Software

DiagHunter and GenoPix2D: programs for genomic comparisons, large-scale homology discovery and visualization

Steven B Cannon*, Alexander Kozik†, Brian Chan†, Richard Michelmore† and Nevin D Young*‡

Addresses: *Plant Biology Department, University of Minnesota, St Paul, MN 55108, USA. †Department of Vegetable Crops, University of California, Davis, CA 95616, USA. ‡Plant Pathology Department, University of Minnesota, St Paul, MN 55108, USA.

Correspondence: Steven B Cannon. E-mail: canno010@umn.edu

Published: 19 September 2003

Genome ***Biology*** 2003, **4:**R68

The electronic version of this article is the complete one and can be found online at http://genomebiology.com/2003/4/10/R68

Received: 1 May 2003
Revised: 19 June 2003
Accepted: 8 August 2003

Abstract

The DiagHunter and GenoPix2D applications work together to enable genomic comparisons and exploration at both genome-wide and single-gene scales. DiagHunter identifies homologous regions (synteny blocks) within or between genomes. DiagHunter works efficiently with diverse, large datasets to predict extended and interrupted synteny blocks and to generate graphical and text output quickly. GenoPix2D allows interactive display of synteny blocks and other genomic features, as well as querying by annotation and by sequence similarity.

Rationale

Numerous programs exist for identifying regions of homology or collinearity between two genomes, including PipMaker [1,2], MUMmer [3,4], FORRepeats [5], REPuter [6,7], ADHoRe [8], GRIMM [9,10], BLASTZ [11] and PatternHunter [12]. Of course, not all of these serve the same needs. Programs designed to identify homologous regions by making sequence alignments (such as PipMaker and BLASTZ) may excel at accurately finding high-quality short alignments, but not at piecing together and reporting larger genomic-scale homologies. Programs designed to work with nucleotide data may not work with peptide data, or raw positional information such as marker positions. With increasing numbers of genomes being sequenced, it is becoming a more frequent task to identify very large, often multi-megabase, corresponding regions between two genomes. These are frequently interrupted by insertions, deletions, or small inversions in one or both genomes, but can nevertheless be considered to have recognizable common ancestry and genomic context, and to show approximate collinearity in gene order. These large chromosomal regions that are similar in content and organization are frequently referred to as synteny blocks [4,10,13-15] or, in the case of a comparison of a genome to itself, segmental duplications [16-19].

Here we report two new programs, DiagHunter and GenoPix2D, which are distributed together, and may be used together or separately to explore comparative genomic data at genome-wide or single-gene scales. DiagHunter has several strengths for large-scale collinearity prediction: it is cross-platform (Windows, Unix, Linux, OS X, and others, requiring Perl and the GD module); it can easily and quickly handle large amounts of new genomic data; it can operate on protein, nucleotide, marker, or other data types; it includes BioPerl and Tcl/Tk-based scripts for processing various input data formats; and it generates simple text output consisting of lists of gene or nucleotide sequence or marker pairs and coordinates. DiagHunter identifies very large-scale synteny blocks - on the order of many megabases - despite high levels of background noise in the comparisons and despite large genomic discontinuities. It is a 'lighter-weight' program than most of those mentioned above, in that it does not carry out sequence

alignments (as do PipMaker, MUMmer, FORRepeats, REPuter, BLASTZ and PatternHunter). But its speed, low memory requirement, simple output format, platform portability, and flexible input format make it a worthwhile addition to the set of genome-exploration tools. It also appears to outperform the programs above in identifying extended, disrupted blocks as parts of larger features.

While DiagHunter is useful for identifying synteny blocks, it does not permit interactive querying of features in a genomic comparison. GenoPix2D provides this function, allowing highlighting of groups of genes (for example, synteny blocks found by DiagHunter) or sets of genes showing a specified level of similarity to two genes from the genomes being compared. GenoPix2D (a Tcl/Tk-based comparative, two-dimensional 'extension' of the GenomePixelizer software [20,21]) can also generate PostScript output files of essentially any size, including query results and simple annotations added during a session of the program.

As more genome sequences are completed, it becomes increasingly important for researchers to be able to reproduce and reanalyze reports of genomic comparisons. It should also be possible to redo comparisons efficiently as genome assemblies are updated and reannotated. For example, several seminal genome-scale comparisons of internal duplications in the *Arabidopsis thaliana* genome have identified large internal synteny blocks, or segmental duplications (for example [18,19,22,23]), but depended on custom bioinformatics efforts and extensive pre- and post-processing tailored to the *Arabidopsis* genome. Comparable predictions of segmental duplications in *Arabidopsis* can be carried out using DiagHunter on a standard, single-processor desktop computer in about 10 minutes if provided with parsed BLAST [24] data (which can be generated in about a day, including the underlying BLAST analysis, with accompanying scripts). Samples from these comparisons are shown in Figure 1.

The DiagHunter/GenoPix2D distribution includes scripts that allow processing of genomic data from various input formats (National Center for Biotechnology Information (NCBI) genomic protein data sets, The Institute for Genomic Research (TIGR) pseudochromosome assemblies, or Ensembl [14] gene queries), to produce finished images and predictions of synteny blocks. Other input formats should be easily accommodated. DiagHunter has been tested primarily with comparisons of predicted genes, but also works with DNA or other comparisons that generate similarity matrices. Certain genomic material, such as comparisons of highly repetitive heterochromatic DNA, present particular challenges, but comparisons between these regions can be masked and excluded from the analysis.

