Supporting Materials

Genetic Regulatory Network of Modules (Geronemo). As discussed in the paper, the Geronemo learning algorithm iterates over two phases: (1) assigning each gene into some regulatory module; (2) learning the regulation program for each module. At a high level, our method follows the module network approach of Segal *et al*. (1). However, as discussed in the paper, we introduced several important innovations that are keys to the success of the method in this new setting. Below, we review both the module network procedure and our extensions.

(1) Regulation programs. We adopted the concept of the regulation program from Segal *et al*. (1). Briefly reviewing, a regulation program of a gene *G* specifies a set of contexts and *G*'s expression values in each context. A context is determined by the qualitative behavior of a small set of regulatory factors that control *G*'s expression and includes a model for *G*'s behavior in that context. This set of contexts is organized as a regression tree in which each path to a leaf in the tree defines a context by using the tests on the path. A regression tree is composed of two basic building blocks: decision nodes and leaf nodes. Each decision node corresponds to one of the regulatory inputs and a query on its value (for example, "is Hap1 up-regulated?"). Each decision node has two child nodes: the right child node is chosen when the answer to the corresponding query is true; the left node is chosen when it is false. For a gene in a given array, one begins at the root node and continues down the tree in a path according to the answers to the queries in that particular array. Thus, each leaf (context) corresponds to the arrays that traverse the same path through the tree. The expression level of *G* in each context is modeled as a normal distribution; this distribution is encoded using a mean and variance stored at the corresponding leaf. The model semantics is that, given a gene *G* and an array *A* in a context, the probability of observing some expression value for gene *G* in an array *A* is governed by the normal distribution specified for that context. For a context in which the expression values are tightly regulated, the distribution may have a small variance; in a context where the expression values are not tightly regulated, the distribution may have a large variance. In the current study, we allowed two types of regulatory factors: the genotype of some chromosomal region defined by a marker, which (in our data) has two possible split values, for the two progenitor alleles; and the expression level of some regulator *R*, whose set of possible splits is the number of different continuous values that the regulator takes on in the expression arrays.

(2) Compilation of candidate regulator set. Our candidate regulator set included two different types of regulators: genotype regulators and expression regulators. As expression regulators, we compiled a list of regulators that potentially might have transcriptional role in the broad sense, including: transcription factors, signaling molecules, chromatin modification factors, and RNA factors (degradation and RNA processing). The list was derived by using Gene Ontology annotations in SGD (2) and further corrected via manual curation. We filtered the list to keep only those candidate regulators showing a significant variation in their expression levels (>90% of the expression values are present and standard

deviation > 0.25), which resulted in 304 regulators (see Table 1). For each query node in the regulation programs, the learning algorithm can choose any candidate gene expression regulator *R* and construct a query "Is *R*'s expression level > *t*?" for any threshold of *R*'s expression *t*. We used the genotype data consisting of the genotype values (BY/RM) for each of 112 recombinants, measured in 2,957 genetic markers distributed over the genomes. We merged into a single regulator adjacent, highly correlated markers (which disagree on genotype values in <5 segregants), resulting in 581 merged markers. For each query node in the regulation programs, the learning algorithm can choose any candidate genotype regulator *R* and construct a query "Is *R*'s genotype BY?".

(3) Geronemo learning algorithm. Learning a Geronemo model involves two tasks: (i) assigning each gene into some regulatory module; (ii) learning the regulation program for each module. We first construct 500 modules obtained by *k*-means clustering. We then iterated over two phases: learning the regulatory program for the current modules, and reassigning genes to modules. In our learning procedure, we prevented genes from regulating the same module in which they appeared, but did not require global acyclicity because the real regulatory network is not acyclic. Our two-step iterative learning procedure attempts to search for the model with the highest score, in each step optimizing one of the model's components: regulation programs or gene partition. An important property of our algorithm is that each iteration is guaranteed to improve the model score, until convergence to a local maximum of the score. We now elaborate on each of the components of our algorithm.

(i) Initialization of modules. Geronemo learning procedure starts with initial modules obtained from *k*-means clustering algorithm, which initializes the clusters with *k* random points. Below, we present results testing the sensitivity of our learning procedure to the number of initial modules (*k*) and to different sets of random initial points of *k*-means clustering.

