Pacific Symposium on Biocomputing 13:539-550(2008)

GSE: A COMPREHENSIVE DATABASE SYSTEM FOR THE REPRESENTATION, RETRIEVAL, AND ANALYSIS OF MICROARRAY DATA

TIMOTHY DANFORD, ALEX ROLFE, AND DAVID GIFFORD

MIT Computer Science and Artificial Intelligence Laboratory
32-G538
77 Massachusetts Ave
Cambridge, MA, 02139

We present GSE, the Genomic Spatial Event database, a system to store, retrieve, and analyze all types of high-throughput microarray data. GSE handles expression datasets, ChIP-chip data, genomic annotations, functional annotations, the results of our previously published Joint Binding Deconvolution algorithm for ChIP-chip, and precomputed scans for binding events. GSE can manage data associated with multiple species; it can also simultaneously handle data associated with multiple 'builds' of the genome from a single species. The GSE system is built upon a middle software layer for representing streams of biological data; we outline this layer, called GSEBricks, and show how it is used to build an interactive visualization application for ChIP-chip data. The visualizer software is written in Java and communicates with the GSE database system over the network. We also present a system to formulate and record *binding hypotheses*- simple descriptions of the relationships that may hold between different ChIP-chip experiments. We provide a reference software implementation for the GSE system.

1. Introduction

1.1. *Large-Scale Data Storage in Bioinformatics*

The data storage and computational requirements for high-throughput genomics experiments have grown exponentially over the last several years. Some methods simultaneously collect hundreds-of-thousands, or even millions, of data points. Microarrays contain several orders of magnitude more probes than just a few years ago. Short read sequencing produces 'raw' datasets requiring over a terabyte of computer disk storage[11]. Combine these with massive genome annotation datasets, cross-species sequence alignments mapped on a per-base level, thousands of publicly-available microarray expression experiments, and growing databases of sequence motif

Pacific Symposium on Biocomputing 13:539-550(2008)

information – and you have a wealth of experimental results (and large scale analyses) available to the investigator on a scale unimagined just a few years ago.

Successful analysis of high-throughput genome-wide experimental data requires careful thought on the organization and storage of numerous dataset types. However, the ability to effectively store and query large datasets has often lagged behind the sophistication of the analysis techniques that are developed for that data. Many publicly available analysis packages were developed to work in smaller systems, such as yeast [19]. Flat files are sufficient for simple organisms, but for large datasets they will not fit into main memory and cannot provide the random access necessary for a browsing visualizer.

Modern relational databases provide storage and query capabilities for these vertebrate-sized datasets. Built to hold hundreds of gigabytes to terabytes of data, they provide easy access through a well-developed query language (SQL), network accessibility, query optimizations, and facilities for easily backing up or mirroring data across multiple sites.

Most bioinformatics tools that have taken advantage of database technology, however, are web applications. Often these tools are the front-end interfaces to institutional efforts that gather publicly-available data or are community resources for particular model organisms or experimental protocols. Efforts like UCSC's genome browser and its backing database[12], or the systems of GenBank[2], SGD[6], FlyBase[4], and many others, are all examples of web interfaces to sophisticated database systems for the storage, search, and retrieval of species-based or experiment-based data.

1.2. *A Desktop Analysis Client and a Networked Database Server*

The system that we describe here bridges the gap between the web applications that exist for large datasets and the analysis tools that work on smaller datasets. GSE consists of back-end tools for importing data and running batch analyses as well as visualization software for interactive browsing and analysis of ChIP-chip data.

The visualization software, distributed as a Java application, communicates over the network with the same database system as the as the middle-layer and analysis tools. Our visualization and analysis software is written in Java and are distributed as desktop applications. This lets us combine much of the flexibility of a web-application interface (lightweight, no flat

Pacific Symposium on Biocomputing 13:539-550(2008)

files to install, and can run on any major operating system) with the power of not being confined to a browser environment. Our system can also connect to datastreams from multiple databases simultaneously, and can use other system resources normally unavailable to a browser application.

