- (a) GO information; (b) Locus information; (c) Protein information

For example, *TPO3* is known to be regulated by the transcription factor Pdr1 in *Candida glabrata*, but based on the PathoYeastract prediction tool, based on the regulators involved in the control of the homologs of Tpo3 in *S. cerevisiae TPO3* is predicted to be controlled by many more transcription factors

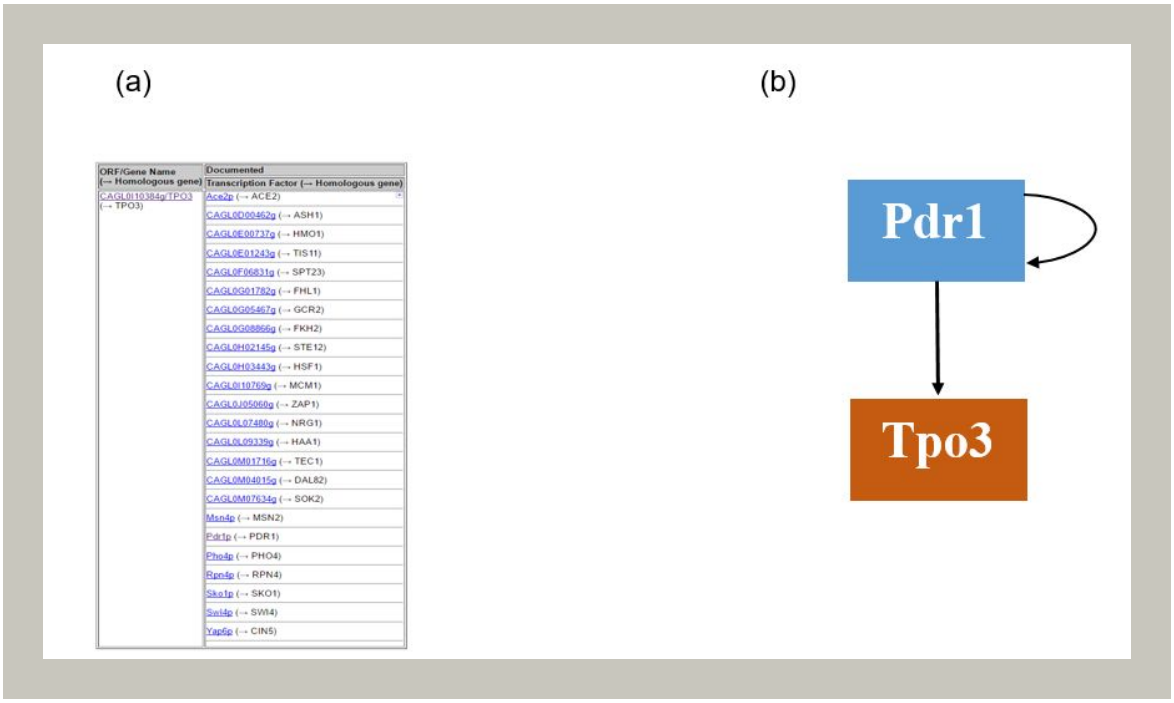

Figure 22: Regulation of the Tpo3 gene in (a) in *Saccharomyces cerevisiae;* (b) *Candida glabrata*

It is interesting to observe that there is no documented regulation found for Tpo3 in *Candida albicans* whilst there has been documented the regulation of Tpo3/Pdr1 regulations in both *S. cerevisiae* and *C. glabrata.* . Among the transcription factors predicted to regulate Tpo3, based on homology to the homologous regulatory network in *S. cerevisiae*, is, for instance, Haa1, the key regulator of weak acid stress resistance, which suggests that Tpo3 may also play a role in this phenomenon.

Alternatively, using the "Search for Genes" query it is possible to pinpoint all the genes that have been shown to be regulated by a given transcription factor. Keeping to the drug resistance case study, using this tool it is possible to observe that there are 399 genes known to be regulated by the *C. glabrata* Pdr1 TF, whereas there are only 46 genes known to be regulated by the *C. albicans* Tac1 TF in (Figure 23).