(ii) Learning the regulatory program. Given a set of modules, we learned a regulatory association for each module by using the candidate expression (e) and genotype (g) regulators. As described above, the association between the genetic regulators and the expression of genes can be represented as a regression tree. The regulation program is learned via a combinatorial search over the space of trees. The tree is grown from the root to its leaves by using a recursive procedure. At any step, the algorithm considers splitting a node that is currently a leaf, by evaluating each possible query at that leaf. We consider all candidate regulators and split values. The different possible queries are evaluated relative to the extent to which they increase the model score, and subsequently accepted or rejected by using an FDR test. A query that is chosen splits the leaf into two distinct contexts, and then the process repeats. The process terminates when no query that improves the score can be found.

Importantly, the candidate genetic regulators for Geronemo consist of both continuous-valued e regulators and discrete-valued g regulators. The continuous-valued candidate

regulators have more possible split values than the discrete-valued ones; therefore, it is possible that e regulators can be captured more often than g regulators because of the spurious correlation with the genes in the module. We extended our learning scheme to correct for these effects. When considering adding a decision node to the regulatory program, we take the following steps. (i) We first choose the e-regulator that achieves the highest score among all candidate e regulators, and the g regulator with the highest score among all candidate g regulators. (ii) For each of the highest-scoring e and g regulator, we calculate a *P* value by using false discovery rate (FDR); see below. (iii) We select either the e or g regulator with the lower FDR. If the chosen FDR is >0.05, we select the next highest scoring regulator, and repeat. If the top three regulators in e and g regulator groups have FDR >0.05, the process terminates. Evaluating the split by using FDR serves two purposes. First, it ensures only statistically significant splits are included in the regulation tree. Second, it ensures that an expression regulator is chosen only when it is significantly more predictive of its targets than the best genotype regulator.

We performed the FDR test as follows: for the regulator being tested, we generate 1,000 permutations, randomly permuting the values of the regulator across individuals. Thus, the composition of values for the regulator remains unchanged, but the order is random and independent of the target genes. We calculated the score for each of the permutations, for g regulators based in the two discrete values and for the e regulators based on the highest scoring split value. The FDR is the frequency with which the score is higher than the original one among the 1,000 random permutations. To ensure that it is the expression value of an e regulator which is predictive, and not its genotype, we permute its expression values within each of its genotypes separately. This ensures that if there is a different distribution of expression values between the two genotypes, the randomization preserves this difference.

(iii) Learning the module assignment. Given the inferred regulation programs, we determine the module whose associated regulation program best predicts each gene's behavior. Specifically, we iterate over all genes one at a time, for each gene *g*, computing the change in score of the model obtained by moving *g* from its current module, into every other possible module *M*. We then move *g* into the module *M* that provides the highest improvement in the score. However, we take care not to assign a regulator gene to a module in which it is also a regulatory input, as it is not surprising that a gene can predict its own expression. This step is guaranteed to improve the score, or leave it the same (if the gene is not moved). We repeated this reassignment process for all genes three times. Importantly, after this step, some modules become empty, at which point they are removed from the algorithm.

To allow for cis-regulation effects, in which a polymorphism in the regulatory region of a gene changes the expression of that gene, we introduced a step that allows genes to “break off” from their module and create a new one. When the module assignment is stabilized to some extent (when the number of genes moved <200), we considered a singleton decision step for each gene $g_{i,}$. In this step, we consider whether g_i might be better described as a cis

linked gene, as follows: Let M_j be the current module of g_i; we consider breaking M_j into two modules – a module S_i containing only g_i, and a module M_j' containing the rest of the genes in M_j. To avoid over-fitting, in the post-split model, the regulation trees of S_i and M_j' (required for computing the score) are learned using only the regulators and their split values used in the regulation program of M_j, and, for S_i, also allowing g_i's genotype as a parent. We then accept the new split only if it improves the overall score. After this singleton decision step is considered for all genes and the new singleton modules are formed, we repeat our module reassignment step. After the module reassignment process is completed, our model generally has a different number of modules.

