CHAPTER 18

Computational Approaches to Finding and Analyzing *cis*-Regulatory Elements

C. Titus Brown

Beckman Institute 139-74
Caltech
Pasadena CA 91125

Abstract

The transcription of almost all developmental genes is driven by tissue- and time-specific regulatory elements. These transcriptional regulatory elements lie in the genomic DNA proximal to the gene, and hence are *cis*-regulatory (as opposed

METHODS IN CELL BIOLOGY, VOL. 87

0091-679X/08 $35.00
DOI: 10.1016/S0091-679X(08)00218-5

to *trans*-regulatory elements like transcription factor genes). Over the past three decades, a number of techniques have been applied to the problem of finding and characterizing these regulatory elements. In this chapter, I discuss some computational approaches that have been particularly useful in identifying developmental *cis*-regulatory regions, and provide a tutorial on how to apply these approaches to the study of chick development.

I. Structure and Function of *cis*-Regulatory Elements

Over the last two decades, a consistent view of the structure of *cis*-regulatory elements has emerged from work in *Drosophila*, sea urchins, and mouse (Davidson, 2006). *cis*-Regulatory apparatus can be spread over hundreds of kilobases of genomic DNA, and even several megabases; this apparatus largely consists of a number of relatively small (300 bp–3 kb) modules of regulatory DNA. Each module contributes to the tissue-specific expression of at least one gene, regulating it either positively or negatively. Modules are often capable of acting independently: that is, an individual module can direct or repress expression of a reporter construct independently of the presence of other regulatory modules. Thus the most common method of testing the *cis*-regulatory function of a region of DNA is to link it to a reporter gene encoding either β-galactosidase (LacZ) or a fluorescent protein, make a transgenic animal or line containing the reporter construct, and then look for reporter activity *in vivo*.

While relatively few regulatory modules have been analyzed in exhaustive detail, those that have been analyzed yield a consistent picture of their internal structure (Davidson, 2006). Each regulatory module contains several – from two to dozens – binding sites for transcription factors. Some of these binding sites bind tissue-specific positive or negative regulators that respond, in turn, to their own regulatory apparatus. Others bind ubiquitous transcription factors that modify DNA structure or link regulatory elements to the basal transcription apparatus.

The ultimate goal of *cis*-regulatory analysis is to understand precisely how the transcription of a particular gene is controlled by upstream factors, and to place this understanding in the larger context of development and disease. There are two subsidiary questions. First, what are the *cis*-regulatory modules controlling the expression pattern of a particular gene? And second, what transcription factors bind within those *cis*-regulatory modules? Once answers are known to these questions, there are many ways to study how the different transcription factors combine to control gene expression.

These two questions are, however, difficult to answer experimentally. Unlike protein-coding genes and small RNAs, there is no systematic whole-genome method for identifying *cis*-regulatory elements, and genome-scale binding site identification is difficult to do *in vivo*. Because vertebrate genomes are so large, with well over 100 kb of genomic DNA surrounding many genes, it is not easy to identify even proximal regulatory elements for a particular gene. Moreover, once located, analyzing

regulatory elements for binding sites is not straightforward: biochemical analysis cannot usually be done *in vivo*, and extracting proteins from small, localized domains of tissue is challenging. Thus the opportunity for computational aid is significant.

Computationally, *cis*-regulatory elements are also difficult to identify. Unlike protein-coding genes, which are transcribed and must make a valid protein sequence, *cis*-regulatory elements have no obvious properties that let us identify them easily in single genomes. And, while we know a number of binding sites, transcription factors bind to quite variable sequences, leading to an immensely high false positive rate that has been estimated at 99.9% when current search techniques are applied to whole-genome assemblies (Wasserman and Sandelin, 2004). Even when we can reliably identify individual binding sites, we may not be able to identify the entire *cis*-regulatory element necessary for proper expression.

The advent of multiple whole-genome sequences across a range of evolutionary distances has led to a revolution in the identification of *cis*-regulatory elements. This is because *cis*-regulatory sequences, like protein-coding genes and other functional elements in the genome, are evolutionarily constrained and can be identified by comparative sequence analysis. At the same time, large-scale techniques including chromatin immunoprecipitation followed by either an array assay ("ChIP-chip") or massively parallel sequencing ("ChIP-seq") have begun to provide genome-wide sets of binding sites, empowering binding site search (Hudson and Snyder, 2006; Johnson *et al.*, 2007).

II. Effective *cis*-Regulatory Sequence Analysis

I discuss computational approaches to solving three different types of *cis*-regulatory problems below: first, the problem of finding putative *cis*-regulatory elements within genomic DNA by using comparative sequence analysis to look for evolutionarily conserved noncoding DNA; second, the problem of identifying binding sites for known transcription factors within regulatory elements; and third, the problem of locating novel binding sites in coregulated DNA. Computational sequence analysis techniques have been applied to all of these questions, and the effectiveness of the solutions is in the order given above: comparative sequence analysis has been effective at finding regulatory elements, and when known transcription factor binding sites are sought within relatively short sequences, this approach too has been valuable. Computational discovery of binding sites within several coregulated sequences has been least effective, both because the computational problem is quite difficult and because the biological information is rarely sufficient to narrow down the search sequence or provide enough coregulated sequence elements.

The approach that I recommend is as follows: locate candidate elements with comparative sequence analysis; test the functional relevance of these candidate elements *in vivo*; scan functional sequences with a database of transcription factor

binding sites; and finally, test these sites for function by site-directed mutagenesis and perturbation analysis of upstream genes. This approach uses computation to restrict the number of experiments, while providing quick feedback as to functional significance of sequence elements; it has been successfully used in several laboratories to study many regulatory elements.

A. Selecting Candidate Genomic Regions in Chick

Many regulatory elements lie within the genomic neighborhood of the gene they regulate, although there are notable exceptions. While there are no rules that will work for every gene, generally the place to start looking for regulatory elements is within the region bounded by neighboring genes (including both upstream and downstream sequence, as well as intronic sequence). Here, I show how to extract the chicken *Sox2* genomic region from the UCSC genome browser and determine the appropriate region for investigation.

1. Finding the *Sox2* Genomic Region in Chick

First, go to the UCSC genome website at http://genome.ucsc.edu/.

Select "Genomes" on the upper left to get to the Genome Browser Gateway.

In the search form at the top of the page, choose "Chicken" in the "Genome" pull-down menu. Enter "Sox2" into the box labeled "position or search term". Click submit (see Fig. 1).

You will now see a list of matches to "Sox2" in the chicken genome annotations. The next task is to choose one of these matches. In the case of *Sox2*, this is easy: *Sox2* is in a well-assembled region of the chicken genome and there is a RefSeq gene match to a location on a specific chromosome (chromosome 9, positions 17990091–17991429). RefSeq matches (and UCSC Gene matches) are the highest quality annotation and should be used when present.

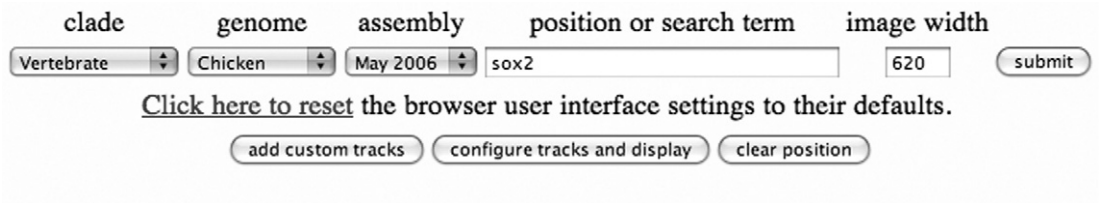


Fig. 1 Searching for chick *Sox2* using the UCSC Genome Browser Gateway.

There are two other common types of search results. The first is a match to an unmapped portion of the genome; for example, if you search for "Dlx3," a member of the *distal-less* gene family, you will find that it is currently (as of the May 2006 assembly) mapped to "chrUn random," which is shorthand for "unmapped region." If your gene of interest is in an unmapped region, you may find that the assembly is imperfect or incomplete; such regions should be treated with caution. (One good tactic for resolving such issues is to compare the region to another, better-assembled genome, such as the mouse genome; if you find discrepancies in exon organization or the position of neighboring genes, then you may be looking at a misassembly.)

Another kind of search result is one in which you simply fail to find a chick RefSeq gene match. For example, a search for "Dlx2" in the May 2006 assembly finds four "Non-Chicken RefSeq genes," corresponding to *Dlx2* TBLASTN matches on chromosome 2. If you examine these matches, you will find that they match to Xenopus *Dlx2*, two zebrafish *Dlx2* paralogs, and a Medaka *Dlx2*. More importantly, the chick gene they match is actually *Dlx5*, not *Dlx2* – *Dlx2* is not present in the May 2006 chick genome assembly! In this case, you are stuck: there is no general way to determine the chick genomic sequence for your gene, although you might be able to use other genomic sequences like mouse or human for comparison or motif search.

Now, select the *Sox2* RefSeq gene match on chromosome 9.

This next Web page shows the *Sox2* RefSeq gene mapped onto the genome; the initial view shows only the *Sox2* RefSeq sequence. In this case, *Sox2* is mapped on the bottom strand (see the "<<" arrows) and only has one exon (see Fig. 2).