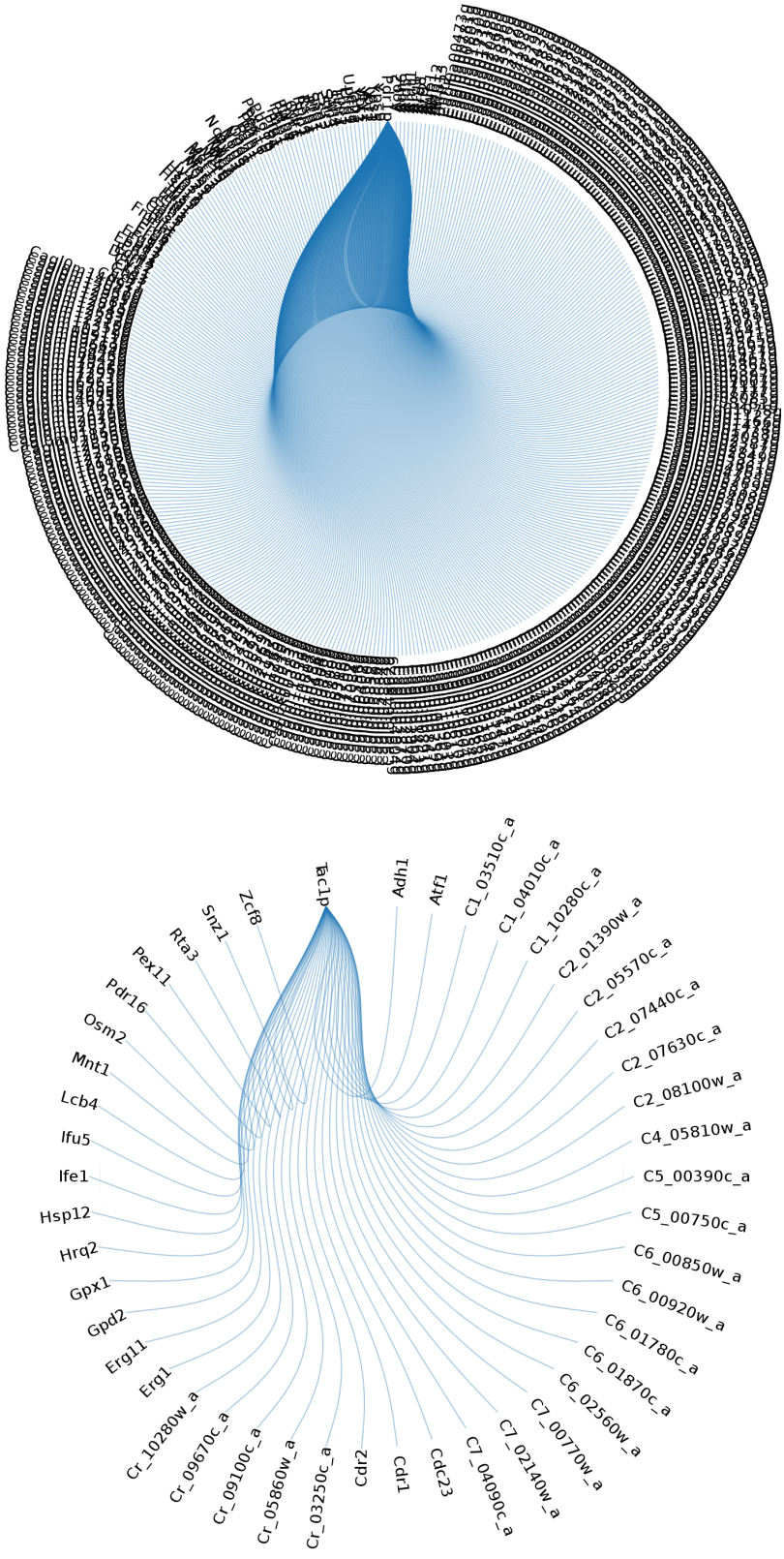

Figure 23 - The Tac1 / Pdr1 regulon in *C. albicans* (top) / *C. glabrata* (bottom) as obtained using the "Search for target genes" in the PathoYeastract database.