Although many of the search steps in this algorithm are heuristic (generally inevitable when searching over a large combinatorial space), the algorithm is nevertheless principled, as it is consistently optimizing a wellfounded, Bayesian scoring function (see below). The key to the performance of this algorithm is its sequential nature, which guarantees that each step taken can only improve the score or leave it unchanged. Our algorithm terminates when it can no longer improve the score. Due to the complexity of the likelihood landscape, our search does not guarantee a global maximum; it converges to a local maximum, where no single assignment change can improve the score.

(iv) Score function. We use a wellfounded Bayesian scoring approach, which tries to maximize the overall joint probability of both the data and of the model structure. Let S represent the "structure" of a model, which includes the assignment of genes into modules (denoted by A) and the regulation trees of the modules (denoted by T). The joint log-likelihood of the model can be expressed as $\log P(D|S) + \log P(S)$, where D represents the data. This score consists of two parts: (i) $\log P(D|S)$: the log of the data marginal-likelihood given the structure, and (ii) $P(S)$: the prior probability distribution of the model structure.

We use a Bayesian approach for computing the data log-likelihood term. We model the prior distribution over the parameters of the structure S (denoted by θ_S) by using a normal-gamma distribution. Here, θ_S means the random variable representing the expression level in each context of the regulation program. Therefore, the data log-likelihood can be computed as:
$\log \mathrm{P}(\mathrm{D} \mid \mathrm{S}) = \log \int_{\theta_S} \mathrm{P}(D \mid S, \theta_S) P(\theta_S) d\theta_S$, where $P(\theta_S)$ is the probability density function (pdf) of normal-gamma distribution.

We consider two kinds of prior distribution for S: a general complexity prior on a regulation program T and a biologically motivated prior on S. The complexity prior penalizes the number of regulators (or number of leaves) in a regulation program. We use the exponential distribution over the total number of leaves (denoted by L_T) in a regulation tree T, which leads to $\log \mathrm{P}(\mathrm{T}) = -\beta L_T$. We also consider a prior for S that can bias the model in favor of more biologically plausible regulation programs, which imposes a sparse prior (*i*) on the number of targets of each regulator and (*ii*) on the number of distinct split

values that a regulator has. The sparse prior on *i* encourages the algorithm to choose a small set of candidate regulators as expression regulators, thereby avoiding over-fitting when there are a number of candidate regulators many of which are not relevant. The prior on *ii* encourages the expression regulators to have small number of split points in the regulation program, which can reduce overfitting caused by the flexibility of choosing the split point of the expression regulators. This prior takes the form of a power-law distribution on the number of trans-targets of a regulator *r*, and on pairs of *r* and its split value v_r. Denoting by *R* all regulators,

$$\log P(S) = -x\sum_{\{r,v\}\text{ for } r\in R, v\in v_r} \log\{w(r,v)+1\} - y\sum_{r\in R}\log\{w(r)+1\},$$

where *w(R,V)* represents the number of trans-regulated targets of the regulator *R* with the split value *V*, and $w(R) = \sum_v w(R,v)$.

Overall, the scoring function is expressed as:

$$\log P(D,S) = \log P(D\mid S) + \log P(S)$$

$$= \log \int_{\theta_S} P(D\mid S,\theta_S)P(\theta_S)d\theta_S - \beta\sum_T L_T$$
$$- x\sum_{\{r,v\}} \log\{w(r,v)+1\} - y\sum_r \log\{w(r)+1\}$$

The parameters of the score (β, *x* and *y*) were determined by optimizing the log-likelihood of test data through 5-fold cross-validation test.

(v) Module dynamics and stop condition. In summary, each iteration of the Geronemo learning procedure performs one of the three operations: (i) learning the regulatory programs, (ii) learning module assignment and (iii) learning module assignment with singleton decision step. Basically, Geronemo runs i once and then ii three times. Once we achieve approximate convergence, so that each of the three consecutive runs of ii moves <200 genes, it performs iii. It iterates this process (a global iteration) until convergence. The stop condition is defined as: the sum of the number of genes moved in ii or iii in one global iteration is <10% of the total number of genes. The number of genes that change their module assignment, and the score in each iteration is illustrated in Fig. 5.