This paper describes the platform that we have developed for the storage of ChIP-chip and other microarray experiments in a relational database. It then presents our system for intepreting ChIP-chip data to identify binding events using our previously published "Joint Binding Deconvolution" (JBD) algorithm[17]. Finally, we show how we can build a system for the dynamic and automatic analysis of ChIP-chip binding calls between different factors and across experimental conditions.

2. A Database System for ChIP-chip Data

The core of our system is a database schema to represent ChIP-chip data and associated metadata in a manner independent of specific genomic coordinates and of the specific array platform.

2.1. *Common Metadata*

Figure 1 shows the common metadata that all subcomponents of GSE share. We define species, genome builds, and experimental metadata that may be shared by ChIP-chip experiments, expression experiments, and ChIP-seq experiments. We represent factors (e.g. an antibody or RNA extraction protocol), cell-types (tissue identifier or cell line name), and conditions as entries in separate tables.

2.2. *Coordinate Independent ChIP-chip Representation*

In our terminology, an experiment aggregates ChIP-chip datasets which all share the same factor, condition, and cell-type as defined in the common metadata tables. Each replicate of an experiment corresponds to a single hybridization performed against a particular micorarray design. In Section 4, we will outline a system for building biological hypotheses out of these descriptive metadata objects.

GSE stores probes separately from their genomic coordinates as shown in Figure 2. Microarray observations are indexed by probe identifier and experiment identifier. The key data retrieval query joins the probe observerations and probe genomic coordinates based on probe identifier and filters the results by experiment identifier (or more typically a set of experiment

Pacific Symposium on Biocomputing 13:539-550(2008)

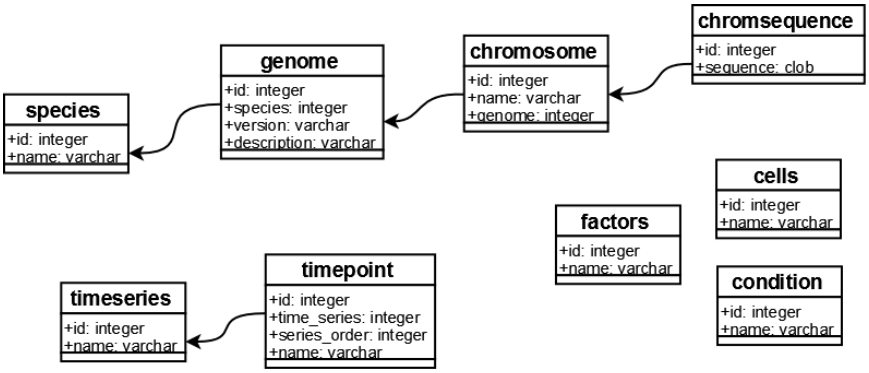


Figure 1. The Genomic Spatial Event database's common metadata defines species, genome assemblies, and terms to describe experiments. *Cells* enumerates the known tissue or cell types. *Conditions* defines the conditions or treatments from which the cells were taken. *Factors* describes antibodies in ChIP-chip experiments or RNA extraction protocols (eg total RNA or polyA RNA) for expression experiments.

identifiers corresponding to replicates of a biological experiment) and genomic coordinate. To add a new genome assembly to the system, we remap each probe to the new coordinate space once and all of the data is then available against that assembly. Since updating to a new genome assembly is a relative quick operation regardless of how many datasets have been loaded, users can always take advantage of the latest genome annotations.

GSE's database system also allows multiple runs of the same biological experiment on different array platforms or designs to be so combined. Some of our analysis methods can cope with the uneven data densities that arise from this combination, and we are able to gather more statistical power from our models when they can do so.