Next, zoom out until you can see all of the genomic sequence surrounding *Sox2* up to the neighboring genes; use the 'move' buttons to shift the view so that you include as few neighboring genes as possible. For *Sox2*, this view should include about 600 kb of genomic DNA (see Fig. 3). (For most genes, this should be less than 250 kb.)

This is the region that you want to use for conservation and motif analyses. The figures below use the region **chr9:17,600,308–18,251,061**.

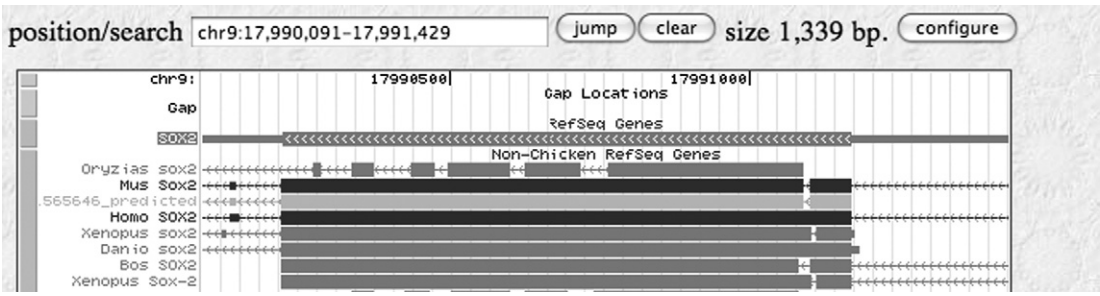


Fig. 2 Chick *Sox2* default view.

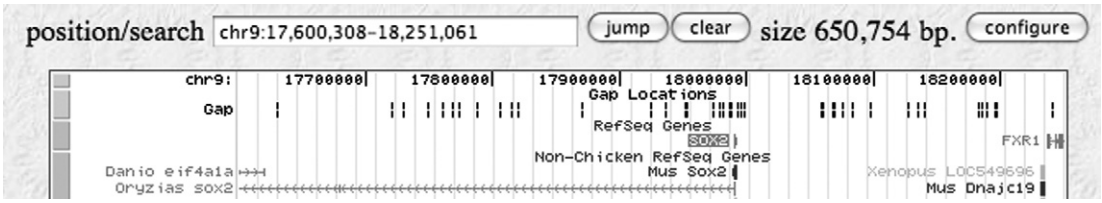


Fig. 3 Chick *Sox2* view, including nearby genes.

2. Saving *Sox2* Genomic DNA into a FASTA File

Most programs do not interact directly with the UCSC browser, and require that you provide DNA sequence in the FASTA format. To extract the genomic region into a FASTA file, go to the UCSC genome view that includes precisely the region you want to analyze, and select "DNA" on the top toolbar. There are several sequence formatting options. The two most important are "Mask repeats," which should be selected (be sure to also select "to N"), and "reverse complement," which (because *Sox2* is on the bottom strand) should be selected in this case. Now select "get DNA."

The next Web page will be a very large text-only screen containing a header (starting with "> galGal3") and all of the genomic sequence you requested.

Use "Save As..." to save this into a text file, or copy and paste it into a word processor such Notepad, TextEdit, or Microsoft Word. Be sure to save the DNA file as a text file (with a ".txt" ending, in plain, text-only format); the extra formatting that is usually added to word processor files will confuse most bioinformatics programs.

At this point, you are ready to import this DNA into a variety of analysis systems (several of which are described below).

III. Approaches and Tools for Finding Conserved Sequence Elements

Conserved noncoding regions are excellent candidates for *cis*-regulatory elements. The most important consideration in looking for conserved noncoding regions seems to be the choice of companion species: there are now over a dozen vertebrate genomes available in draft form, ranging from teleosts to primates. A number of studies have shown that regulatory elements can be conserved between species as distant as primates and teleosts, but there are some indications that the mammalian genomes are the optimal partners for comparison with the

chicken genome. For example, a systematic experimental analysis of the *Sox2* regulatory regions in chick showed that 10 distinct chick regulatory regions were conserved between human, mouse, and chick (Uchikawa *et al.*, 2004). A study of the *SCL* gene region found that five of eight known mouse enhancers were present in the chick *SCL* region, while only two were found in *Fugu rupribes*, a teleost, and none were found in zebrafish (Gottgens *et al.*, 2002). Systematic analysis of *Gdf6* regulation in mouse demonstrated that many of the known regulatory regions were also conserved between mouse and chick, but fewer were present in zebrafish (Portnoy *et al.*, 2005). Thus it seems that while comparisons with zebrafish sequence can find some regulatory regions in amniotes, there are amniote-specific regulatory elements that can only be found by comparisons between chick and mammals.

A. Comparing Sequences

There are three kinds of computational approaches used in comparative sequence analysis: global alignments, local alignments, and dotplots. Global alignments (like those computed by AVID and LAGAN) seek to generate an alignment of two entire sequences, and either do not detect or else minimize rearrangements in the two sequences. With a global alignment, each base on one sequence is aligned to a base or a gap in the other sequence.

Local alignment programs (like NCBI BLAST and blastz) compute all stretches of local similarity between two sequences. Each position in one sequence may be aligned to zero, one, or many other positions in the other sequence. Local alignments can detect evolutionary duplications, shuffled pieces of DNA, or regions that have been inverted in orientation. Local alignment programs usually use simple heuristics to "seed" alignments around points of high local similarity, following which the alignment is expanded subject to a significance threshold; this results in marked speedups, which is why BLAST-style algorithms can be used for whole-genome comparisons.

The third kind of comparison is the dotplot comparison, which uses a very simple algorithm to compare all fixed-width (ungapped) windows of DNA on one sequence with all fixed-width windows of DNA on the other sequence. Because dotplots do an exhaustive comparison, they tend to be much slower but potentially more sensitive than the other kinds of comparisons.

Later, I will show how to establish and visually compare all three types of comparisons. Although there has been no systematic analysis of the effectiveness of each kind of comparison, where they vary in utility may be at the extremes of evolutionary distance: global alignments are unlikely to work well at large evolutionary distances, where sequences may be very dissimilar. At small evolutionary distances, regions are unlikely to have been shuffled or inverted, so a global alignment may more specifically outline conserved sequences. Regardless, all three types

of comparisons have been successfully used to find conserved noncoding regions, and in general it seems that all three types of comparisons identify the same regions of DNA.

B. Tools for Finding Conserved Regions in Genomic DNA

There are a number of tools that can be used for finding conserved regions in genomic DNA, and essentially all of them work well on vertebrate sequence. Two of the most popular are the VISTA browser (Frazer *et al.*, 2004) and PipMaker (Schwartz *et al.*, 2000). One that we have used successfully to study chick elements is the Cartwheel/FamilyRelationsII system, which lets users run several different kinds of analyses and display them simultaneously (Brown *et al.*, 2005).

Later, I will show how to use the UCSC genome browser to survey conservation, and then give a detailed guide to establishing your own comparative analysis in Cartwheel and then viewing it in FamilyRelationsII.

The *Sox2* region has been well studied experimentally, so we will use it for our examples (Inoue *et al.*, 2007; Takemoto *et al.*, 2006; Uchikawa *et al.*, 2003). Later I will show you how to extract homologous regions to *Sox2* from the mouse genome for use in a pairwise comparison.

1. Using the UCSC Genome Browser to Survey Conservation near to *Sox2*

The UCSC Genome Browser is an excellent place to start looking at conservation. By default, the genome browser shows conservation scores calculated using the MULTIZ program (Blanchette *et al.*, 2004). See Fig. 4 for a view of conservation between the chick *Sox2* locus and several other vertebrates; the blue peaks represent the overall strength of conservation, while the individual lines below represent matches in the human, mouse, rat, opossum, frog, and zebrafish genomes.

The UCSC Genome Browser conservation track is useful for an overall view, and it will show you whether or not there is any conservation at all in the area. However, it is not possible to do your own searches or change conservation thresholds within the UCSC Genome browser. Later I will show you how to use the Cartwheel server and FamilyRelationsII software to display several different conservation and similarity calculations, adjust parameters, and extract DNA for further work.

2. Finding the *Sox2* Genomic Region in Mouse and Extracting DNA

To extract the *Sox2* genomic region from mouse, repeat the process for finding the *Sox2* genomic region in chick but select the mouse genome instead. (Be sure to reverse complement the mouse genomic sequence as well, since the *Sox2* gene is on the bottom strand in the mouse genome view.) Save the mouse genomic region into a different FASTA file.