The output of DiagHunter is in the form of text files and PNG (portable network graphics) image files. Output includes two text files containing gene names and/or coordinates, and a detailed and a thumbnail PNG image for each chromosome-by-chromosome comparison. BLAST hits above a specified threshold are shown in the image files, with direct repeats shown in one color and inverted repeats in another color. Predicted diagonals are highlighted and numbered. All chromosomes in one or more genomes can be run in a batch process.

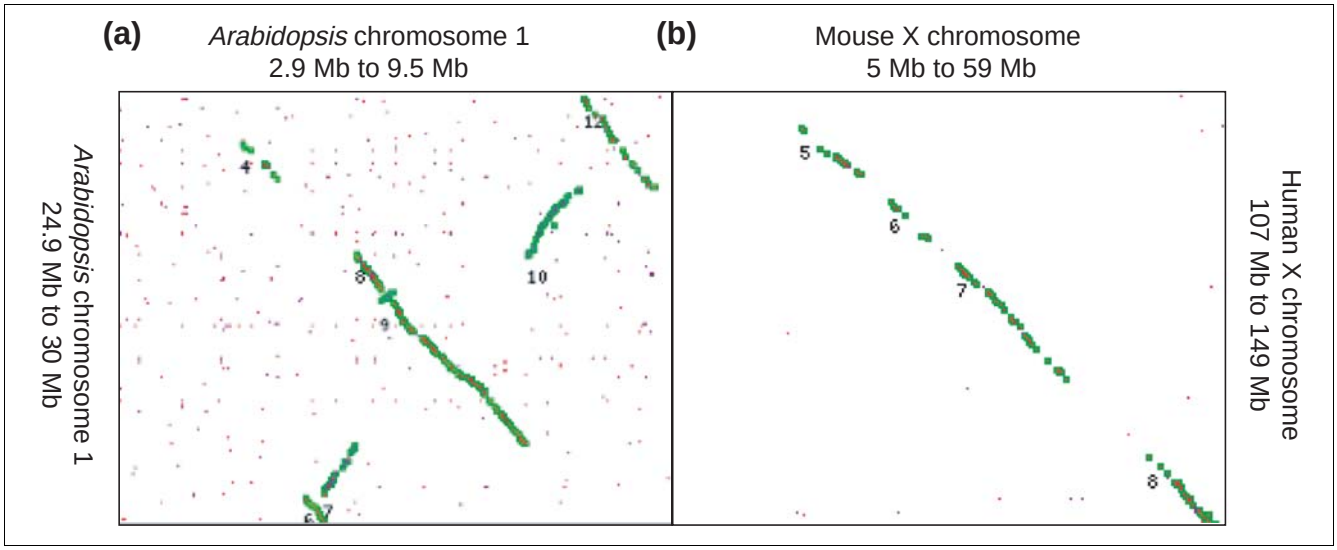

Figure 1
DiagHunter output. **(a)** Internal duplications within portions of *Arabidopsis* chromosome 1. **(b)** A comparison of predicted genes in portions of the mouse and human X chromosomes (mouse on the horizontal axis and human on the vertical axis). Numbers are assigned by DiagHunter in the order in which synteny blocks are identified. Note that the mouse/human window dimensions are approximately 12 times those shown in the *Arabidopsis* comparison.

To view DiagHunter predictions, as well as any other two-way genomic similarity comparisons, GenoPix2D provides interactive viewing, querying, and plotting.

DiagHunter algorithm and implementation

DiagHunter is implemented in Perl, and requires the BioPerl [25] and GD.pm modules [26] for BLAST parsing and image generation, respectively. DiagHunter takes as input simple, tab-delimited, parsed BLAST (or other similarity comparison) text files. Although designed to work with protein sequences, the actual requirements are for a file of coordinates of similarity 'hits', hit strengths, and (optionally but recommended) strand orientation. A BioPerl-based BLAST controller and parser that generate this format are included in the distribution.

A genomic collinearity finder must deal with the following characteristics of genomic comparisons. First, collinear regions must, by their nature, produce diagonal features in a dot plot. Second, in genomic data, these features may be interrupted by gene losses or by insertions in one or both regions. Third, there may be background noise in the form of homologies outside of collinear regions - such as local gene duplications, or high-copy gene families. The algorithm deals with these characteristics by looking locally for the best diagonal candidates, compressing the search matrix to reduce the span between hits in diagonals, and heavily weighting against purely vertical or horizontal steps (which might cause a search to incorrectly follow a line of highly repetitive hits).

The collinearity-identifying algorithm of DiagHunter works by walking through a pre-computed array of filtered similarity hits. At each hit, it checks in the neighborhood for hits that might either be other members of 'direct repeats' (forward steps in both genomic regions) or 'inverted repeats' (opposite steps in the two genomic regions), choosing the nearest and most favorable positions first. Once a candidate diagonal has been initiated, only hits with appropriate combinations of orientations are considered (if working with predicted coding-sequence data that include strand-orientation information). The program follows these chains of best local diagonal steps recursively, checking up to 75 possible positions for each step in the vicinity. Each position contributes a score from a scoring matrix that gives the best scores to the nearest and most-nearly-diagonal hits. A representation of the DiagHunter scoring matrix is shown in Figure 2, where X represents the location of the 'hit' at the current position in the search, and the area of the circles represents the preferred search order and score assigned to hits at any of these locations in the compressed hit matrix. Forward searches use the right-hand scoring matrix, and reverse searches use the left-hand matrix. At each step, a running-average score for the candidate diagonal is computed. The program repeats the search to extend the current diagonal until a score threshold is passed or until no other candidate hits are located. Local gene duplications are