Interestingly, the list of genes whose expression is controlled by Pdr1 or Tac1 is also quite diverse in terms of associated functions, far beyond the classical targets, the multidrug efflux pumps of the ATP-Binding Cassette superfamily Cdr1 and Cdr2. Both lists include shared genes and functions, such as Hsp12, a stress resistance related protein chaperone, the Erg11 gene, the target of azole antifungal drugs, but also genes associated to central metabolic pathways. For example, *ADH1* and *SNZ1* were identified as targets of the Tac1 TF, being related to central carbon metabolism and vitamin B synthesis, respectively. This observation, raises the possibility of either Tac1 playing additional roles in *C. albicans* biology or that Adh1 and Snz1 contribute somehow to drug resistance.

3.2. Analyzing Genome-wide regulation

One of the key features of the genome-wide regulation study in the PathoYestract is aimed to study the regulatory network of a given sets of genes/TFs which controls a certain genome-wide expression remodeling. To use this feature, the query "Rank by TF" is used. By using this tool, the genome-wide regulation of the target genes can be predicted. The user provides a list of genes which are upregulated under certain environmental condition, the program searches for all the predicted TFs which are responsible for regulating the genes provided by the user and rank them by importance. For example, considering the list of genes which are targets of Upc2a in *Candida glabrata*, the "Rank by TF" query shows that, as expected Upc2ap regulates 100% of the gene list, but also highlights the fact that Pdr1 is the regulator of 50% of the Upc2a target genes. This is in agreement with the observation that many of the Upc2a targets encode enzymes of the ergosterol biosynthesis pathway, which is clearly involved in azole drug resistance in *Candida glabrata* [23].

Transcription Factor	% in user set	% in PathoYeastract	p-value ▲	Target ORF/Genes
Upc2ap	100.00%	80.00%	0.000000000000000	ERG4 ERG1 ERG11 ERG3 ERG6 ERG2 ERG9 ERG5
Pdr1p	50.00%	1.00%	0.000083739276869	ERG4 ERG11 ERG3 ERG6
Ap1p	25.00%	0.74%	0.005123886498722	ERG3 ERG6
Yap7p	12.50%	0.58%	0.023242054327728	ERG11
CAGL0I07755g	12.50%	0.33%	0.066565558472723	ERG4
CAGL0G08844g	12.50%	0.29%	0.080678203921511	ERG2

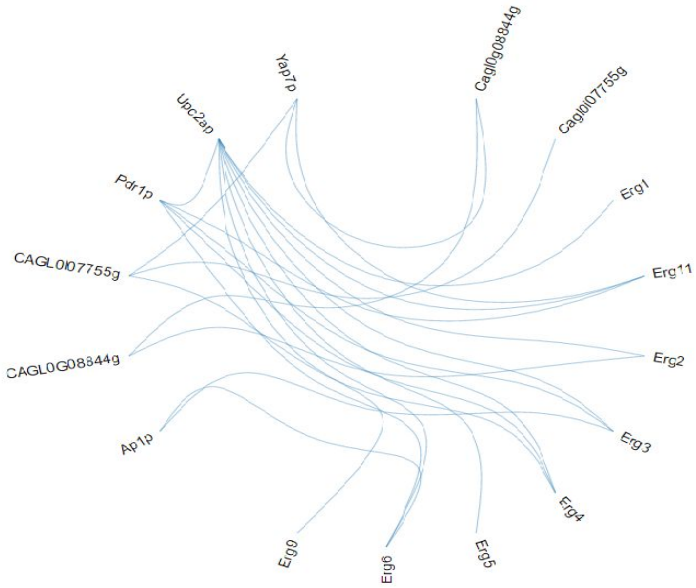

Figure 24: The TFs that control the Upc2a target genes, ranked by their relative importance (top) or displayed as a regulatory network (bottom)