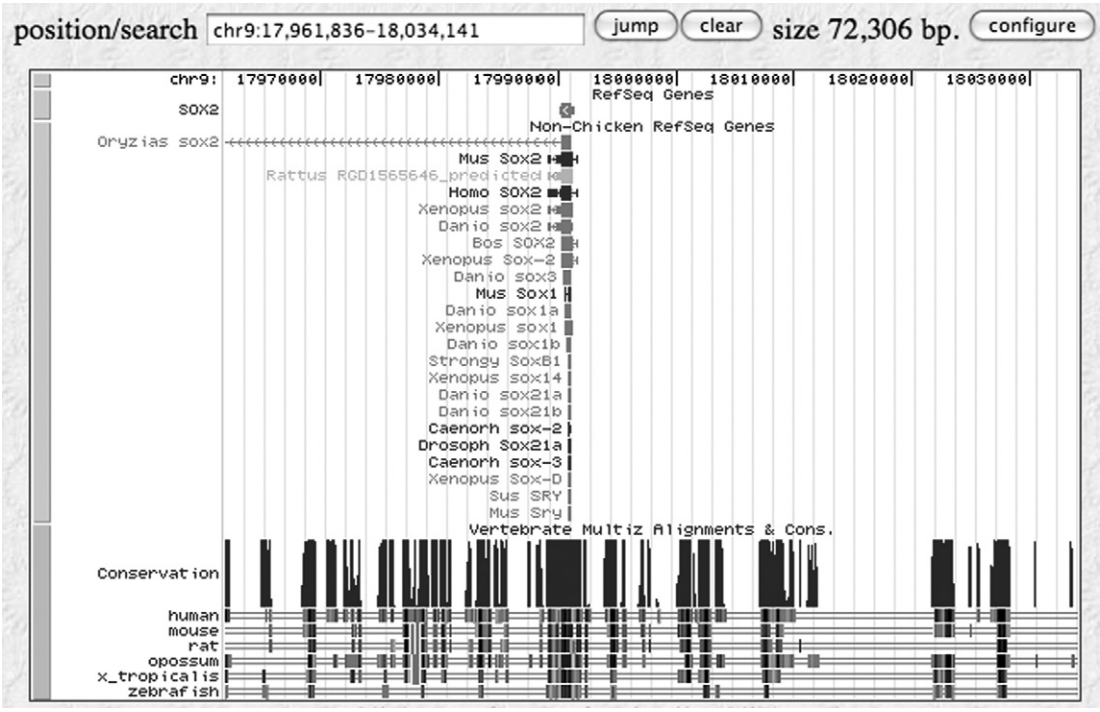


Fig. 4 Sequence conservation near chick *Sox2* in the UCSC Genome Browser.

3. Comparing Chick *Sox2* to Mouse *Sox2* with Cartwheel

The Cartwheel server is a website that lets users upload sequences and build custom annotations and comparisons of the uploaded sequences (Brown *et al.*, 2005). Unlike the UCSC Genome Browser, Cartwheel is targeted at custom analyses of relatively small sequences; it was originally developed to compare BAC-sized genomic sequences from two sea urchins (Brown *et al.*, 2002). An introductory tutorial is available at http://family.caltech.edu/tutorial/.

To make use of Cartwheel, first go to the Cartwheel website, http://woodward.caltech.edu/canal/. Create an account (using "add account") and choose a lab space (the "Public Space for Testing" lab is a good default, if your lab does not yet have its own space).

Now log into your account, enter a folder name describing your project, and click on the "create a new subfolder" button at the bottom of the page. The folder type to create should be left at the default, "analysis folder". Now click on the new folder name.

You will now be in your new folder, which is your personal work space. Next, we need to upload the sequences. Click on "manage sequences," and then select

"upload a FASTA file." Do this for both the mouse and chick sequences you saved from above.

Once you are back at the sequence list, you should rename the sequences to something shorter than the default names, e.g. "chick sox2 genomic" and "mouse sox2 genomic."

Next, we will create a custom view of these sequences containing some comparisons. To do this, return to the folder view, and click on "create an analysis group." Name it "chick/mouse sox2," select "Pairwise analysis of two sequences," and then select "create analysis group," On the next screen, set the top sequence to "chick sox2 genomic" and the bottom sequence to "mouse sox2 genomic." Select "set sequences."

Next, create several pairwise comparisons. We will add two to start with: first, a global alignment with a VISTA-like comparison, and second, a local alignment with a BLAST algorithm.

To do this, select "add pairwise comparison." The next page offers a menu of choices; let us start by running a VISTA comparison with LAGAN. Select this menu option, and then select "create analysis." Set the analysis name to "LAGAN analysis" and select "continue." The next screen offers some options: the window-size specifies how big a window to use for the comparison score (see "Global alignments", above), and the "minimum percent" sets a threshold for the similarity, below which features will not be shown. The default window size (100) and percent similarity (50) are both good places to start. The third option, orientation, specifies whether the second sequence should be compared in forward orientation or reverse complement; this depends on whether or not the *Sox2* genes in the sequences extracted from UCSC are on the same strand in both sequences. (If you followed the directions above, both genes should be on the top strand.) By default, Cartwheel assumes that the sequences should be compared in the same orientation.

Now repeat the process, but add a "PipMaker-style blastz comparison" instead. There are no parameters set for this kind of comparison.

These comparisons are now entered into a processing queue, and we need to wait for them to run. In general, neither type of comparison is very time-consuming, so this should take only a moment or two; reload the browser window a few times until "Job Status" changes to "completed."

The next step is to view the analyses in the FamilyRelationsII viewer.

4. Viewing the Comparisons in FamilyRelationsII

To view the comparisons, you need to download the FamilyRelationsII ("FRII") desktop viewer. This (freely available) viewer loads analyses from Cartwheel and supports zooming, threshold changing, and copy-and-pasting of DNA sequences.

To download the FRII viewer, go to http://family.caltech.edu/ and select the "download" link. There are versions available for both Windows and Mac OS X. Unpack it and start the FRII application.

Now enter your Cartwheel username and password, and go find the analysis folder you created above. Double click on the name of the analysis.

After a moment while the analysis loads, you will see a dotplot-style view of the two sequences, with a series of blue dots marking points of strong similarity between the two sequences (Fig. 5A). Where the two sequences absolutely identical, there would be a blue line of dots running diagonally from the upper left corner to the lower right corner; however, because conservation is patchy between the chick and mouse *Sox2* loci, you will see regions of high similarity interspersed with regions of no similarity. By clicking on "Pair View," you can also switch to an alignment-style view where lines are drawn between similar points on the two sequences (Fig. 5B).

At this point there are a number of things to try.

First, uncheck the "show comparison" box on the right side, in the (blue) "blastz" parameter window. This turns off display of the blastz comparison, and shows only the LAGAN-VISTA matches (in red). These two comparisons overlap almost entirely, demonstrating that (in the case of *Sox2*) the global and local alignments show nearly identical points of similarity.

One can also adjust parameters specific to each analysis. In the case of both LAGAN-VISTA and blastz comparisons, there is only one parameter: threshold. The LAGAN-VISTA threshold is the number of base in a 100 base window that must be aligned to identical bases in order to be displayed.

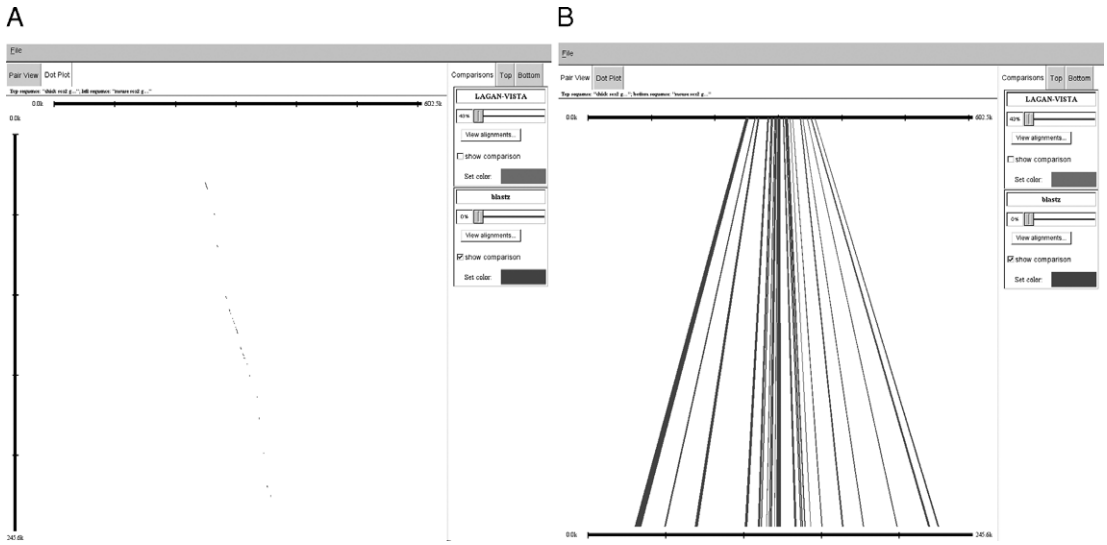


Fig. 5 A FamilyRelationsII view of blastz and LAGAN-VISTA comparisons between the chick (top) and mouse (left/bottom) *Sox2* sequences. The left figure is a dotplot-style view where similarities are plotted at the intersection of positions in the chick sequence (top) and the mouse sequence (left); the right figure is an alignment-style view where lines are drawn between similar points on the chick sequence (top) and the mouse sequence (bottom).

Next, select some small portion of the comparison and zoom. You can do this in two ways: either drag-select on the sequences, by pushing down the left mouse button and moving the mouse along the sequence, or drag-select a rectangle on the dotplot view. Now click in the blue boxes to zoom in a bit. Holding down the SHIFT key and clicking on the sequence will zoom the view back out.

You can zoom in to view the actual sequence. For blastz, select a region of interest, and click on "view alignments." This will bring up a window showing a close-up view of the alignments, together with a window showing the actual nucleotide matches in black (exact match) and red (mismatch) (Fig. 6A).