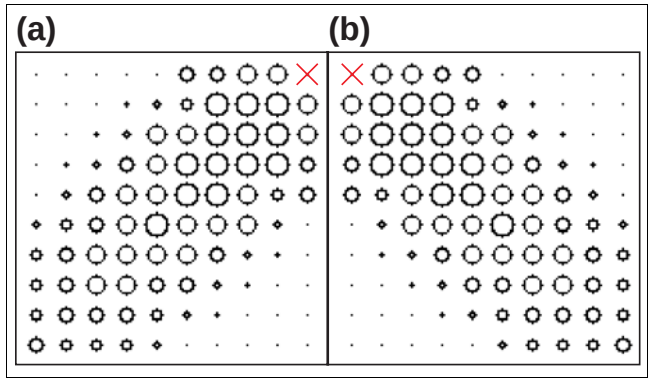

Figure 2
Search order and scoring matrix for DiagHunter. As the program walks through a matrix of similarity hits, each hit (represented by a red X) initiates a search for a diagonal. Candidates for extension of the diagonal are picked first from those closest to the present hit, and closest to either a forward or reverse diagonal. The search order and scores are represented by the area of the circles in the figure. Once either a forward or a reverse diagonal is initiated, only the reverse **(a)** or forward **(b)** version of the search order/score matrix is used for extension of the candidate diagonal, and the diagonal is recursively extended while candidate hits are available and the accumulated diagonal quality score remains above the quality threshold parameter. The sensitivity and selectivity of the search is increased if coding-strand information is included, as this reduces the number of 'random' hits that might be considered at any step.

added to the current diagonal and are removed from subsequent searches. This helps to consolidate diagonals in which significant local duplications have occurred. If the diagonal meets the selection parameters (numbers of hits in the diagonal, and average diagonal score) then the diagonal is retained, and all hits picked up in this search are removed from subsequent searches. The program then starts the search again from where the previous diagonal search was initiated. If the sparse hits are brought 'closer together' by compressing the original matrix (a DiagHunter parameter), sensitivity may be increased and computational time decreased (as described below under DiagHunter performance). If this sort of compression is done, the original coordinates are recovered at the end.

As described earlier, the search order/scoring matrix is used in two ways: first, to specify the order in which the recursive search algorithm checks the hit matrix for the next diagonal element; and second, to determine whether the overall diagonal score is sufficient to continue the current search. The more critical and sensitive function is the determination of search order. This is particularly true during vertical or horizontal (V/H) gap extensions (accomplished by any move that pushes the search in either a vertical-only or a horizontal-only direction). The reason this kind of V/H gap extension is a special case is that in actual genomic similarity data, long runs of hits are much more common than would be expected in random data. These occur because of gene families, common domains, or (in nucleotide data) high-copy repetitive

sequences. At the same time, it is not uncommon for an insertion or a deletion in one genome to introduce a V/H gap in the hit matrix. Therefore, it is essential to allow V/H extensions, but only after more preferable diagonal moves have been tried and exhausted. In Figure 2, this compromise is apparent as relatively small values in the positions representing vertical or horizontal moves. However, tightly regulating near-vertical moves is less critical, because these kinds of relationships are not as common in genomic data.

The order of the algorithm, for a sparse hit matrix between two genomic regions of sizes M and N, with a compression factor of C, is $O(MN/C^2)$. In practice, with a standard desktop computer, an appropriately compressed comparison of either mouse versus human or *Arabidopsis* versus *Arabidopsis* chromosomes takes less than a minute. Processing times are not substantially longer for comparisons of the proteomes of the much larger mouse versus human genomes. Although M and N (genome sizes measured in nucleotides) are much larger in the mouse versus human case, C should also be proportionately larger because gene density is so much lower in mammalian genomes than in *Arabidopsis*. In practice, therefore, the order for proteome comparisons appears to depend approximately on the gene (or other similarity) counts in the two genomes rather than on nucleotide-count genome size.

DiagHunter performance

The performance of DiagHunter was tested with comparisons of predicted proteins in *Arabidopsis* versus itself, and nucleotide versus nucleotide comparisons of *Arabidopsis* versus itself and mouse versus human. In these tests, predictions were consistent with and were approximately as sensitive and selective as other published analyses of these datasets [10,11,13,18,22,23].

In a comparison of *Arabidopsis* chromosome 1 with itself (biologically interesting because of extensive internal segmental duplication), DiagHunter identified the same 24 duplication regions as reported by Simillion *et al.* [23]. Some of these regions are shown in the close-up of *Arabidopsis* 1 × 1 diagonals in Figure 1a, and all are shown in the thumbnail image of the whole-chromosome comparison in Figure 3a (the black call-out indicates the portion of the figure that is shown in detail in Figure 1). Results for the other *Arabidopsis* chromosomes also closely match predictions by Simillion *et al.* [23]. A small modification of the hit matrix during data pre-processing (using only the top 10 BLAST hits in addition to filtering by hit strength, and then lowering the acceptable BLAST hit-strength threshold in DiagHunter) appears to significantly improve DiagHunter's sensitivity and selectivity. For example, this results in the prediction of two additional probable synteny blocks in the chromosome 1 × 1 comparison (data available at [27]). Simulation and sensitivity tests in this paper, however, focus on the more difficult task of predicting synteny blocks in BLAST data not filtered by hit number.