2.3. *Discovering Binding Events from ChIP-chip Data*

Modern, high-resolution tiling microarray data allows detailed analyses that can determine binding event locations accurate to tens of bases. Older low-resolution ChIP-chip microarrays included just one or two probes per gene[9,10]. Traditional analysis applied a simple error model to each probe to produce a bound/not bound call for each gene rather than measurements associated with genomic coordinates[22]. Our Joint Binding Deconvolution (JBD) exploits the dozens or hundreds of probes that cover each gene an intergenic region on modern microarrays with a complex statistical model

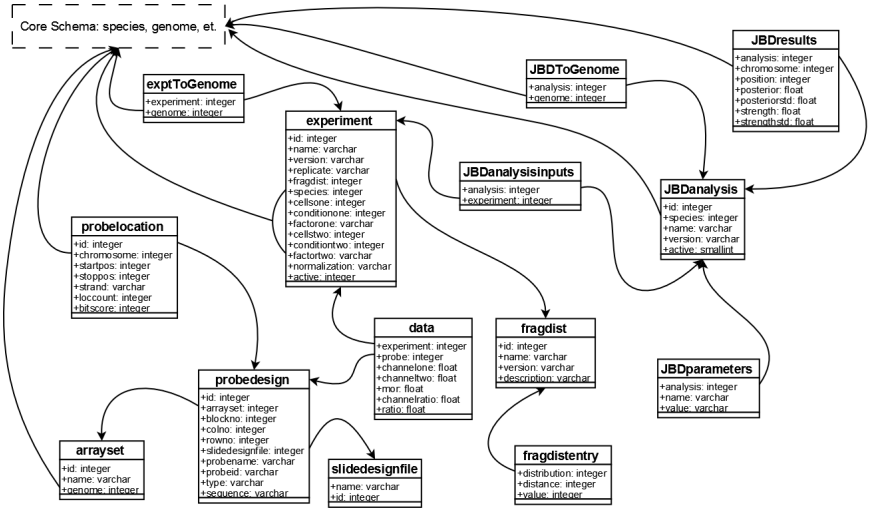


Figure 2. The ChIP-chip schema stores microarray designs, raw microarray observations, and the resulting analyses. We store probe designs as information about a single spot on a microarray. Probes are grouped by slide and by slide sets (arrayset). Genomic coordinates for each probe reside in a separate table to allow easy remapping of probes to new genome assemblies.

that incorporates the results of multiple probes at once and accounts for the possibility of multiple closely-spaced binding events.

JBD produces a probability of binding at any desired resolution (e.g. a per-base probability that a transcription factor bound that location). Figure 2 shows the tables that store the JBD output and figure 3 shows a genomic segment with ChIP-Chip data and JBD results. Unlike the raw probe observations, JBD output refers to a specific genome assembly since the spatial arrangement of the probe observations is a key input. GSE's schema also records which experiments led to which JBD analysis.

2.4. *Prior Work and Performance*

We modeled portions of GSE after several pre-existing analysis and data-handling systems. The core design of an analysis system supported by a relational database was made after experience with the GeneXPress package and its descendant, Genomica[19]. We modeled portions of the GSEBricks system, our modular component analysis system, after the Broad Institute's GenePattern software [18]. There are also several widely-used standards for microarray data storage and annotation databases that we were aware of

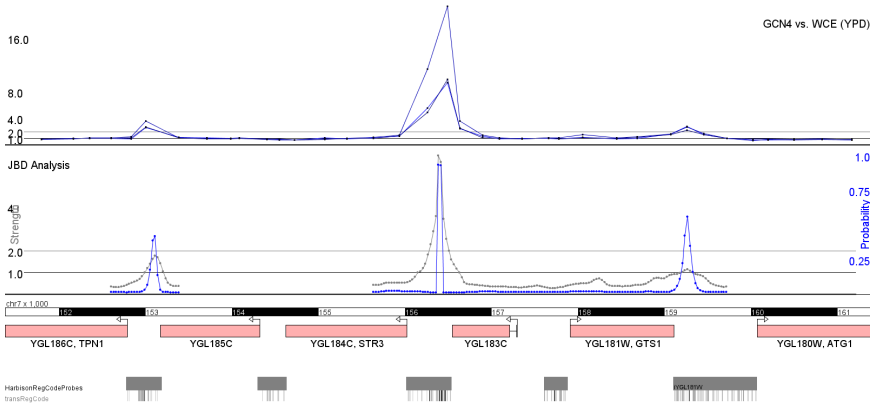


Figure 3. A screenshot from the GSE Visualizer. The top track represents 'raw' high-resolution GCN4 data in yeast, and the bottom track shows two lines for the two output variables of the JBD algorithm. At the bottom are a genomic scale, a representation of gene annotations, and a custom painting of the probes and motifs from the Harbison et. al. Regulatory Code dataset.