You can also do a motif search; to bring up the motif search window, right-click or CTRL-click on the sequence of interest, and select motif search. A new window will open up with forms to search for up to five motifs (Fig. 6B). The motif entry form takes IUPAC-style fuzzy motifs like "WGATAR," discussed in detail below. There is also a mismatch slider for each motif, with which you can select how many mismatches to allow for each motif match.

In order to extract sequences, simply select a region of DNA and click on it with the right mouse button (or CTRL-click on a computer with a single mouse button). A menu will pop up; select "copy sequence," and the selected DNA sequence will be put in the system paste buffer, where it can be pasted into another program.

The purpose of the FamilyRelationsII view is to let users establish some comparisons, adjust parameters, and explore the resulting analyses in order to

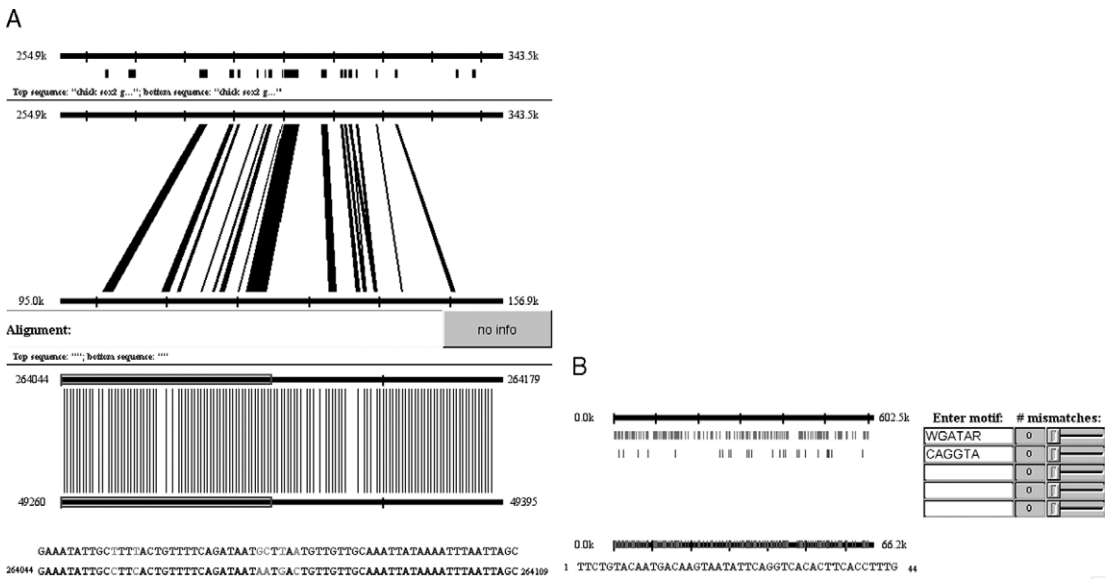


Fig. 6 An alignment close-up view (A) and a motif search window (B) in FamilyRelationsII.

determine which regions of genomic sequence are most likely to be interesting for an experimental test. LAGAN-VISTA and blastz both find similar regions of DNA, and they have effectively helped find regulatory elements in vertebrate comparisons; another program, paircomp, may be slightly more sensitive to trace similarities but can only be used on smaller pieces of DNA.

5. Doing more Sensitive Local Comparisons with Paircomp

The paircomp program does a very straightforward all-by-all dotplot comparison of two regions of DNA, and can be used on regions where each sequence is less than 250 kb of DNA (Brown *et al.*, 2005). Unlike LAGAN-VISTA and blastz, which first build gapped alignments, paircomp compares all fixed-width windows on one sequence to all fixed-width windows on the other sequence and reports those with similarity above a particular threshold. Paircomp was originally developed for sea urchin BAC comparisons, where we found that it worked well on draft unannotated sequence. While no systematic comparison has been done between paircomp, blastz, and LAGAN-VISTA, the paircomp program allows users to specify more stringent or more sensitive comparisons by changing the window size and threshold used. Thus it has been useful in mouse/human comparisons, with a high level of background similarity, as well as in comparisons of fast-evolving sea urchin DNA, with only some trace similarities (Adachi and Rothenberg, 2005; Ransick and Davidson, 2006).

To establish a paircomp comparison, go back to the pairwise analysis in the Cartwheel web server and select ''add pairwise comparison.'' Now select ''add paircomp analysis'' and set a name. The next page contains several parameters: window size and threshold, which control the size of the fixed-width window and the number of matching bases required for a match to be displayed, and also start/stop coordinates for the analysis on both sequences. To select the region proximal to the *Sox2* gene, set the start coordinate to '200,000' and the stop to '400,000' for the chick sequence. On the mouse sequence, choose 200kb centered on the *Sox2* coding region. The choice of window size and threshold can be set to individual taste; for smaller regions (50 kb by 50 kb), a 20 bp window with an 80% threshold is a good place to start.[1] Now submit the analysis to the job queue. Note that paircomp analyses may take a few minutes to run.

Once the analysis is finished, load the pairwise comparison group into Family RelationsII again. The paircomp analysis will be displayed in a third color (green, by default), and the threshold parameter can be adjusted upwards to make the analysis more stringent.

Figure 7 shows a paircomp analysis of the *Sox2* locus in chicken and mouse. Both the coding region (marked in red) and surrounding noncoding sequence show many matches at a threshold of 80% (40 of 50 bases match in each window);

[1] For more information on window size/threshold combinations, see the FamilyRelationsII tutorial at http://family.caltech.edu/tutorial/.

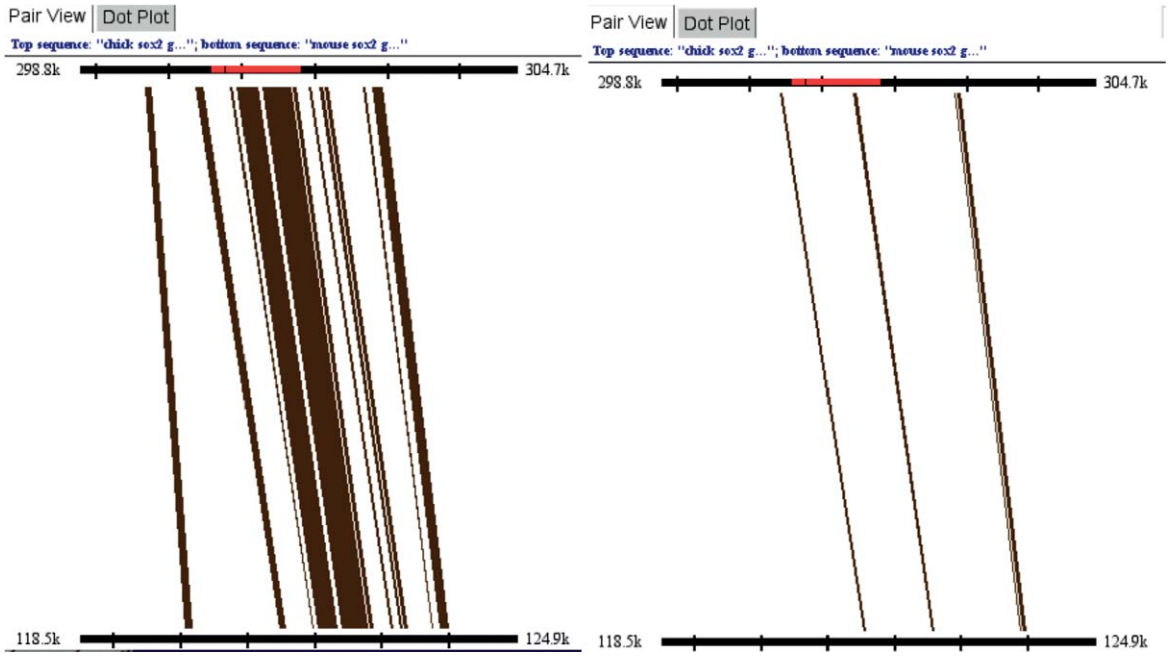


Fig. 7 A view of a 50 bp paircomp analysis comparing chicken (top) and mouse (bottom) DNA surrounding the *Sox2* gene (marked in red on the chicken sequence). (A) 80% threshold and (B) 90% threshold.

by increasing the threshold to 90% (45 of 50 bases match), only three distinct conserved patches remain.

6. Choosing Sequences to Test

Whether you are using LAGAN-VISTA, blastz, or paircomp to do comparisons, the ultimate goal is to test regions for regulatory functionality. How should you choose the regions to test?

In general, regulatory elements seem to be co-linearly conserved that is, regulatory elements are rarely inverted or shuffled in order. This is not just because some programs are biased in favor of detecting co-linear features: while LAGAN-VISTA performs a global alignment first, thus restricting its similarity analysis to co-linear features, both blastz and paircomp are capable of finding points of similarity that have been inverted or shuffled. This conservation of order and orientation in regulatory elements seems to be a general feature of regulatory elements in deuterostomes, and is probably due to the nature of change in genomic

DNA (Cameron *et al.*, 2005). I recommend adjusting comparison thresholds until the only features remaining are co-linear, and then extracting and testing those features. (All of the features in the *Sox2* comparison in Fig. 5 are co-linear: none of the lines in the pairwise view cross each other.)