It is interesting to contrast the peptide-based comparison in Figure 3a with the nucleotide-based comparison in Figure 3b. The nucleotide-based comparison was generated with a BLASTN-based comparison of all 1-kb chunks of the nucleotide sequence of chromosome 1 pseudochromosome assembly against all others.

DiagHunter predicts the large segmental duplications similarly in both datasets, but in the nucleotide comparison misses some of the smaller ones discovered by the more sensitive protein-protein comparison (although a PatternHunter [12] or BLASTZ [11] DNA-based homology search may provide greater DNA-DNA sensitivity than this rather crude block-based BLASTN approach). Likely reasons for the lower sensitivity in this nucleotide comparison are: the protein-based comparison uses strand-orientation information (reducing the potential 'random background' hits by half once any diagonal search is initiated); the BLASTP similarity search is inherently more sensitive than BLASTN [24]; and the BLASTP hit matrix includes fewer BLAST hits that are due to repetitive DNA. In fact, the highly repetitive centromeric region produces such a dense feature in the nucleotide-nucleotide comparison that this region had to be masked and excluded from the search.

In a comparison of mouse and human X chromosomes (biologically interesting because of the high degree of homology that has been retained in this sex chromosome), the program found 15-18 extended diagonals, depending on search parameters (15 were found in the conservative search that produced Figure 1b). These extended up to a total of 16.8 Mb. This is similar (in number and, generally, in diagonal quality) to the 16 synteny blocks identified by Pevzner and Tesler [10], although it is instructive to contrast the methods and results in more detail. Pevzner and Tesler began with a set of "bidirectional best local similarities (also called 'anchors')", generated by PatternHunter [12]. This step is analogous to the BLASTP search conducted in the pre-processing step for DiagHunter, although PatternHunter aligns DNA sequences rather than protein sequences (as we used in the BLASTP search). Because DiagHunter relies only on a similarity matrix, suitably parsed output from PatternHunter or other similarity-search methods could also be used as DiagHunter input - which would take better advantage of noncoding sequences and unannotated DNA. The next step in the Pevzner and Tesler study (before inferring genome rearrangements, the paper's primary objective) was to construct synteny blocks based on these anchors. This step, carried out by the GRIMM-Synteny algorithm [9,10], connects anchor points (similarity hits), if the distance between points is smaller than a given gap size, to form clusters of anchor points. 'Small' clusters are then deleted, leaving synteny blocks. In the comparison of human versus mouse X chromosomes, with a gap threshold of 100 kb and PatternHunter-generated anchor points, the GRIMM-Synteny algorithm identifies 16 synteny blocks (see Figure 1b in [10]). This