(vi) Final set of modules. Applied to our data set, Geronemo converged to a set of 198 regulatory modules. However, not all genes were wellexplained by the resulting model. Specifically, a small number of modules did not represent a coherent expression profile, indicating that they did not provide a good explanation of the data, and therefore were less likely to represent true biological relationships. We defined the coherence of a module as the average Bayesian score of the genes in the module (averaged over the number of genes). Based on this module coherence score, we filtered out the 33 least coherent modules –

those whose average score per gene was lowest, resulting in 165 modules that we subsequently evaluated.

(vii) Robustness of Geronemo to initialization. As the initialization of Geronemo is done from the output of a randomly initialized *k*-means clustering, we tested the robustness of the results obtained both to the value of *k* and to the randomly selected starting point for *k*-means. To test sensitivity to the number of initial modules *k, w*e initialized Geronemo by using various values of *k* values ranging from 100 to 900. Fig. 10 presents the results of this evaluation.

While the final number of modules in the module network procedure shows obvious dependence on *k*, Geronemo exhibits very little sensitivity to *k*, especially when *k*>400. For lower values of *k*, Geronemo also shows more invariance to *k* than module network, which is mainly due to the module breaking procedure. We conclude that the Geronemo learning procedure that actively merges and breaks a module makes it possible to automatically determine the number of final modules if it starts with enough number of initial modules. We chose *k*=500 for the analysis in the paper.

We then evaluated the sensitivity of our learning procedure to the random initialization of *k*-means clustering. Fixing *k* = 500, we generated 100 different sets of initial modules from 100 different initial points of *k*-means clustering and compared each of the 100 resulting models with the model analyzed in the paper. To perform the comparison, we first needed to map modules in two runs, one of the 100 models from the new runs and our target model, analyzed in the paper. This mapping was defined by mapping each module *M* in a new run to a module *M'* in the target run that contained >50% of the genes in *M* (if one existed). Then, we computed the fraction of genes in a new run that are assigned to the corresponding module in the target model. Among 100 runs, the fraction of genes in a new run that are assigned to the corresponding module in the analyzed model ranges from 66.18% to 71.83% (mean = 68.85%, stdev = 0.974). As a control experiment, we tried 100 sets of clusters from 100 different runs of *k*-means clustering. In this control experiment, the fraction ranges from 39.94% to 47.36% (mean = 43.9%, stdev = 2.8). We also computed the fraction of cis-genes in a new run that are also cis-genes in the analyzed model. The fraction ranges from 78.52% to 91.6% (mean = 83.4%, stdev = 2.327) among 100 models. Overall, our results indicate that the results obtained by Geronemo are very robust to the initial starting point.

Enrichment Analysis for Number of Polymorphisms in a Gene Group. As discussed in the paper, we applied an enrichment analysis of polymorphisms for several groups of genes: cis-genes, regulators in the vicinity of markers selected as parents, and the genes in the Swi/Snf complex. For coding sequences, we evaluated enrichment for nonsynonymous SNPs, and for promoter regions, we evaluated enrichment for any SNPs, taking the regions to include the 500 bp upstream of the transcription start site. In either case, we computed enrichment of the polymorphisms in a gene group of interest (case

group) by comparing the distribution of the number of polymorphisms between the group and a control group. The control group consisted of the genes that are adjacent (one upstream and one downstream) to the genes in the group of interest. The distribution of the number of polymorphisms is far from any standard distribution used in general statistical hypothesis tests. Therefore, we devised a nonparametric FDR correction method that evaluates significance by using 1,000 permutation tests. In each case group *S*, we computed the hyper-geometric enrichment of the case group, relative to the control group, for genes that contain >*n* polymorphisms. We varied *n* such that the *P* value based on the hyper-geometric distribution is minimized. If such minimum *P* value is smaller than 0.01, we performed 10,000 permutation tests to verify the significance. In each permutation test *i*, we chose a "random case group" S_i with the same number of genes as *S*, and defined the control group to be the genes adjacent to S_i. We then varied the cutoff *n* in the same way, and calculated the *P* value. The FDR-corrected P value is the fraction of permutation tests where the P value of the random case group S_i is smaller than that of *S*.