C. Beyond Pairwise Comparisons

This discussion has focused entirely on pairwise comparisons. Where there are more avian genomic sequences, multiway sequence comparisons might be useful for finding avian-specific fast-evolving sequences, as has been done within the mammals (Margulies *et al.*, 2003). Until the zebra finch genome is completed, pairwise comparisons between chick and a mammal may be the best way to find chick regulatory elements. This is because the most distant genomic sequence has a disproportionate impact on conservation scores (Pollard *et al.*, 2006).

D. "Lies, Damned Lies, and Statistics"

Before continuing, let us return to the question of what kind of comparison algorithm to use. Each kind of comparison – global alignments (LAGAN-VISTA), local alignments (blastz), and dotplot comparisons (paircomp) – implicitly contains assumptions about how sequence evolves, and uses these assumptions to calculate a significance or recommend a threshold. These assumptions are often hidden in the algorithm, and parameters are chosen based on how well a particular choice of parameters performs for specific alignment problems; whether or not the default parameters work well for your problem is unknown.

In this sense, the biggest advantage of dotplot algorithms is that they assume very little about the nature of sequence evolution, and they do not base significance calculations on the length of the match. This is one reason a dotplot comparison like paircomp can be more sensitive (although increased sensitivity can also lead to more false positives).

Thus our advice is to use LAGAN-VISTA and blastz to examine conservation on a large scale, and then do more sensitive, targeted analyses of sequence elements with dotplots.

E. Distal Regulatory Elements and Synteny

If you are interested in a gene that maintains long distance syntenic relationships, then you may need to look further afield for regulatory elements. For example, an enhancer for the *Shh* gene is located one megabase away from the *Shh* gene itself, and this may be the driving force behind the extremely conserved gene order across the 7q36 human locus containing *Shh* (Goode *et al.*, 2005). Long-range gene regulation may also be one cause of the highly conserved *Hox* gene complex (Lee *et al.*, 2006). Kikuta *et al.* (2007) speculate that maintenance of long-range relationships between regulatory elements and genes may be a general constraint on genomic rearrangements leading to synteny (Kikuta *et al.*, 2007).

F. When does Comparative Sequence Analysis not Work?

While comparative sequence analysis has been an effective way to find candidate regulatory elements, there are two scenarios in which it simply cannot work. The first is the obvious case in which the gene in the species of interest has a different expression pattern than the gene in the species being used for comparison. In this case, either the regulatory elements themselves or the spatiotemporal activity of the upstream regulators has diverged, and the regulatory elements may not be present in both species. This case can often be detected by looking at the expression pattern of the gene in the available species.

The other scenario in which comparative sequence analysis fails is when regulatory sequence for a particular gene has diverged even though the expression pattern driven by the two different sequences is similar. This is a form of neutral evolution, and it can be difficult to detect without exhaustive experimental *cis*-regulatory analysis or detailed knowledge of the important binding sites. While there are a number of examples of divergent regulatory sequence with similar regulatory function outside the vertebrates, we know of only one clear example in the vertebrates: the *SCL* gene locus in Fugu contains a regulatory element that shares no detectable sequence identity with the known regulatory elements in mouse, yet is capable of driving correct expression in *Fugu* (Barton *et al.*, 2001). There is no effective way of finding such elements computationally. Note that this may be one of reasons why McGaughey *et al.* (2008) found that comparative tools identified regulatory elements with relatively poor sensitivity across great evolutionary distances.

IV. Identifying Transcription Factor Binding Sites Computationally

Precisely locating individual transcription factor binding sites is distinct from the problem of locating functional *cis*-regulatory modules. While *cis*-regulatory modules are often capable of driving or repressing gene expression in reporter assays independently of any other regulatory element, binding sites rarely act alone; *cis*-regulatory modules often contain combinations of binding sites for several different transcription factors (Davidson, 2006). Thus *cis*-regulatory modules are larger than individual binding sites and are often amenable to detection by conservation analysis, perhaps because the strength, spacing, and orientation of the binding sites is constrained (Cameron *et al.*, 2005).

Individual transcription factor binding sites, however, are much harder to find and characterize. Even fairly sequence-specific transcription factors may recognize and bind to a variety of sites; these sites may be as long as 15–20 bases, but the invariant core sequence can be as few as 4 bases, e.g., for a GATA factor. This degeneracy in binding sites contributes to many false positives and complicates the *de novo* discovery of binding sites based solely on patterns in sequence.

Experimental techniques for finding binding sites do not always work well: electrophoretic mobility shift assay (EMSAs), or "gel shifts," require large amounts of starting protein, as do DNAse footprints; chromatin immunoprecipitation requires many input cells as well as a good antibody; and large-scale assay techniques are expensive and lack well-defined positive controls.

The degeneracy of known binding sites and the paucity of detailed biochemical models of specific transcription factor binding make large-scale sequence search for known binding sites a tricky business. Nonetheless, there are a variety of techniques that can be used to find potential binding sites for a specific transcription factor family, given some knowledge of existing binding sites. Discovering new binding sites from sequence data alone is much less reliable, and seems to be very dependent on the input data, search tools, and parameters used for the search.

Here I discuss useful techniques for performing a computational search with known binding sites, and also mention the problem of predicting new binding sites from sequence data.

A. Searching for Known Binding Sites

Suppose that we are interested in searching for binding sites for a particular transcription factor within a given sequence, and some known binding sites for that transcription factor have already been garnered from *in vitro* or *in vivo* assays. Were one to search for only the known binding sites, a large number of slightly divergent sites would be missed. Therefore the approach of choice in this case is to *generalize* from the existing set of binding sites.

There are two commonly used techniques for generalizing from a set of known binding sites. The first technique is a search with a fuzzy motif, which can be written in a variety of ways. For example, if one wanted to match either an A or a T, then G-A-T-A, and then either an A or a G, the search motif would be written as [AT]GATA[AG], or alternatively as WGATAR using IUPAC notation. The second technique is to use a matrix scoring system, variously known as a position-frequency matrix (PFM), position-weight matrix (PWM) or a position-specific scoring matrix (PSSM) (Wasserman and Sandelin, 2004). A matrix-based search allows for the definition of preferred or accepted nucleotides in each position. For example, this matrix finds [AT]GATA[AG] binding sites with equal weighting on an A or a T at the beginning, and an A or a G at the end:

	1	2	3	4	5	6
A	0.5	-	1.0	-	1.0	0.5
C	-	-	-	-	-	-
G	-	1.0	-	-	-	0.5
T	0.5	-	-	1.0	-	-

One can read this matrix from left to right, as allowing either an A or a T, then a G, A, T, A, and then either an A or a G. (For clarity, the "-" marks represent zeroes in the matrix.) If, however, you know that an A is preferred over a T in the first position, you can change the matrix accordingly:

	1	2	3	4	5	6
A	**0.8**	-	1.0	-	1.0	0.5
C	-	-	-	-	-	-
G	-	1.0	-	-	-	0.5
T	**0.2**	-	-	1.0	-	-

For the first position, this matrix weights an A higher than a T, which in turn is weighted higher than a C or a G. Note that the consensus binding site is read off simply by finding the nucleotide with the highest score in each column – AGATA [AG], for this matrix.

In order to construct a motif, we start with a set of known binding sites, gathered either from a high-throughput experiment like SELEX or from individually verified sites (Ellington and Szostak, 1990). These sites are subsequently aligned without gaps, using (for example) CLUSTALW with a high gap-opening penalty. Then, for each position in the alignment, which of the four nucleotides show up in that position is recorded. For example, given an alignment of three short motifs,

AGATAA
TGATAA
AGATAG
—****—

one can see that the core four bases are always "GATA," with either an A or a T in the first position and an A or a G in the last position. This yields a motif of [AT]GATA[AG].

Much the same procedure is used to construct a search matrix, except that frequency of each nucleotide is taken into account. For example, the alignment above would yield the following PFM:

	1	2	3	4	5	6
A	$\frac{2}{3}$	-	$\frac{3}{3}$	-	$\frac{3}{3}$	$\frac{2}{3}$
C	-	-	-	-	-	-
G	-	$\frac{3}{3}$	-	-	-	$\frac{1}{3}$
T	$\frac{1}{3}$	-	-	$\frac{3}{3}$	-	-

This weight matrix can then be used to score *any* subsequence of six bases as a potential match: for each position in the putative site, one uses the matrix to look up the score for the nucleotide at that position, and then sums the scores to obtain a total. This total is then used to rank the sites. For example, using this matrix to score the three sites used to construct the matrix, we get:

AGATAA	2/3+4+2/3	=16/3
TGATAA	1/3+4+2/3	=15/3
AGATAG	2/3+4+1/3	=15/3

As would be expected from looking at the matrix, "AGATAA" is the highest-scoring site of the three. Also note that this matrix gives site TGATAG a score of 14/3, ranking it only one step below the three known sites but above all other sites – suggesting that TGATAG may also be a valid binding site for this transcription factor. This ability to generalize from a small set of known binding sites to a larger set of sites is the reason why people use both motifs and matrix-based approaches, as I mentioned earlier.

There are two more interesting features of matrices. The first is that the highest-scoring binding site calculated from matrices may not actually be in the list of sites used to calculate the matrix. For example, suppose we construct a PFM from the following four sites:

AGATAG
AGATAT
TGATAA
GGATAA
—****—

The highest-scoring motif under the resulting matrix will be "AGATAA," which is not in the list of input sites! This is because the matrix construction process counts the frequency of nucleotides in each position independently, so all that matters is that there are two 'A's in both the first and last columns. The other interesting feature of matrices is that the scores calculated from specific types of matrices – PWMs, which are calculated by taking the logarithm of each entry in a PFM – are, in theory, related to the actual biochemical binding strength of the site. Thus not only are high-scoring sites predicted to be more likely actual binding sites, but these high-scoring sites may be more strongly bound by the actual transcription factor. While this relies on a number of limiting assumptions – the primary being that each position in a site must contribute independently to the strength of binding – there are some indications that it holds true for some transcription factors (Stormo, 2000).