when designing our system. For instance, the MIAME standard for microarray information [7] is well-known format and specification for microarray data – however, we made the decision to store significantly less metadata about our ChIP-chip experiments than MIAME requires, since much of it is not immediately useful for biological analysis and it made it harder for our biological collaborators to enter new data into the system. We are also familiar with the DAS system[5], and GSE benefited from close discussions with one of DAS's co-creators, during its design and early implementation. However GSE solves a different problem than DAS, as it is mainly focused on providing a concentrated resource for (often-unpublished) data accumulation and an analysis platform for a small to mid-sized group of researchers.

Measuring the exact performance of a distributed system such as ours is difficult. The system consists of multiple servers running on several heterogeneous platforms, with as many as twenty or thirty regular users. Performance statistics are affected by system load, network latency conditions, and even the complexity of the data itself (the JBD algorithm's runtime is data-dependent, taking longer when the data is more "interesting"). Our group currently runs two database servers, one Oracle and one MySQL, and our computational needs are served by 16 rack-mounted machines with dual 2.2GHz AMD Opteron processors and 4 GB of memory each. We currently

Pacific Symposium on Biocomputing 13:539-550(2008)

store approximately 338 GB of total biological data, which includes 1460 ChIP-chip experiments, 1115 separate results of the JBD algorithm, and over 240 million probe observations. Given this amount of data, and users scattered among at least eight collaborating groups, we are still able to serve up megabase visualizations of most ChIP-chip experiments in a matter of seconds, and to scan single experiments for binding events in times on the order of about 1-2 minutes.

3. GSEBricks: A Modular Library for Biological Data Analysis

GSE's visualization and GUI analysis tools depend on a library of modular analysis and data-retrieval components collectively titled 'GSEBricks'. This system provides a uniform interface to disparate kinds of data: ChIP-chip data, JBD analyses, binding scans, genome annotations, microarray expression data, functional annotations, sequence alignment, orthology information, and sequence motif instances. GSEBricks' components use Java's Iterator interface such that a series of components can be easily connected into analysis pipelines.

A GSEBricks module is written by extending one of three Java interfaces: `Mapper`, `Filter`, or `Expander`. All of these interfaces have an 'execute' method, with a single `Object` argument which is type-parameterized in Java 5. The `Mapper` and `Filter` execute methods have an `Object` (also parameterized) as a return value. `Mapper` produces `Objects` in a one-to-one relationship with its input, while a `Filter` may occasionally return 'null' (that is, no value). The `Expander` execute method, on the other hand, returns an `Iterator` each time it is called (although the `Iterator` may be empty).

3.1. *Ease of Integration and Extensibility*

Each GSEBricks datastream is represented by an `Iterator` object and datastreams are composed using modules which 'glue' existing `Iterators` into new streams. Because we extend the Java `Iterator` interface, the learning curve for GSEBricks is gentle even for novice Java programmers. At the same time, its paradigm of building 'Iterators out of Iterators' lends itself to a Lisp-like method of functional composition, which naturally appeals to many programmers familiar with that language.

Because our analysis components implement common interfaces (eg, `Iterator<Gene>` or `Iterator<BindingEvent>`), it is easy to simply plug

Pacific Symposium on Biocomputing 13:539-550(2008)

them into visualization or analysis software. Furthermore, the modular design lends itself to modular extensions. We have been able to quickly extend our visualizer to handle and display data such as dynamically re-scanned motifs (on a base-by-base level within the visualized region), automatic creation of 'meta-genes'[21] (averaged displays of ChIP-chip data from interactively-selected region sets), and the display of mapped reads from ChIP-PET experiments[13].