Proportion of Genetic Variance Explained by Genetic Regulators. We estimated the percentage of genetic variance (PGV) explained by the identified genetic regulators, using the same method used by Brem and Kruglyak (3) in analyzing the eQTL approach. We randomly divided the data of 112 segregants into two sets of the same size (56 segregants): the detection set and the estimation set. Using the detection set, we identified the regulation program through our procedure (Fig. 6). Then, we used the estimation set to calculate the PGV on the basis of the identified regulation programs. The PGV of a gene *G* is defined as

$$\mathrm{PGV}_G = \frac{R^2_{\mathrm{adj}}(\mathrm{e}_G, \mathrm{g}_G)}{\hat{\mathrm{h}}^2_G} \times 100 .$$

The denominator $\hat{h}^2_G$ is the heritability of *G*, defined as $\hat{h}^2 = (\sigma_s^2 - \sigma_p^2)/\sigma_s^2$, where σ_s^2 and σ_p^2 are the variance among its expression values in the segregants and its pooled variance among parental measurements, respectively. The numerator $R^2_{adj}(e_G, g_G)$ is the adjusted squared correlation coefficient between the expression values of *G* in the estimation set (e_G) and the expression values for *G* by using a single-factor ANOVA, where the factors are the contexts in the regulation program learned from the detection set (g_G). Specifically, the predicted expression values are computed as follows: for each context (leaf) in the regulation tree, we compute the average expression, in the estimation set, over all of the arrays in the context and all of the genes in the module. The adjusted squared correlation coefficient is now defined using the correction described by Utz *et al*. (4):

$$R^2_{adj}(e_G, g_G) = R^2 - \left(\frac{z}{N - z - 1}\right)(1 - R^2),$$

where R^2 is the square correlation coefficient between e_G and g_G, *z* is the number of contexts of the regulation program, and *N* is the number of expression values in the estimation set, which is the product of the number of individuals in the estimation set times the number of genes in the module. There are significant differences between the complexity of the Geronemo model and of the eQTL model: On the one hand, the Geronemo model allows more parameters per regulatory program, by allowing combinatorial regulation and therefore more than two contexts. On the other hand, the module-based approach forces sharing of parameters between all of the genes in a module. The formula for R^2_{adj} corrects both for the number of contexts in the regulatory programs (by the *z* term) and for the use of module-based models (by the *N* term). We note that, overall, the Geronemo model had significantly fewer parameters: an average of <0.6 contexts per linked gene, compared to the standard 2 contexts per linked gene found in eQTL.

We repeated this detection/estimation process 10 times with different random splits of data. As in the work of Brem and Kruglyak (3), if the PGV was <0, we assigned it to be identically 0, if it was >1, we excluded it; genes with estimated heritabilities <0 were also excluded. We computed the PGV_g for each gene by taking the average of PGV of the gene over 10 runs. If the gene is not assigned a regulatory program in a run, we considered its PGV in that run to be 0. In Fig. 1, we plotted PGV_g (*y* axis) of 3,152 genes (*x* axis) used in our analysis and sorted by their PGV_gs. We also plotted the result reported by Brem and Kruglyak (3) who conducted the same experiment to calculate the PGVs in 10 runs, each of which detected a linkage for 1,038 genes on average, and presented the fraction of PGVs in the given ranges (red boxes in Fig. 1).

Validation relative to additional data sources. We evaluated the biological relevance of each regulatory module in the context of other genomic data sets. We used the hypergeometric *P* value to test for functional enrichment of 1,601 GO categories taken from SGD (5). To associate transcription factors (and sometimes their known upstream signaling molecule) to targets, we used ChIP binding data for transcription factors (6) and chromatin modifying factors (7, 8) and used a $P < 0.001$ cutoff to define a target set for each regulator. For the Isw2 protein (8), we considered the genes whose $\log_2$ ratio of IP/input >1 to be Isw2 targets. We also used a map of putative TF binding sites (motifs) for 65 motifs from the Fraenkel lab web site (6). We took the union of binding sites conserved in at least two other yeast species and binding sites that match the Harbison *et al.* (6) binding data. We used the resulting motif map both to test for enrichment in our modules and for the actual location of the predicted binding site to mark SNPs potentially perturbing a transcription factor binding site. As an additional resource for associating regulators to (not necessarily direct) targets we used differentially expressed genes (DEGs) from deletion mutants of different regulators in (9-11). For each deletion we defined a threshold for differentially expressed as described in (12). For the full set of annotations (function, binding data and DEG targets), we removed all annotations associated with less than five genes from our gene set. For each module and each annotation, we counted the number of predicted target genes associated with that particular annotation, and calculated a *P* value by using the hypergeometric distribution. We carried out a FDR correction for multiple independent hypotheses and took values of $P_{\text{corrected}} < 0.005$ to be significant.