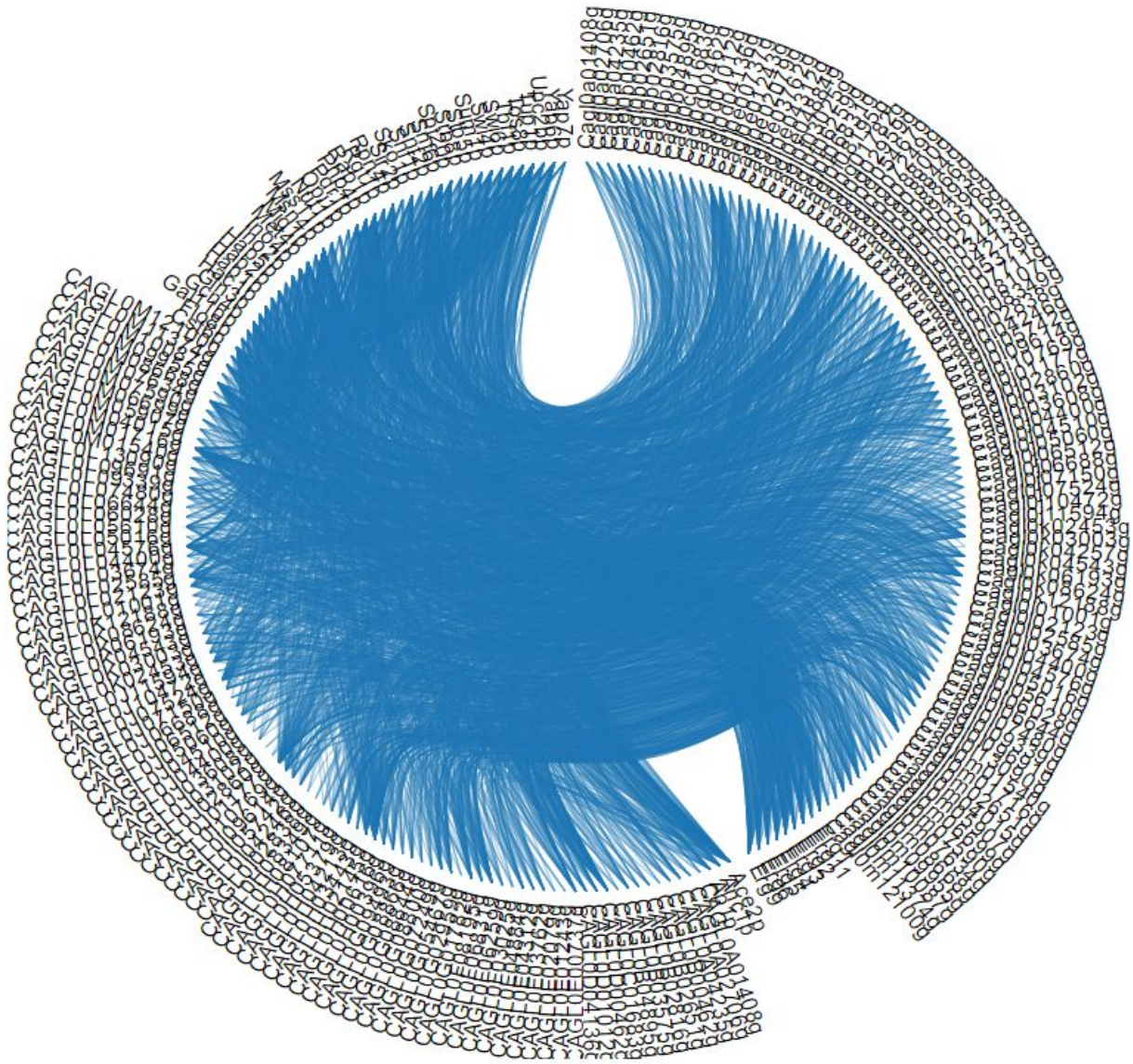

Figure 25: Regulators of the C. glabrata Upc2a target genes, based on their homologous regulation in *Saccharomyces cerevisiae*, as predicted by the "Rank by TF" tool in PathoYestract database