The final advantage of GSEBricks is the extensibility of the GSEBricks system itself. By modifying the code we use to glue the Iterators together, we can replace sequential-style list-processing analysis programs with networks of asynchronously-communicating modules that share data over the network while exploiting the parallel processing capabilities of a pre-defined set of available machines.

3.2. *GSEBricks Interface*

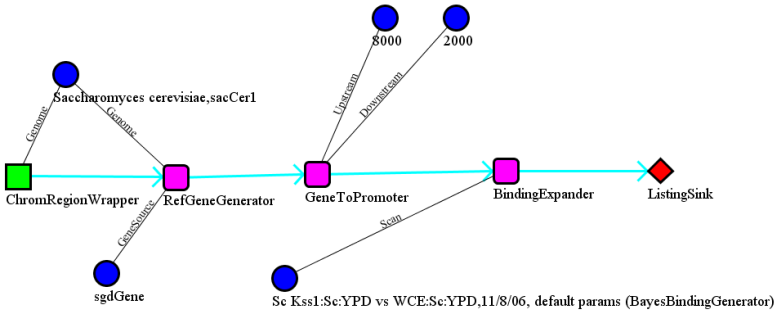
Figure 4 shows a screenshot from our interface to the GSEBricks system. Users can graphically arrange visual components, each corresponding to an underlying GSEBricks class, into structures that represent the flow of computation. This extension also allows non-sequential computational flows – trees, or other non-simply connected structures – to be assembled and computed. The interface uses a dynamic type system to ensure that the workflow connects components in a typesafe manner.

Workflows which can be laid out and run with the graphical interface can also be programmed directly using their native Java interfaces. The second half of Figure 4 gives an example of a code-snippet that performs the same operation using the native GSEBricks components in Java.

4. Representing and Storing ChIP-chip Binding Hypotheses

The final element of the GSE database is a system to store not just raw experimental data but also a representation of a scientist's beliefs about that data.

Investigators often wish to discover the "regulatory networks" of binding that describe transcriptional control in a particular condition or cell type. For a single experiment, the network is simply a set of genes located near high-confidence binding sites[3,16,14]. With multiple experiments, each set of gene targets (the 'network') is characterized by the binding profiles of multiple factors simultaneously. If the investigator is interested in the




```
BindingScanLoader loader = new BindingScanLoader();
Genome sacCer1 = Organism.findGenome("sacCer1");

ChromRegionWrapper chroms = new ChromRegionWrapper(sacCer1);
Iterator chromItr = chroms.execute();

RefGeneGenerator rgg = new RefGeneGenerator(sacCer1, "sgdGene");
Iterator geneItr = new ExpanderIterator(rgg, chromItr);

GeneToPromoter g2p = new GeneToPromoter(8000, 2000);
Iterator promItr = new MapperIterator(g2p, geneItr);

BindingScan kss1 = loader.loadScan(sacCer1, kss1_id);
BindingExpander exp = new BindingExpander(loader, kss1);
Iterator bindingItr = new ExpanderIterator(exp, promItr);

while(bindingItr.hasNext()) {
    System.out.println(bindingItr.next());
}
```

Figure 4. A GSEBricks pipeline to count the genes in a genome. Each box represents a component that maps objects of some input type to a set of output objects. The circles represent constants that parameterize the behavior of the pipeline. The code on the right replicates the same pipeline using Java components.

behavior of those regulating factors, she will need to summarize the behaviors of the regulators across multiple sets of genes[14]. Once a biologist has outlined what she thinks is the "regulatory network" of a collection of factors, she is faced with the problem of formalizing those conclusions in a way that is useful to other scientists, or even to herself at some distant time in the future.

GSE gives the user a simple language to express relationships between different ChIP-chip experiments whose binding events have been precalculated and saved. GSE also provides the user with a schema for storing those

Pacific Symposium on Biocomputing 13:539-550(2008)

hypotheses in the database and for automatically checking those hypotheses against new and unexamined experiments. In this way, we can think of the Hypothesis system as a kind of basic "lab notebook" for the analysis of ChIP-chip binding data.