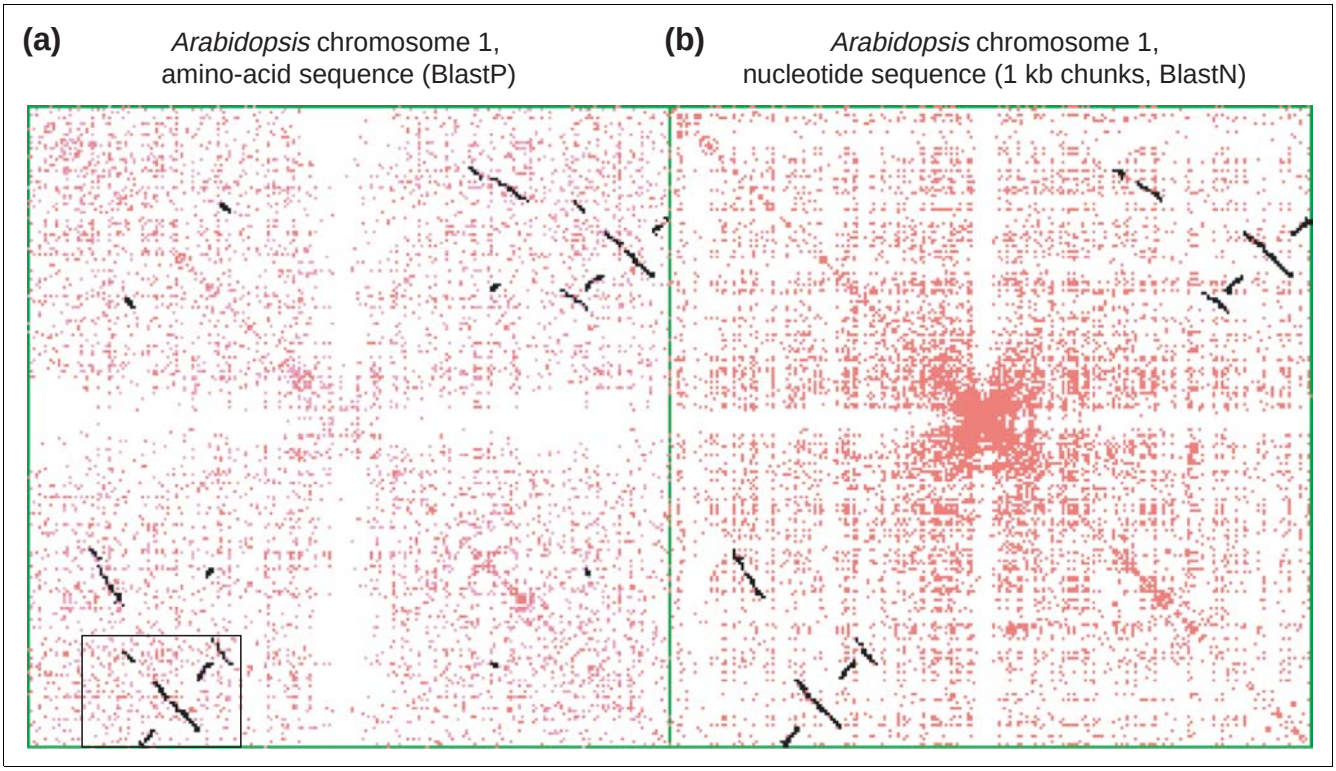

Figure 3
Internal duplications within portions of *Arabidopsis* chromosome 1 (whole chromosome). **(a)** A comparison based on predicted genes. **(b)** A comparison based on a similarity search between all 1 kb nucleotide segments from the TIGR chromosome 1 assembly. The region in the black rectangle in (a) is shown in Figure 1. In the protein-protein comparison (a), hits with between genes with Watson-Watson or Crick-Crick ('forward diagonal') orientations are shown in red, and hits between genes with Watson-Crick or Crick-Watson ('reverse diagonal') orientations are shown in pink. The use of strand orientation in the protein-protein comparison helps to explain the greater sensitivity of this search, compared with the nucleotide comparison in (b). The dense centromeric region encompassing the main diagonal in the nucleotide-nucleotide comparison was excluded from this search.

compares with 15 homology regions identified by DiagHunter (the search that produced Figure 1), using a matrix compression factor of 220. At these thresholds, and using sparser protein-protein comparisons than were generated by GRIMM anchor points, the DiagHunter algorithm misses two small diagonals identified by GRIMM, and makes one additional split of a diagonal. These differences probably depend primarily on the PatternHunter-BLASTP differences, although both algorithms are also sensitive to gap- or matrix-compression parameters.

The DiagHunter search depends on several empirically determined parameters: the matrix compression, hit score cutoff, the average diagonal score, and the minimum hits to accept per diagonal. These parameters may differ greatly from genome to genome, so for any new genome comparison some experimentation will be needed to find parameters that give a good trade-off between sensitivity and selectivity. In *Arabidopsis*, for example, gene density is roughly one gene per 5 kb, but in the X chromosomes of mouse and human, the gene density is approximately one gene per 130 kb. For perspective, the entire *Arabidopsis* genome, at roughly 125 Mb, is smaller than either the mouse or human X chromosomes (at roughly 150 Mb), even though those chromosomes contain fewer than 1,100 predicted genes in Ensembl [14]. The level of background noise is also much higher in the *Arabidopsis* self-comparison, in which ongoing gene duplication, loss, transposition, and so on generate strong homologies unrelated to ancient segmental duplications. This is evident in the many off-diagonal BLAST hits in the *Arabidopsis* comparison in Figures 1 and 3. The lower background noise in the mouse versus human comparison allows higher matrix compression levels, because random off-diagonal hits are unlikely to be brought close enough together to produce spurious predictions of collinearity.

A comparison of various parameters for the *Arabidopsis* peptide self-comparison is shown in Figures 4 and 5. Figure 4 shows true and false positives in the chromosome 1 × 1 comparison under various matrix compression factors and diagonal 'quality' thresholds. True and false positives were judged relative to published literature on these duplications [18,19,23]. Generally, as matrix compression was increased over the range of 10× to 30×, the number of true positives increased, and the number of diagonal fragments decreased - corresponding to an increase in the 'coherence' of the

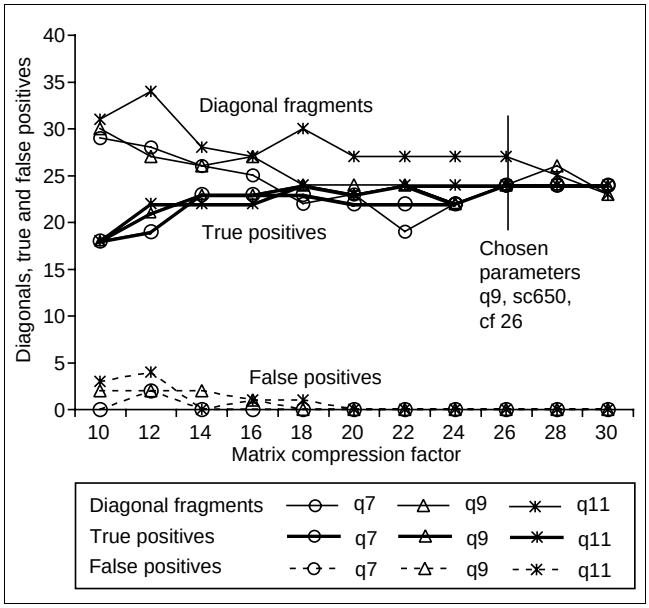

Figure 4
True-positive and false-positive diagonal predictions in *Arabidopsis* chromosome 1 × 1 under various DiagHunter search parameters. Compare with false-positive predictions shown in Figure 5. True and false positives are judged relative to predictions reported in the literature [18,19,22,23]. At low matrix compression factors, some diagonals are missed (bold graph lines), and those that are identified are fragmentary (thin solid graph line). At higher compression factors, random background hits should begin to 'merge' to give false-positive predictions. This has not occurred in these parameter ranges (dotted graph lines), though the program did mistakenly predict short diagonals in dense clusters of locally duplicated genes at low compression factors, particularly at less stringent diagonal quality thresholds (lines with asterisks). Parameters used in Figures 1 and 3 are: quality = 9, bit score = 650 (approximately corresponding in this dataset to BLAST expect values of < 10^{-68}), compression factor = 26. These selections are indicated in a vertical line in the graph. Abbreviations: q, diagonal quality; sc, bit score; cf, compression factor.