Similarly, for the same set of genes, but considering the predicted regulators of the homologous gene set in *Saccharomyces cerevisiae*, the number or regulators is comparatively enormous. Some of the transcription factors are similar in both analysis, for example Pdr1, predicted to control 62.50% of the list of genes.. It can be predicted from this study that other regulators such as the general stress response TF Msn4,could be responsible for the regulations of a set of upregulated genes in *C. glabrata* under the specific environmental conditions.

4. Conclusions

In this section the achievements of my Master's thesis project and the scopes for future work and a clear direction are discussed.

4.1 Summary of the achievements

With the radical change and development in the field of Systems Biology research in the past decade, the study of gene regulatory networks has become the inevitable state of the art technology. This cutting edge approach is not only fast, it is accurate in most of the cases. The greatest motivation to develop the PathoYeastract website and database is to aid the scientific community all around the world with a reliable and complete tool to focus and study on the transcription regulations of pathogenic yeasts on the molecular basis of candidaemia and its prophylaxis and treatment.

I have contributed to develop the working tool to integrate the both the documentation of the regulation data and constructing the gene regulatory network based on the search as well as represent them visually for the sake of the usability at the external user level. At present the web portal (http://pathoyeastract.org/) is fully functioning being publicly available and freely accessible. If someone is interested to study a specific set of TFs/genes for either *Candida glabrata* or *Candida albicans*, for the easement there had been two different tabs assigned on the homepage to redirect them to the databank and the webpage of the specific species. As there are regulation data for all the three species i.e. *Candida glabrata, Candida albicans* and *Saccharomyces cerevisiae* available on the database, the user can easily compare the gene regulations interspecies easily which provides the scope to study the gene regulatory network amongst different species resulting wider and more apprehensive study at a cross-species level.

However, as the size of the gene regulatory network increases, it makes it difficult to document and study the gene regulations. When the size of the network exceeds a certain size, it makes it eventually impossible to identify the interesting regulations and verify the predictions against the experimental results while the automated simulation using the computational tool can perform the tasks seamlessly. The curation of the website and the periodical update has been an important aspect from the very beginning of the development of the website. I was finally able to make the architecture of the database to

adapt to the system so that although the verification of an updated source is manual, the update is automated with a script, but it needs human supervision for the unpredictable cases that always appear.

It enables the website to be always updated with new documented data relevant to the database. Though the curators of the website are always keen to provide the user with most updated and current data, it is important to empower the database system to perform the task on its own without the need of somebody performing the update manually which can be possibly done through further work on the database development.

4.2. Future Direction

Although the PathoYestract database and website is up and running in a fully functional state at this moment, there is still scope for further development and advancement. For example, at this moment there are documented regulations, potential regulations and TF binding sites for two *Candida* species and the model organism *Saccharomyces cerevisiae* have been documented in the website. There is a window of opportunity to expand the database with other closely related *Candida* species such as; *Candida parapsilosis* and *C. dubliniensis*, *C. orthopsilosis*, *C. krusei* and *C. lusitaniae* which can be proven significant to document genetic regulations similar to that in the species which have already been documented. The study of gene regulatory network in pathogenic yeast is emerging all over the world as researchers are focusing on different organisms. In the future the expansion of the database is possible especially through the development of more complex dedicated tools.