Matrices are a flexible way to search for binding sites, and they are at the foundation of most binding site search programs. Unfortunately, there are

essentially no tools that let biologists create matrix-based motifs from lists of known sites, although websites like CONSITE (discussed later) let you search with predefined matrices and there are several programs that support motif discovery (also discussed later).

So, which of the two search techniques – motifs or matrices – is better? The unfortunate answer is that the easier-to-use technique supported by many tools, searching with motifs, is considerably worse. This is because as the number of known binding sites increases, the resulting motif must become less specific in order to include all of the permutations of the individual bases, while position-weight matrices usually become more specific given more known motifs. Moreover, matrices rank candidate binding sites by putative strength or similarity to the known binding sites, unlike motifs which simply report matches. Motifs are therefore generally poor choices for transcription factor binding site search, because they lead to less specific searches and do not rank sites. Conversely, matrices are problematic because they require choosing a threshold below which you no longer consider a site to be relevant, and there are no generally agreed-upon techniques for choosing that threshold computationally. (Note that motifs are technically a special case of matrices: every motif search can be implemented as a matrix search, although not every matrix can be represented as a motif.)

Both matrices and motifs have several drawbacks as models of transcription factor binding. First, neither matrices nor motifs can represent synergistic interactions between positions in binding sites; were the above WGATAR binding site a strong binding site only when an A was present in both the first and last positions at the same time, neither search technique would be able to accurately rank the site. Second, neither matrices nor motifs can find sites that have insertions or deletions (''indels'') in them. And, of course, both matrices and motifs score *individual* sites, and not collections of sites.

With these drawbacks, it is worth asking why we should use either motifs or matrices! While there are more complicated binding site representations that can take into account synergistic interactions between positions in a binding site, gaps, or indels, and neighboring binding sites, we rarely have enough information about real binding sites to make use of these other representations. For example, to accurately calculate pairwise interactions between each pair of positions from a set of known sites, one needs several *hundred* known sites. Moreover, by moving to a more complicated binding site representation, one also must move to a more complicated statistical calculation that is likely to admit more false positives into the search results. Thus it seems that matrices, as imperfect as they are, represent a good starting point for motif searches.

B. Practical Motif Searching

The primary practical problem with searching for binding sites in vertebrate sequence is that there is so much sequence to search! On average, each gene in the human genome has over 70 kb of noncoding sequence surrounding it, all of which

may contain functional binding sites. While there are some relatively simple ways to narrow the search – for example, looking at sequence conservation and synteny relationships, as described earlier – you may still need to search several kilobases of DNA. Given the short length of most known sites and the low specificity of matrices, this can lead to many more predicted sites than are likely to be functional.

While there is no general way to limit this false positive rate, there are several control "experiments" that can give you an intuitive feeling for the likely false positive rate for a particular search. If searching only within conserved regions, one can also search nearby nonconserved sequence to get an estimate of how many sites are found in DNA that is unlikely to be functional. Alternatively, one can do the same motif search in sequence taken from somewhere else in the genome, near to genes that are unlikely to be regulated by the transcription factor in question. While it is always possible that real binding sites are situated in nonconserved sequence or randomly chosen genomic sequence, it is unlikely enough to serve as a reasonable control. Be sure to avoid comparing searches of your candidate DNA with searches on repetitive sequence or searches on sequence with a very different GC content; both repeat elements and GC- or AT-rich sequences will bias your results.

Another source of high false positive rates is a poor choice of threshold for a matrix search. As described earlier, matrices rank binding sites with a numerical score, and the lower the score the less likely it is that that site is a real binding site. Where should the threshold be set? Again, there is no general technique to determine a good threshold, but the scores of the sites that were used to construct the matrix should give an idea of what scores to consider: a threshold that includes many but excludes a few of the known sites is likely to be safe. For example, using the GATA-site PFM above, we score the three input sites as follows:

AGATAA	2/3 + 4 + 2/3	= 16/3
TGATAA	1/3 + 4 + 2/3	= 15/3
AGATAG	2/3 + 4 + 1/3	= 15/3

In this case, these three sites are the three highest ranked sites possible with this matrix. The next best site is "TGATAG," with a score of 14/3, and there are four sites (GGATAT, GGATAC, CGATAT, CGATAC) that have a score of 12/3. All other six-base sites have scores of 11/3 or below; because of the combinatorics of DNA, there are dozens of sites that have a score of 10/3. By ranking the input sites, we can guess that a choice of 15/3 for a threshold is good, and anything lower than 12/3 will probably admit many false positives.

This illustrates a general problem with matrices: as the cutoff threshold decreases, the number of matching sites increases dramatically. For example, as you decrease the threshold for the JASPAR HFH-2 PWM from its maximum of 58 to 40, the number of sites found by the PWM rises from 1 to over 420,000 (Fig. 8).

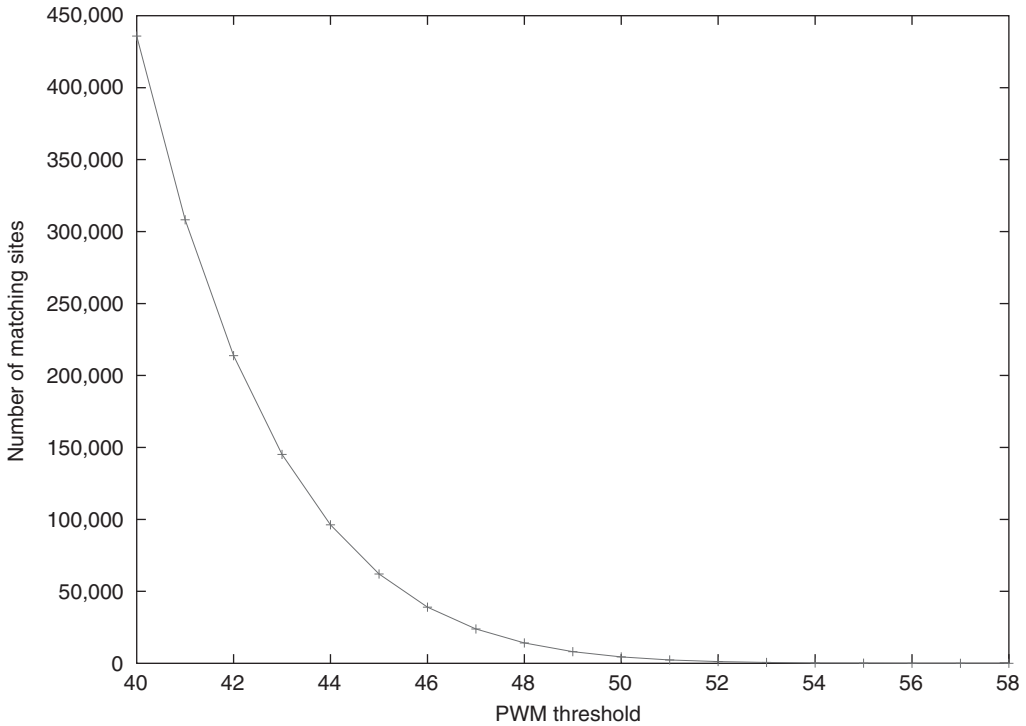


Fig. 8 How the number of sites matching a PWM increases as the threshold decreases. Calculations were done using the HFH-2 Forkhead matrix taken from the JASPAR binding site collection, against a genomic background with no A/T bias. The Y axis is the number of possible 12 base sites that match the PWM at or above the given threshold. Calculations were done with the motility toolkit, unpublished software available at http://cartwheel.caltech.edu/motility/.

This corresponds to a PWM/threshold combination that discovers two sites in every 1 kb fragment!

This result demonstrates the mathematical tendency of binding site searches towards false positives that underlies the so-called "Futility Theorem" (Wasserman and Sandelin, 2004). The Futility Theorem states that 99.9% of binding site predictions in undistinguished genomic DNA are probably false positives, or, equivalently, that a given binding site match is 99.9% likely to be a false positive.

C. Using JASPAR and CONSITE to Scan DNA with a Binding Site Library

There are several sites that can search a sequence against a library of binding site matrices. The JASPAR collection of sites is a high-quality library of multicellular transcription factor binding sites based primarily on SELEX data (Vlieghe *et al.*, 2006). The CONSITE service uses JASPAR as a source of binding site

data and lets users search any piece of DNA against the library (Sandelin *et al.*, 2004).

To demonstrate this service, let us take the *Sox2* N-1c regulatory element and use CONSITE to scan it for known binding sites.

First, retrieve the 55 bp N-1c sequence from UCSC by searching for the region **chr9:17,980,31817,980,373** in the *Gallus gallus* genome. Export this DNA as described above; note that to get it in the same orientation as in Takemoto *et al.* (2006) you need to export the reverse complement.

Now go to the CONSITE website, http://www.phylofoot.org/consite/, and select "Analyze single sequence." Paste the sequence into the top sequence box, and set the sequence name to "Sox2 N-1c." Select "Proceed to transcription factor selection."