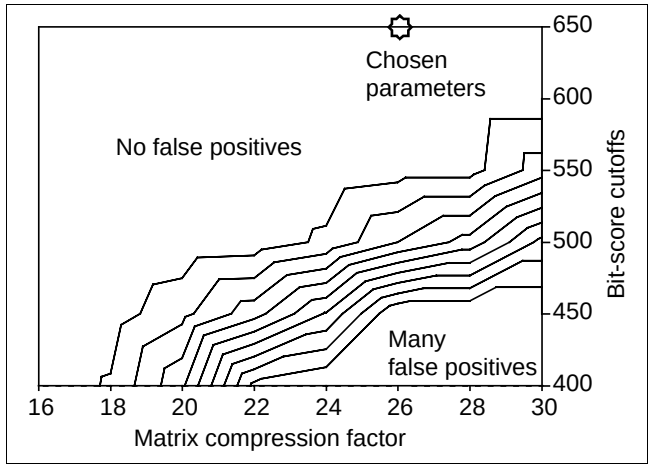

Figure 5
False-positive homology predictions in simulated data matrices with densities equivalent to those in 'background' duplications in a BLASTP comparison of *Arabidopsis* chromosomes 1 and 2. Compare with true positive and false positives in actual data in Figure 4. False positives increase rapidly if the matrix compression factor (horizontal axis) and hit density (vertical axis; determined by bit-score cutoff) are such that simulated hits occur close enough together that spurious diagonals having the specified quality and hit number can be found in the simulated data. Parameters should be chosen to avoid false positives. Parameters used in Figures 1 and 3 are: quality = 9, bit score = 650 (approximately corresponding in this dataset to BLAST expect values of < 10^{-68}), compression factor = 26. These selections are indicated with a star in this graph.

predicted diagonals. In other words, as the hit matrix is more highly compressed, diagonal fragments are brought close enough together that they merge. Beyond a certain point, diagonals that should be considered as separate might be artificially brought together, and hits that are not part of homology regions might be misjudged as members of diagonals. The coordinates of nearby hits may also collapse to the same coordinate in the compressed matrix, resulting in loss of some information in the predicted diagonal pairs. In the parameter ranges used in these simulations, additional diagonals are not mispredicted in the chromosome 1 × 1 comparison, though interestingly, some false positives are seen at low compression factors as the program misidentifies diagonals in dense 'knots' of local gene duplications. Therefore, there appear to be more disadvantages in this dataset from undercompression of the hit matrix than from what might be considered overcompression. Figure 4 also shows results for three different diagonal quality thresholds. The quality score is a running average of scores assigned from the hit score matrix depicted in Figure 2. A lower-quality threshold is more stringent. Generally, a higher threshold produces more true-positive predictions at a given matrix compression factor.

Figure 5 shows false-positive predictions in randomly distributed data that have the same density of 'background' hits in *Arabidopsis* × *Arabidopsis* peptide comparisons, at the indicated BLAST bit score cutoffs and matrix compression factors. Intuitively, if random data is brought close enough together, then it will all merge so that essentially all hits will be falsely predicted as members of diagonals. In simulations, data densities were determined by observing the number of 'background' hits in *Arabidopsis* chromosome 1 × 2 at bit scores ranging from 400 to 650 (in this dataset, these cutoffs correspond to expect values of approximately 10^{-39} to 10^{-68}, respectively). Matrix compression factors ranging from 14 to 30 were set in runs of DiagHunter. At each indicated compression factor and bit score cutoff (grid intersections in the figure), DiagHunter was run on 10 randomly distributed data sets at these densities. Each gradient represents the observed and extrapolated parameter combinations at which one additional false positive would be observed. Parameters for these data, therefore, should be chosen outside of the false-positive region.

For a new genome comparison, tests with a range of compression factors fairly quickly show the point at which obviously

spurious diagonals are predicted, and the sort of analysis presented here quickly guides the user to identify reasonable parameters. In Figures 1 and 3, the chosen parameters were compression factor = 26 and bit score cutoff = 650, well outside of the false-positive region in these simulations. These are good, but probably not optimal, parameter choices. It appears that better results can be obtained by additionally filtering to the top 5-20 BLAST hits, and lowering the bit score cutoff and matrix compression (data not shown, but available at [27]. This has the effect of decreasing background 'noise' due to high-copy sequences, while allowing the program to use more distant similarity relationships.

Features of GenoPix2D

GenoPix2D is a general-purpose, cross-platform, flexible, queryable genomic dot-plotter/gene-family viewer, well suited to analyzing results of DiagHunter or similar programs. GenoPix2D is a desktop application implemented in Tcl/Tk [28], and should run on all platforms that support the Tcl/Tk interpreter. GenoPix2D uses an interactive graphical user interface to display, query, filter, zoom, and plot 2D genome comparisons. GenoPix2D takes as input simple, tab-delimited files containing BLAST hit information and gene coordinates. The format of these files is compatible with GenomePixelizer, described previously [20,21]. Images generated by GenoPix2D are interactively searchable for particular elements (genes). Results of a search (for example, gene-family members, or diagonals detected by DiagHunter) can be painted in different colors defined by the user. Figure 6 displays the same region of the *Arabidopsis* genome as in Figure 1a, this time displayed by GenoPix2D. Different gene families are distinguishable by different colors. In Figure 6, members of three large gene families are shown: the NBS (nucleotide-binding site)-containing resistance gene candidates [29], the cytochrome P450s [30], and the LRR (leucine-rich repeat) protein kinases [31]. Interactive exploration of the region makes it possible to find all homologs to any selected dot (BLAST hit) on the image. These features of GenoPix2D provide detailed analysis of syntenic regions detected by DiagHunter at the single-gene level. It is also worth noting that GenoPix2D can be used view comparative genomic data independently of DiagHunter, requiring only the basic Tcl/Tk installation.