Bibliography

1. Mayer FL, Wilson D, Hube B. *Candida albicans* pathogenicity mechanisms. *Virulence*. 2013;4(2):119-128.
2. Pfaller MA, Diekema DJ. Epidemiology of Invasive Candidiasis: a Persistent Public Health Problem. *Clinical Microbiology Reviews*. 2007;20(1):133-163.
3. Caggiano G, Puntillo F, Coretti C, et al. Candida Colonization Index in Patients Admitted to an ICU. *International Journal of Molecular Sciences*. 2011;12(10):7038-7047.
4. Klempp-Selb B, Rimek D, Kappe R (2000) Karyotyping of Candida albicans and Candida glabrata from patients with Candida sepsis. Mycoses 43:159–163.
5. H. J. Lo, J. R. Köhler, B. DiDomenico, D. Loebenberg, A. Cacciapuoti, G. R. Fink. Nonfilamentous C. albicans mutants are avirulent. Cell. 1997 September *5*; 90(*5*): 939–949.
6. Dias, P. J., & Sá-Correia, I. The drug: H+ antiporters of family 2 (DHA2), siderophore transporters (ARN) and glutathione: H+ antiporters (GEX) have a common evolutionary origin in hemiascomycete yeasts. BMC genomics, (2013), 14(1), 901.
7. Sá-Correia, I., dos Santos, S. C., Teixeira, M. C., Cabrito, T. R., & Mira, N. P. Drug: H+ antiporters in chemical stress response in yeast. *Trends in microbiology*, (2009). *17*(1), 22-31.
8. Coste AT, *et al.* TAC1, transcriptional activator of CDR genes, is a new transcription factor involved in the regulation of Candida albicans ABC transporters CDR1 and CDR2. *Eukaryot Cell* (2004) 3(6):1639-52.
9. Morschhauser J, *et al.* The transcription factor Mrr1p controls expression of the MDR1 efflux pump and mediates multidrug resistance in Candida albicans.*PLoS Pathog* (2007),3(11):e164.
10. Clarissa J. Nobile *et. al.,* A Recently Evolved Transcriptional Network Controls Biofilm Development in Candida albicans, Cell, (2012) Volume 148, Issues 1-2, p126– 138, 20.
11. Catarina Costa, Jonathan Ribeiro, Isabel M. Miranda, Ana Silva-Dias, Mafalda Cavalheiro, Sofia Costa-de-Oliveira, Acácio G. Rodrigues and Miguel C. Teixeira. Clotrimazole Drug Resistance in *Candida glabrata* Clinical Isolates Correlates with Increased Expression of the Drug:H^+ Antiporters CgAqr1, CgTpo1_1, CgTpo3, and CgQdr2, *Frontiers in microbiology,* (2016).
12. Miguel C. Teixeira, Pedro T. Monteiro, Joana F. Guerreiro, Joana P. Gonçalves, Nuno P. Mira, Sandra C. dos Santos, Tânia R. Cabrito, Margarida Palma, Catarina Costa,

Alexandre P. Francisco, Sara C. Madeira, Arlindo L. Oliveira, Ana T. Freitas, Isabel Sá-Correia. The YEASTRACT database: an upgraded information system for the analysis of gene and genomic transcription regulation in *Saccharomyces cerevisiae,* Nucl. Acids Res., (2014) 42: D161-D166, Oxford University Press.