Our hypotheses, H, have a simple grammar: $F := \{\text{factors}\}$ and $H := F|H \rightarrow H$. We can treat a hypothesis h as a predicate on the set of distinct genomic coordinates, G. If $h = \texttt{F}$, then $h(x)$ if and only if a binding event of F is located at x. We can also relax this condition to include binding "within a certain distance" from one factor to another. The $\rightarrow$ of our hypothesis language is material implication from logic. If $h = \texttt{H1} \rightarrow \texttt{H2}$, then $h(x)$ holds if and only if either $\texttt{H2}(x)$ or $\neg\texttt{H1}(x)$.

We will evaluate hypotheses in reverse- instead of asking how much the data supports a particular hypothesis, we search for examples that contradict the hypothesis. In other words, we treat different (and distant) genomic coordinates as independent witnesses to the validity of a particular hypothesis and we ask how many locations seem to invalidate the hypothesis. The approach is computationally simple because the logical structure of our language will make it easy to quickly evaluate a fixed set of hypotheses against wide regions of genome which have been assayed with large numbers of binding experiments. We will also be able to easily leverage the high-throughput nature of our experiments, which might slow more complex algorithms to an unusable speed. Our approach is also useful because it gives the user a way to systematically enumerate and test the set of exceptions to a hypothesis.

In Table 1, we show the automatic results generated by our Hypothesis system when compared against the Harbison yeast regulatory code dataset[8]. For three factors we report the top ten ranked hypotheses about genes regulated by Fkh2, Rap1, and Ste12. Each column is followed by the number of 'inconsistent' probes that were found by the Hypothesis system. The results are not given a probabilistic interpretation, or even a description beyond just their ranked lists. It is, however, reassuring that such a simple analysis can easily recover most of the known related or interacting factors for these three simple cases[15,20,1].

5. Conclusion

We have described GSE, a system to represent microarray data and metadata in a relational database, and described a software system which reads and presents that data in a modular, extensible way. A reference imple-

	FKH2	#errors	RAP1	#errors	STE12	#errors
0	→ FKH1	82	→ FHL1	131	→ DIG1	63
1	→ NDD1	86	→ GAT3	195	→ TEC1	98
2	→ SWI6	112	→ YAP5	199	→ NDD1	114
3	→ SWI4	114	→ PDR1	201	→ SWI6	115
4	→ MBP1	116	→ SMP1	205	→ MCM1	116

mentation of this system will be available through the Gifford Lab group website, `http://cgs.csail.mit.edu`. This implementation includes an interactive, Java application for visualization and analysis that uses this modular system to browse and view ChIP-chip experiments and genome annotation data.

We have outlined our opinion that the automatic discovery of regulatory relationships from databases like GSE can only occur when the database itself stores hypotheses about the data. We have sketched a rudimentary hypothesis system which can automatically read simple hypotheses from the GSE database and check them in a non-probabilistic way against precomputed binding event scans.

In the near future, we will extend our system to handle new kinds of large-scale ChIP-based data. Specifically, we are developing a schema and a set of GSEBricks components to efficiently handle the multi-terabyte datasets we expect to receive from new ChIPSeq machines[11].

References

1. Ziv Bar-Joseph and Georg et al. Gerber. Computational discovery of gene modules and regulatory networks. *Nature Biotechnology*, 21:1337–1342, October 2003.
2. DA Benson, I Karsch-Mizrachi, DJ Lipman, J Ostell, and DL Wheeler. Genbank. *Nucleic Acids Research*, 35:21–25, January 2007.
3. LA Boyer, TI Lee, MF Cole, SE Johnstone, SS Levine, JP Zucker, MG Guenther, RM Kumar, HL Murray, RG Jenner, DK Gifford, DA Melton, R Jaenisch, and RA Young. Core transcriptional regulatory circuitry in human embryonic stem cells. *Cell*, 122(6):947–956, September 2005.
4. MA Crosby, JL Goodman, VB Strelets, P Zhang, WM Gelbart, and Flybase Consortium. Flybase: genomes by the dozen. *Nucleic Acids Research*, 35:486–491, 2007.
5. R. Dowell, R. Jokerst, A. Day, S. Eddy, and L. Stein. The distributed annotation system. *BMC Bioinformatics*, 2, Oct 2001. 10.1186/1471-2105-2-7.
6. SS et. al. Dwight. Saccharomyces genome database: underlying principles and organisation. *Brief Bioinformatics*, 5(1):9–22, Mar 2004.
7. Brazma et. al. Minimum information about a microarray experiment (mi-