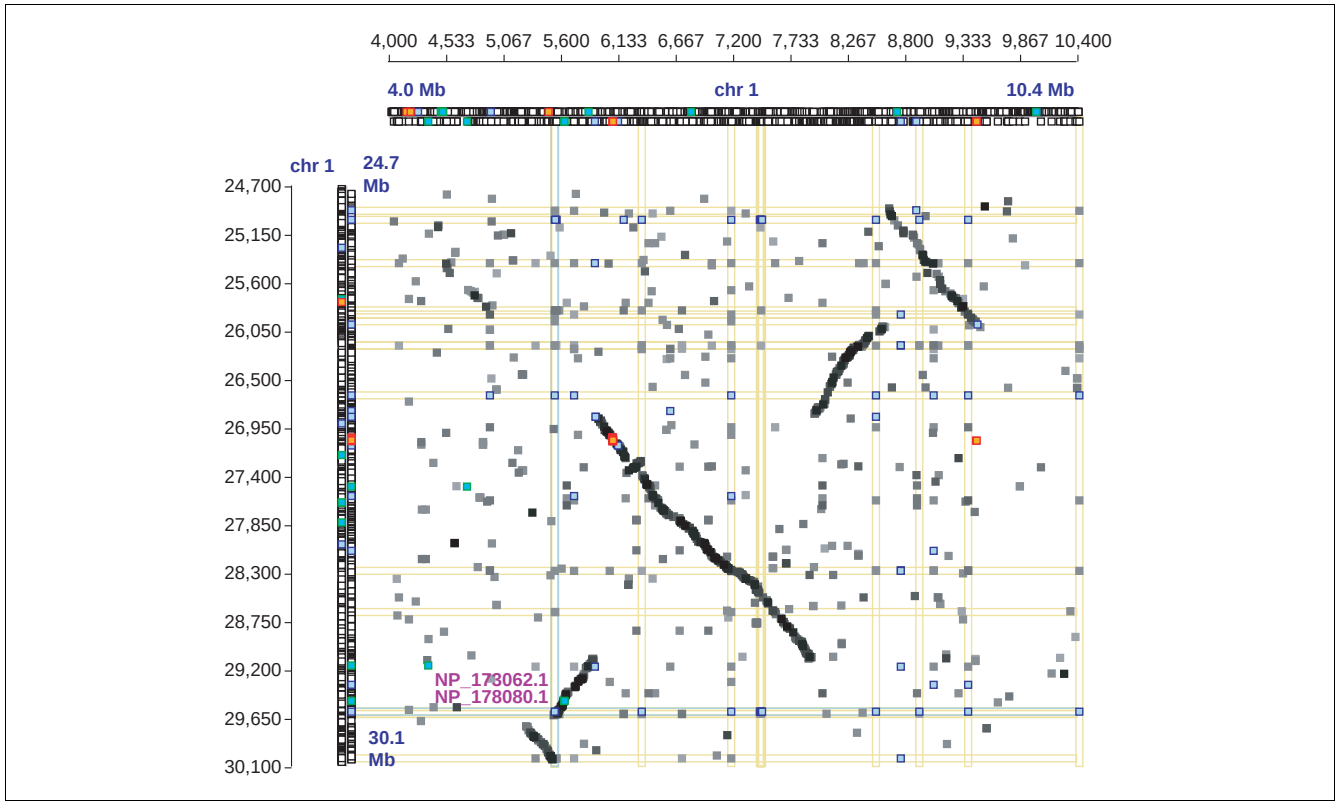

Figure 6
The same region of *Arabidopsis* genome as in Figure 1a viewed by GenoPix2D. Three large gene families, the NBS (nucleotide-binding site)-containing resistance gene candidates [29], the cytochrome P450s [30], and the LRR (leucine-rich repeat)-protein kinases [31] are highlighted in red, green and blue respectively. Blue crossed lines show an example of what is seen when a gene hit is selected: all homologs to those two genes are highlighted with vertical or horizontal yellow lines.

Conclusions

DiagHunter is a cross-platform program that efficiently makes genome-scale comparisons and effectively predicts and reports large-scale homology regions. Its advantages include the ability to work with similarity 'hit' matrices from essentially any source. It is particularly tuned and suited for comparisons based on predicted proteins, for which strand orientations can be identified. Interactive visualization by GenoPix2D allows exploration of homology regions and of groups of related genes. DiagHunter is freely available at [27] and both DiagHunter and GenoPix2D are available together at [32].

Acknowledgements

We thank Andy Baumgarten and Georgiana May for helpful conversations during development. Work on DiagHunter was supported by a USDA National Needs fellowship, a University of Minnesota Plant Molecular Genetics Institute fellowship to S.C., and NSF award DBI-0110206. Work on GenoPix2D was supported by NSF Plant Genome Grant DBI-9975971.