13. Dário Abdulrehman, Pedro T. Monteiro, Miguel C. Teixeira, Nuno P. Mira, Artur B. Lourenço, Sandra C. dos Santos, Tânia R. Cabrito, Alexandre P. Francisco, Sara C. Madeira, Ricardo S. Aires, Arlindo L. Oliveira, Isabel Sá-Correia, Ana T. Freitas. YEASTRACT: providing a programmatic access to curated transcriptional regulatory associations in *Saccharomyces cerevisiae* through a web services interface Nucl. Acids Res., (2011) 39: D136-D140, Oxford University Press.
14. Pedro T. Monteiro, Nuno Mendes, Miguel C. Teixeira, Sofia d'Orey, Sandra Tenreiro, Nuno Mira, Hélio Pais, Alexandre P. Francisco, Alexandra M. Carvalho, Artur Lourenço and Isabel Sá-Correia, Arlindo L. Oliveira, Ana T. Freitas. YEASTRACT-DISCOVERER: new tools to improve the analysis of transcriptional regulatory associations in *Saccharomyces cerevisiae*. Nucl. Acids Res., (2008) 36: D132-D136, Oxford University Press.
15. Miguel C. Teixeira, Pedro Monteiro, Pooja Jain, Sandra Tenreiro, Alexandra R. Fernandes, Nuno P. Mira, Marta Alenquer, Ana T. Freitas, Arlindo L. Oliveira, and Isabel Sá-Correia. The YEASTRACT database: a tool for the analysis of transcription regulatory associations in *Saccharomyces cerevisiae*. Nucl. Acids Res., (2006) 34: D446-D451, Oxford University Press.
16. Tsai,H.K., Chou,M.Y., Shih,C.H., Huang,G.T., Chang,T.H. and Li,W.H. MYBS: a comprehensive web server for mining transcription factor binding sites in yeast. Nucleic Acids Res., (2007) 35, W221–W226.
17. Wingender,E., Chen,X., Fricke,E., Geffers,R., Hehl,R., Liebich,I., Krull,M., Matys,V., Michael,H., Ohnhauser,R. et al. The TRANSFAC system on gene expression regulation. Nucleic Acids Res., (2001) 29, 281–283.
18. van Helden,J. Regulatory sequence analysis tools. Nucleic Acids Res., (2003) 31, 3593–3596.
19. Chang,D.T., Huang,C.Y., Wu,C.Y. and Wu,W.S. YPA: an integrated repository of promoter features in Saccharomyces cerevisiae. Nucleic Acids Res., (2011) 39, D647–D652.
20. de Boer,C.G. and Hughes,T.R. YeTFaSCo: a database ofevaluated yeast transcription factor sequence specificities. Nucleic Acids Res., (2012) 40, D169–D179.
21. Inglis DO, Arnaud MB, Binkley J, Shah P, Skrzypek MS, Wymore F, Binkley G, Miyasato SR, Simison M, Sherlock G. The Candida genome database incorporates

multiple Candida species: multispecies search and analysis tools with curated gene and protein information for Candida albicans and Candida glabrata. *Nucleic Acids Res* (2012) 40(Database issue): D667-74.

22. Cherry JM, Hong EL, Amundsen C, Balakrishnan R, Binkley G, Chan ET, Christie KR, Costanzo MC, Dwight SS, Engel SR, Fisk DG, Hirschman JE, Hitz BC,Karra K, Krieger CJ, Miyasato SR, Nash RS, Park J, Skrzypek MS, Simison M, Weng S, Wong ED. Saccharomyces Genome Database: the genomics resource of budding yeast. Nucleic Acids Res. 2012 Jan;40(Database issue): D700-5.

23. Whaley SG, Caudle KE, Vermitsky JP, Chadwick SG, Toner G, Barker KS, Gygax SE, Rogers PD. UPC2A is required for high-level azole antifungal resistance in Candida glabrata. Antimicrob Agents Chemother. 2014 Aug;58(8):4543-54.

24. The Gene Ontology Consortium (2000) Gene Ontology: tool for the unification of biolog. Nature Genet., 25: 25-29

25. The Gene Ontology Consortium (2015) Gene Ontology Consortium: going forward. Nucleic Acids Res., 43(Database issue): D1049-D1056

26. Iupac-Iub Comm. On Biochem. Nomencl (1970) Abbreviations and symbols for nucleic acids, polynucleotides, and their constituents. Biochemistry. 9: 4022–4027

27. Attrill H, Falls K, Goodman JL, Millburn GH, Antonazzo G, Rey AJ, Marygold SJ; the FlyBase Consortium (2016) FlyBase: establishing a Gene Group resource for Drosophila melanogaster. Nucleic Acids Res. 44(D1):D786-D792

28. Bult CJ, Eppig JT, Blake JA, Kadin JA, Richardson JE; and the Mouse Genome Database Group (2016) The Mouse Genome Database 2016. Nucleic Acids Res. 2016 Jan 4;44 (D1): D840-D847