Identifying Orthologous Genes Between BY and RM. We downloaded the genome sequences of S288C (isogenic to BY) and RM from the *Saccharomyces* Genome Database (5) and Broad Institute of Fungal Genome Initiative (www.broad.mit.edu), respectively (sequences were retrieved on 12 January 2005). In order to define orthologous genes between BY and RM, we used reciprocal best BLAST hit (13) (protein sequences of S288C were download from SGD (5) on 12 January 2005). We retrieved the genomic sequences, 500 bp upstream of each orthologous pair, and aligned them by using LAGAN (14). Out of 6,683 genes in 16 nuclear chromosomes, 6,292 (94.1493%) have reciprocal best matches between the two strains.

Positive Selection Test for Regulatory GO Categories. We checked the enrichment for the genes that are likely to be under positive selection in 38 GO categories related to various regulatory roles (see list in table below). We collected 1,097 genes with dN>dS from the dN/dS test (15) (listed in Table 3) in these 38 regulatory categories. For each category, we computed the *P* values representing the significance of the overlap between the genes with dN>dS and the genes belonging to the category. We sorted the categories by *P* value, as listed below. As we discussed in the paper, the Swi/Snf complex is the only category with the *P* value <0.01.

Gene Ontology category	***P* value**	**No. of genes with dN>dS in (i)**	**(i) GO genes**	**No. of genes with dN>dS in (ii)**	**(ii) Others**
SWI/SNF complex	0.001471	6	11	1091	5726
Regulation of meiosis	0.015522	5	12	1092	5725
Regulation of transcription from RNA polymerase II promoter	0.015599	20	69	1077	5668
Specific RNA polymerase II transcription factor activity	0.037232	12	41	1085	5696
Chromatin remodeling complex	0.04219	4	11	1093	5726
Histone acetyltransferase complex	0.050093	5	15	1092	5722
Transcriptional activator activity	0.069418	9	32	1088	5705
Rho protein signal transduction	0.088299	5	17	1092	5720
Protein tyrosine phosphatase activity	0.105621	3	10	1094	5728
SAGA complex	0.111913	5	18	1092	5719
Specific transcriptional repressor activity	0.141865	3	11	1094	5726
H4/H2A histone acetyltransferase complex	0.141865	3	11	1094	5726
Histone deacetylase complex	0.211988	4	17	1093	5720
Chromatin remodeling	0.23467	11	50	1086	5687
Protein serine/threonine kinase activity	0.287139	7	33	1090	5704
Transcriptional repressor activity	0.295709	2	10	1095	5727
Regulation of transcription, DNA-dependent	0.311781	6	29	1091	5708
Protein kinase activity	0.327968	10	49	1087	5688
General RNA polymerase II transcription factor activity	0.3505	8	40	1089	5697
Chromatin silencing at telomere	0.381095	8	41	1089	5696
Regulation of cyclin dependent protein kinase activity	1	2	14	1095	5723
Chromatin silencing at ribosomal DNA	1	0	12	1097	5725
Transcription factor activity	1	7	57	1090	5680
RNA polymerase II transcription factor activity	1	1	17	1096	5720
Transcription coactivator activity	1	1	11	1096	5726
Transcription corepressor activity	1	3	16	1094	5721
GTPase activity	1	10	57	1088	5681
Histone acetyltransferase activity	1	1	14	1096	5723
Histone deacetylase activity	1	2	11	1095	5726
Signal transducer activity	1	3	23	1094	5714