On this next screen, you can select a search set of JASPAR transcription factors from the list, or simply ask CONSITE to search for all of them. For this search, we will ask for a global search, which is the default.

The other important setting on this page is the specificity setting, which is set to "10 bits" by default. This is analogous to the threshold for matrix searches described above, and this parameter can be varied to get different results: lowered, for less stringency, or raised, for more. The mathematical meaning of a bit in this case is that each precisely specified individual base is worth two bits, so a 10 bit threshold would find an exact match to a 5-base site. (The meaning is not actually so straightforward because matrices perform fuzzy matching, but the math works out to 2 bits per exact base match.)

Leave the setting to 10 bits and select "Analyze the sequence(s) with all TFs." This will run a search against the entire JASPAR library. You will see an image like that in Fig. 9A, where the name of the matrix match is provided along with a pointer to the location of the match. To see the match position, matrix itself, the transcription factor name, and the organism in which the data for the matrix was recorded, click on the matrix name (Fig. 9C).

Rerun the analysis with a lower threshold of 8 bits (Fig. 9B).

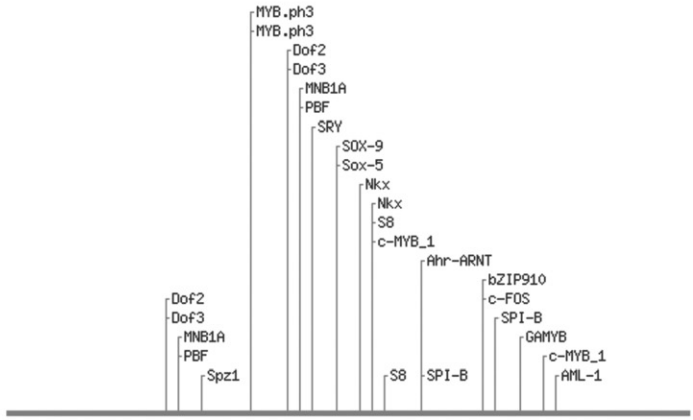
As expected, lowering the threshold only a little bit increases the number of matches significantly; this is again because of the fuzzy matching "Futility Theorem" problem described earlier. Moreover, none of the binding sites found match the known binding sites in this region, because the actual binding sites are too different from the matrices in the library; one would have to admit many more false positives in order to find the two Tcf/Lef binding sites in this region, for example. Nonetheless, JASPAR and CONSITE are good tools to have in the toolbox, and as we have shown it is quite easy to try them out.

D. Searching for Combinations of Known Motifs

Very few transcription factors act alone, and many seem to work in combination with the same partners to coregulate different genes. Therefore, searching for closely grouped combinations of known motifs can be an effective way to reduce

A **Putative transcription factor binding sites found along *sox2_n-1c***

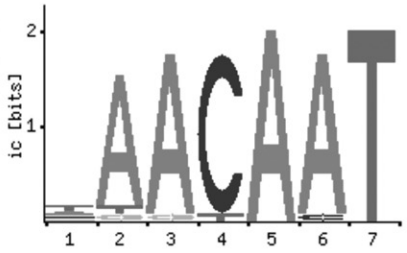
B **Putative transcription factor binding sites found along *sox2_n-1c***



C

Score	Position	Sequence
7.562	28	sox2_n-1c

ID	MA0087
Name	Sox-5
Class	HMG
Species	Mus musculus
Sequence	
Lit ref	
Medline	




```
A  [ 8 21 22  0 23 22  0 ]
C  [ 4  0  0 22  0  1  0 ]
G  [ 3  1  1  0  0  0  0 ]
T  [ 8  1  0  1  0  0 23 ]
```

Fig. 9 Using CONSITE to search for JASPAR matrix matches in the *Sox2* N-1c enhancer. (A) Results with a more stringent threshold of 10 bits; (B) results with a less stringent threshold of 8 bits and (C) information on the Sox-5 match from (A). The matrix is given in both numbers of each nucleotide at each position and in the ''sequence logo'' format, where the height of each nucleotide indicates the importance of that nucleotide to the match score.

the false positive rate inherent in the single-site search techniques discussed above. However, the sparseness of our current biological knowledge of sites and associated factors limits the effectiveness of combinatorial search, because it is more than likely that we know neither what partners are involved nor what their binding sites are.

One of the easiest to use combinatorial search tools is the ClusterBuster program developed by Zhiping Weng's lab at Boston University (Frith *et al.*, 2003). Let us first try a simple search recapitulating the discovery of two TCF/Lef binding sites in the *Sox2* N-1c enhancer (Takemoto *et al.*, 2006).

For this search, you will need to retrieve the genomic sequence surrounding the N-1c enhancer. The genomic coordinates for the 1 kb including this region in the May 2006 chicken genome assembly are **chr9:17,980,000–17,981,000**; go to the UCSC genome browser site and enter this position into the search box (Fig. 10). Now extract the DNA so that you see it in your browser window (click on "DNA," followed by "Submit") and copy it into your paste buffer.

Now go to the ClusterBuster website search form, http://zlab.bu.edu/cluster-buster/cbust.

Paste the DNA into the search box at the top of the form, and enter the following matrix (representing a TCF/Lef binding site) into the motif box, under "Select a bunch of motifs."

```
>tcf-lef
0100
0110
0001
0001
0001
0010
1000
```

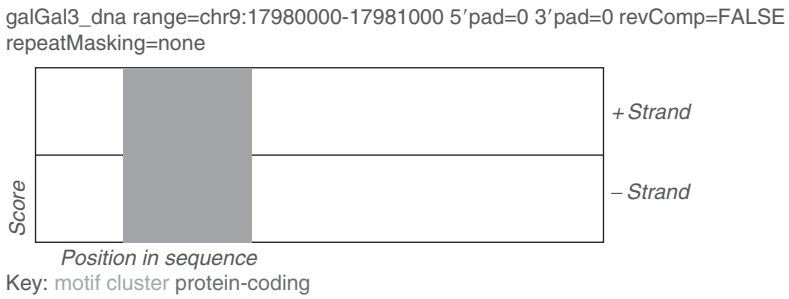


Fig. 10 Using ClusterBuster to find a pair of TCF/Lef binding sites in the *Sox2* N-1c enhancer.

Also change the gap parameter to 5, to allow the binding sites to be close together. Now submit the search with "Find my clusters."

The result will show you a single cluster, in green, located 150 bases into the 1 kb fragment; this is the cluster identified and tested experimentally by Takemoto *et al.* (2006).

ClusterBuster can also be used to search larger sequence regions. Taneyhill *et al.* (2007) identified a *Cadherin6b* regulatory element by searching for pairs of *Snail2* binding sites within the 8 kb of genomic sequence surrounding the start codon of *Cadherin6B*. If you recapitulate their search with ClusterBuster, using the DNA from chick **chr2:70,535,000–70,545,000**, a cluster threshold score of 2, and the following snail matrix representing "CAGGTA,"

```
>snail
0100
1000
0010
0010
0001
1000
```

you will see multiple clusters (Fig. 11), the second, fourth, and fifth of which correspond to those tested in Taneyhill *et al.* (2007).

Although both of these examples show searches for homotypic clusters, or clusters of the same binding site, it is possible to add multiple matrix entries into the search box. In this case, combinations of different matrices will be used in the search.

The drawback of this kind of search is that one needs to know what binding sites are likely to be found in combination. It is also possible, although much more difficult, to discover unknown binding sites given several coregulated regulatory regions.

E. Predicting Unknown Binding Sites Computationally

There are a number of methods for predicting unknown binding sites computationally, but they do not work well until at least four or five sequences containing similar motifs are available. Even then, results are poor, especially on metazoan sequences; an initial survey of computational tools found that sensitivity and

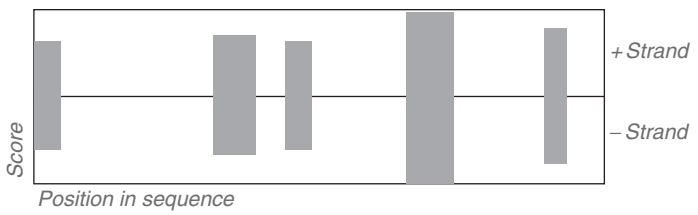


Fig. 11 Using ClusterBuster to find a pair of snail binding sites near the *Cadherin6b* translation start.

predictive value of most tools were under 15% (Tompa *et al.*, 2005). While such assessments are difficult to do properly because of our general lack of known-good binding sites, these results do not bode well for biologists attempting to use such programs on their own sequences. One encouraging result of this study was that combining results from multiple binding site search programs is valuable. In one case, combining the predictions from two programs doubled the correlation between nucleotides predicted to be part of a site and those actually part of a site. To quote,

> Biologists would be well advised to use a few complementary tools in combination rather than relying on a single one and to pursue the top few predicted motifs of each rather than the single most significant motif.

The review contains links to a number of programs for identifying "common" binding sites in a selection of sequences (Tompa *et al.*, 2005). The WebMOTIFs site at http://fraenkel.mit.edu/webmotifs/ is one site that allows combined searches with several motif discovery programs.