Pacific Symposium on Biocomputing 13:539-550(2008)

ame)[mdash]toward standards for microarray data. *Nature Genetics*, 29:365–371, Dec 2001. 10.1038/ng1201-365.
8. Harbison et al. Transcriptional regulatory code of a eukaryotic genome. *Nature*, 431:99–104, September 2004.
9. Lee et al. Transcriptional regulatory networks in *s*accharomyces cerevisiae. *Science*, 298:799–804, October 2002.
10. Ren et al. Genome-wide location and function of dna binding proteins. *Science*, 290:2306–2309, December 2000.
11. David S. Johnson, Ali Mortazavi, Richard M. Myers, and Barbara Wold. Genome-wide mapping of in vivo protein-dna interactions. *Science*, 316(5830):1497–1502, 2007.
12. D Karolchik, R Baertsch, M Diekhans, TS Furey, A Hinrichs, YT Lu, KM Roskin, M Schwartz, CW Sugnet, DJ Thomas, RJ Weber, D Haussler, and WJ Kent. The ucsc genome browser database. *Nucleic Acids Research*, 31(1):51–54, 2003.
13. YH et. al Loh. The oct4 and nanog transcription network regulates pluripotency in mouse embryonic stem cells. *Nature Genetics*, 38:431–440, March 2006.
14. DT Odom, RD Dowell, ES Jacobsen, W Gordon, TW Danford, KD MacIsaac, PA Rolfe, CM Conboy, DK Gifford, and E Fraenkel. Tissue-specific transcriptional regulation has diverged significantly between human and mouse. *Nature Genetics*, 39:730–732, 2007.
15. Yitzhak Pilpel, Priya Sudarsanam, and George M. Church. Identifying regulatory networks by combinatorial analysis of promoter elements. *Nature Genetics*, 29:153–159, September 2001.
16. Dmitry K. Pokholok, Julia Zeitlinger, Nancy M. Hannett, David B. Reynolds, and Richard A. Young. Activated signal transduction kinases frequently occupy target genomes. *Science*, 313:533–536, July 2006.
17. Yuan Qi, Alex Rolfe, Kenzie MacIsaac, Georg Gerber, Dmitry Pokholok, Julia Zeitlinger, Timothy Danford, Robin Dowell, Ernest Fraenkel, Tommi Jaakkola, Richard Young, and David Gifford. High-resolution computational models of genome binding events. *Nature Biotechnology*, 24(8):963–970, August 2006.
18. M. Reich, Liefeld T., J. Gould, J. Lerner, P. Tamayo, and J.P. Mesirov. Genepattern 2.0. *Nature Genetics*, pages 500–501, 2006.
19. E. Segal, R. Yelensky, A. Kaushal, T. Pham, A. Regev, D. Koller, and N. Friedman. Genexpress: A visualization and statistical analysis tool for gene expression and sequence data. 2004.
20. Priya Sudarsanam, Yitzhak Pilpel, and George M. Church. Genome-wide cooccurrence of promoter elements reveals a cis-regulatory cassette of rrna transcription motifs in *s*accharomyces cerevisiae. *Genome Research*, 12(11):1723–1731, November 2002.
21. F.C. Wardle and D.T. et al Odom. Zebrafish promoter microarrays identify actively transcribed embryonic genes. *Genome Biology*, 7(R71), August 2006.
22. L Weng, H Dai, Y Zhan, Y He, S Stepaniants, and D Bassett. Rosetta error model for gene expression analysis. *Bioinformatics*, 22(9):1111–1121, 2006.