DNA-directed RNA polymerase II, core complex	1	1	11	1096	5726
Chromatin silencing	1	1	16	1096	5721
Transcription	1	3	29	1094	5708
Transcription from RNA polymerase II promoter	1	4	40	1093	5697
Signal transduction	1	5	36	1092	5701
Small GTPase mediated signal transduction	1	4	21	1094	5717
Ras protein signal transduction	1	1	14	1096	5723
Chromatin modification	1	3	19	1094	5718

References

1. Segal, E., Shapira, M., Regev, A., Pe'er, D., Botstein, D., Koller, D. & Friedman, N. (2003) *Nat. Genet.* **34,** 166-176.
2. Cherry, J. M., Ball, C., Weng, S., Juvik, G., Schmidt, R., Adler, C., Dunn, B., Dwight, S., Riles, L., Mortimer, R. K. & Botstein, D. (1997) *Nature* **387,** 67-73.
3. Brem, R. B. & Kruglyak, L. (2005) *Proc. Natl. Acad. Sci. USA* **102,** 1572-1577.
4. Utz, H. F., Melchinger, A. E. & Schon, C. C. (2000) *Genetics* **154,** 1839-1849.
5. Balakrishnan, R., Christie, K. R., Costanzo, M. C., Dolinski, K., Dwight, S. S., Engel, S. R., Fisk, D. G., Hirschman, J. E., Hong, E. L., Nash, R., Oughtred, R., Skrzypek, M., Theesfeld, C. L., Binkley, G., Dong, Q., Lane, C., Sethuraman, A., Weng, S., Botstein, D. & Cherry, J. M. (2005) *Nucleic Acids Res.* **33,** D374-D377.
6. Harbison, C. T., Gordon, D. B., Lee, T. I., Rinaldi, N. J., Macisaac, K. D., Danford, T. W., Hannett, N. M., Tagne, J. B., Reynolds, D. B., Yoo, J., Jennings, E. G., Zeitlinger, J., Pokholok, D. K., Kellis, M., Rolfe, P. A., Takusagawa, K. T., Lander, E. S., Gifford, D. K., Fraenkel, E. & Young, R. A. (2004) *Nature* **431,** 99-104.
7. Lieb, J. D., Liu, X., Botstein, D. & Brown, P. O. (2001) *Nat. Genet.* **28,** 327-334.
8. Gelbart, M. E., Bachman, N., Delrow, J., Boeke, J. D. & Tsukiyama, T. (2005) *Genes Dev.* **19,** 942-954.
9. Bernstein, B. E., Tong, J. K. & Schreiber, S. L. (2000) *Proc. Natl. Acad. Sci. USA* **97,** 13708-13713.
10. Hughes, T. R., Marton, M. J., Jones, A. R., Roberts, C. J., Stoughton, R., Armour, C. D., Bennett, H. A., Coffey, E., Dai, H., He, Y. D., Kidd, M. J., King, A. M., Meyer, M. R., Slade, D., Lum, P. Y., Stepaniants, S. B., Shoemaker, D. D., Gachotte, D., Chakraburtty, K., Simon, J., Bard, M. & Friend, S. H. (2000) *Cell* **102,** 109-126.
11. Sudarsanam, P., Iyer, V. R., Brown, P. O. & Winston, F. (2000) *Proc. Natl. Acad. Sci. USA* **97,** 3364-3369.
12. Pe'er, D., Regev, A., Elidan, G. & Friedman, N. (2001) *Bioinformatics* **17**, Suppl 1**,** S215-S224.
13. Altschul, S. F., Madden, T. L., Schaffer, A. A., Zhang, J., Zhang, Z., Miller, W. & Lipman, D. J. (1997) *Nucleic Acids Res.* **25,** 3389-3402.
14. Brudno, M., Do, C. B., Cooper, G. M., Kim, M. F., Davydov, E., Green, E. D., Sidow, A. & Batzoglou, S. (2003) *Genome Res.* **13,** 721-731.
15. Nei, M. & Gojobori, T. (1986) *Mol. Biol. Evol.* **3,** 418-426.
16. Hereford, L. M., Osley, M. A., Ludwig, T. R., 2nd, & McLaughlin, C. S. (1981) *Cell* **24,** 367-375.