F. Integrating Conservation and Motif-Search Evidence

Above, I suggest starting with experimental analysis of conserved regions to identify functional *cis*-regulatory modules, and then proceeding to binding site analysis to identify individual binding sites. Many discussions of binding site analysis focus on using conservation as independent evidence for functionality. Unfortunately, if conservation is used to narrow down the amount of sequence to be searched with a motif, conservation cannot then be used as independent evidence of binding site functionality. Given the generally poor quality of binding site search and the high false positive rate of bioinformatic methods in general, I have found that the relatively straightforward procedure of experimentally testing conserved elements is the most effective way to find functional DNA. Once functional regulatory elements are known, they become amenable to the more sensitive but less specific bioinformatics techniques such as motif searching, and they can also be probed with a variety of experimental techniques such as deletion, mutation, footprinting, and ChIP analysis.

If conservation analysis does not find the regulatory element of interest, I recommend trying the "shotgun" approach used by Uchikawa *et al.* (2003).

Acknowledgments

Tatjana Sauka-Spengler, Erich Schwarz, and Sonja McKeown gave many useful comments and corrections on this chapter. This work was supported by NIH grant P50 HG004071.

References

Adachi, S., and Rothenberg, E. V. (2005). Cell-type-specific epigenetic marking of the il2 gene at a distal cis-regulatory region in competent, nontranscribing t-cells. *Nucleic Acids Res.* **33**(10), 3200–3210.

Barton, L. M., Gottgens, B., Gering, M., Gilbert, J. G., Grafham, D., Rogers, J., Bentley, D., Patient, R., and Green, A. R. (2001). Regulation of the stem cell leukemia (SCL) gene: A tale of two fishes. *Proc. Natl. Acad. Sci. USA* **98**(12), 6747–6752.

Blanchette, M., Kent, W. J., Riemer, C., Elnitski, L., Smit, A. F., Roskin, K. M., Baertsch, R., Rosenbloom, K., Clawson, H., Green, E. D., Haussler, D., and Miller, W. (2004). Aligning multiple genomic sequences with the threaded blockset aligner. *Genome Res.* **14**(4), 708–715.

Brown, C. T., Rust, A. G., Clarke, P. J., Pan, Z., Schilstra, M. J., De Buysscher, T., Griffin, G., Wold, B. J., Cameron, R. A., Davidson, E. H., and Bolouri, H. (2002). New computational approaches for analysis of cis-regulatory networks. *Dev. Biol.* **246**(1), 86–102.

Brown, C. T., Xie, Y., Davidson, E. H., and Cameron, R. A. (2005). Paircomp, familyrelationsii and cartwheel: Tools for interspecific sequence comparison. *BMC Bioinform.* **6,** 70.

Cameron, R. A., Chow, S. H., Berney, K., Chiu, T. Y., Yuan, Q. A., Kramer, A., Helguero, A., Ransick, A., Yun, M., and Davidson, E. H. (2005). An evolutionary constraint: Strongly disfavored class of change in dna sequence during divergence of cis-regulatory modules. *Proc. Natl. Acad. Sci. USA* **102**(33), 11769–11774.

Davidson, E. H. (2006). ''The Regulatory Genome''. Academic Press, New York.

Ellington, A. D., and Szostak, J. W. (1990). *In vitro* selection of RNA molecules that bind specific ligands. *Nature* **346**(6287), 818–822.

Frazer, K. A., Pachter, L., Poliakov, A., Rubin, E. M., and Dubchak, I. (2004). Vista: Computational tools for comparative genomics. *Nucleic Acids Res.* **32**(Web Server issue), W273–W279.

Frith, M. C., Li, M. C., and Weng, Z. (2003). Cluster-buster: Finding dense clusters of motifs in dna sequences. *Nucleic Acids Res.* **31**(13), 3666–3668.

Goode, D. K., Snell, P., Smith, S. F., Cooke, J. E., and Elgar, G. (2005). Highly conserved regulatory elements around the *Shh* gene may contribute to the maintenance of conserved synteny across human chromosome 7q36.3. *Genomics* **86**(2), 172–181.

Gottgens, B., Barton, L. M., Chapman, M. A., Sinclair, A. M., Knudsen, B., Grafham, D., Gilbert, J. G., Rogers, J., Bentley, D. R., and Green, A. R. (2002). Transcriptional regulation of the stem cell leukemia gene (SCL)–comparative analysis of five vertebrate scl loci. *Genome Res.* **12**(5), 749–759.

Hudson, M. E., and Snyder, M. (2006). High-throughput methods of regulatory element discovery. *Biotechniques* **41**(6), 673675, 677.

Inoue, M., Kamachi, Y., Matsunami, H., Imada, K., Uchikawa, M., and Kondoh, H. (2007). Pax6 and Sox2 dependent regulation of the Sox2 enhancer N-3 involved in embryonic visual system development. *Genes Cells* **12**(9), 1049–1061.

Johnson, D. S., Mortazavi, A., Myers, R. M., and Wold, B. (2007). Genome-wide mapping of *in vivo* protein DNA interactions. *Science* **316**(5830), 1497–1502.

Kikuta, H., Laplante, M., Navratilova, P., Komisarczuk, A. Z., Engstrom, P. G., Fredman, D., Akalin, A., Caccamo, M., Sealy, I., Howe, K., Ghislain, J., Pezeron, G. *et al.* (2007). Genomic regulatory blocks encompass multiple neighboring genes and maintain conserved synteny in vertebrates. *Genome Res.* **17**(5), 545–555.

Lee, A. P., Koh, E. G., Tay, A., Brenner, S., and Venkatesh, B. (2006). Highly conserved syntenic blocks at the vertebrate Hox loci and conserved regulatory elements within and outside *Hox* gene clusters. *Proc. Natl. Acad. Sci. USA* **103**(18), 6994–6999.

Margulies, E. H., Blanchette, M., Haussler, D., and Green, E. D. (2003). Identification and characterization of multi-species conserved sequences. *Genome Res.* **13**(12), 2507–2518.

McGaughey, D. M., Vinton, R. M., Huynh, J., Al-Saif, A., Beer, M. A., and McCallion, A. S. (2008). Metrics of sequence constraint overlook regulatory sequences in an exhaustive analysis at phox2b. *Genome Res.* **18**(2), 252–260.

Pollard, D. A., Moses, A. M., Iyer, V. N., and Eisen, M. B. (2006). Detecting the limits of regulatory element conservation and divergence estimation using pairwise and multiple alignments. *BMC Bioinformatics* **7,** 376.

Portnoy, M. E., McDermott, K. J., Antonellis, A., Margulies, E. H., Prasad, A. B., Kingsley, D. M., Green, E. D., and Mortlock, D. P. (2005). Detection of potential gdf6 regulatory elements by multispecies sequence comparisons and identification of a skeletal joint enhancer. *Genomics* **86**(3), 295–305.

Ransick, A., and Davidson, E. H. (2006). cis-Regulatory processing of notch signaling input to the sea urchin glial cells missing gene during mesoderm specification. *Dev. Biol.* **297**(2), 587–602.

Sandelin, A., Wasserman, W. W., and Lenhard, B. (2004). Consite: Web-based prediction of regulatory elements using cross-species comparison. *Nucleic Acids Res.* **32**(Web Server issue), W249–W252.

Schwartz, S., Zhang, Z., Frazer, K. A., Smit, A., Riemer, C., Bouck, J., Gibbs, R., Hardison, R., and Miller, W. (2000). Pipmaker–A web server for aligning two genomic DNA sequences. *Genome Res.* **10** (4), 577–586.

Stormo, G. D. (2000). DNA binding sites: Representation and discovery. *Bioinformatics* **16**(1), 16–23.

Takemoto, T., Uchikawa, M., Kamachi, Y., and Kondoh, H. (2006). Convergence of Wnt and Fgf signals in the genesis of posterior neural plate through activation of the Sox2 enhancer N-1. *Development* **133**(2), 297–306.

Taneyhill, L. A., Coles, E. G., and Bronner-Fraser, M. (2007). Snail2 directly represses cadherin6b during epithelial-to-mesenchymal transitions of the neural crest. *Development* **134**(8), 1481–1490.

Tompa, M., Li, N., Bailey, T. L., Church, G. M., De Moor, B., Eskin, E., Favorov, A. V., Frith, M. C., Fu, Y., Kent, W. J., Makeev, V. J., Mironov, A. A. *et al*. (2005). Assessing computational tools for the discovery of transcription factor binding sites. *Nat. Biotechnol.* **23**(1), 137–144.

Uchikawa, M., Ishida, Y., Takemoto, T., Kamachi, Y., and Kondoh, H. (2003). Functional analysis of chicken Sox2 enhancers highlights an array of diverse regulatory elements that are conserved in mammals. *Dev. Cell* **4**(4), 509–519.

Uchikawa, M., Takemoto, T., Kamachi, Y., and Kondoh, H. (2004). Efficient identification of regulatory sequences in the chicken genome by a powerful combination of embryo electroporation and genome comparison. *Mech. Dev.* **121**(9), 1145–1158.

Vlieghe, D., Sandelin, A., De Bleser, P. J., Vleminckx, K., Wasserman, W. W., van Roy, F., and Lenhard, B. (2006). A new generation of JASPAR, the open-access repository for transcription factor binding site profiles. *Nucleic Acids Res.* **34**(Database issue), D95–D97.

Wasserman, W. W., and Sandelin, A. (2004). Applied bioinformatics for the identification of regulatory elements. *Nat. Rev. Genet.* **5**(4), 276–287.