References

1. **PipMaker and MultiPipMaker** [http://bio.cse.psu.edu/pipmaker]
2. Schwartz S, Zhang Z, Frazer KA, Smit A, Riemer C, Bouck J, Gibbs R, Hardison R, Miller W: **PipMaker - a web server for aligning two genomic DNA sequences.** *Genome Res* 2000, **10:**577-586.
3. **The MUMmer home page** [http://www.tigr.org/software/mummer]
4. Delcher AL, Phillippy A, Carlton J, Salzberg SL: **Fast algorithms for large-scale genome alignment and comparison.** *Nucleic Acids Res* 2002, **30:**2478-2483.
5. Lefebvre A, Lecroq T, Dauchel H, Alexandre J: **FORRepeats: detects repeats on entire chromosomes and between genomes.** *Bioinformatics* 2003, **19:**319-326.
6. Kurtz S, Choudhuri JV, Ohlebusch E, Schleiermacher C, Stoye J, Giegerich R: **REPuter: the manifold applications of repeat analysis on a genomic scale.** *Nucleic Acids Res* 2001, **29:**4633-4642.
7. **REPuter** [http://bibiserv.techfak.uni-bielefeld.de/reputer]
8. Vandepoele K, Saeys Y, Simillion C, Raes J, Van De Peer Y: **The automatic detection of homologous regions (ADHoRe) and its application to microcolinearity between *Arabidopsis* and rice.** *Genome Res* 2002, **12:**1792-1801.
9. Tesler G: **GRIMM: genome rearrangements web server.** *Bioinformatics* 2002, **18:**492-493.
10. Pevzner P, Tesler G: **Genome rearrangements in mammalian evolution: lessons from human and mouse genomes.** *Genome Res* 2003, **13:**37-45.
11. Schwartz S, Kent WJ, Smit A, Zhang Z, Baertsch R, Hardison RC, Haussler D, Miller W: **Human-mouse alignments with BLASTZ.** *Genome Res* 2003, **13:**103-107.
12. Ma B, Tromp J, Li M: **PatternHunter: faster and more sensitive homology search.** *Bioinformatics* 2002, **18:**440-445.
13. Bowers JE, Chapman BA, Rong J, Paterson AH: **Unravelling angiosperm genome evolution by phylogenetic analysis of chromosomal duplication events.** *Nature* 2003, **422:**433-438.
14. Clamp M, Andrews D, Barker D, Bevan P, Cameron G, Chen Y, Clark L, Cox T, Cuff J, Curwen V, *et al.*: **Ensembl 2002: accommodating comparative genomics.** *Nucleic Acids Res* 2003, **31:**38-42.
15. Ku HM, Vision T, Liu J, Tanksley SD: **Comparing sequenced segments of the tomato and *Arabidopsis* genomes: large-scale duplication followed by selective gene loss creates a network of synteny.** *Proc Natl Acad Sci USA* 2000, **97:**9121-9126.
16. Ziolkowski PA, Blanc G, Sadowski J: **Structural divergence of chromosomal segments that arose from successive duplication events in the *Arabidopsis* genome.** *Nucleic Acids Res* 2003, **31:**1339-1350.
17. Vandepoele K, Raes J, De Veylder L, Rouze P, Rombauts S, Inze D: **Genome-wide analysis of core cell cycle genes in *Arabidopsis*.** *Plant Cell* 2002, **14:**903-916.
18. Vision TJ, Brown DG, Tanksley SD: **The origins of genomic duplications in *Arabidopsis*.** *Science* 2000, **290:**2114-2117.
19. Blanc G, Hokamp K, Wolfe KH: **A recent polyploidy superimposed on older large-scale duplications in the *Arabidopsis* genome.** *Genome Res* 2003, **13:**137-144.
20. Kozik A, Kochetkova E, Michelmore R: **GenomePixelizer - a visualization program for comparative genomics within and between species.** *Bioinformatics* 2002, **18:**335-336.
21. **GenomePixelizer: genome visualization tool** [http://www.atgc.org/GenomePixelizer]
22. Blanc G, Barakat A, Guyot R, Cooke R, Delseny M: **Extensive duplication and reshuffling in the *Arabidopsis* genome.** *Plant Cell* 2000, **12:**1093-1101.
23. Simillion C, Vandepoele K, Van Montagu MC, Zabeau M, Van de Peer Y: **The hidden duplication past of *Arabidopsis thaliana*.** *Proc Natl Acad Sci USA* 2002, **99:**13627-13632.
24. Altschul SF, Madden TL, Schaffer AA, Zhang J, Zhang Z, Miller W, Lipman DJ: **Gapped BLAST and PSI-BLAST: a new generation of protein database search programs.** *Nucleic Acids Res* 1997, **25:**3389-3402.
25. Stajich JE, Block D, Boulez K, Brenner SE, Chervitz SA, Dagdigian C, Fuellen G, Gilbert JG, Korf I, Lapp H, *et al.*: **The Bioperl toolkit: Perl modules for the life sciences.** *Genome Res* 2002, **12:**1611-1618.
26. **GD.pm - interface to graphics library** [http://stein.cshl.org/WWW/software/GD]
27. **Steven Cannon's webpage: software** [http://www.tc.umn.edu/~cann0010/Software.html]
28. **Tcl Developer Xchange** [http://tcl.activestate.com]
29. Meyers BC, Kozik A, Griego A, Kuang H, Michelmore RW: **Genome-wide analysis of NBS-LRR-encoding genes in *Arabidopsis*.** *Plant Cell* 2003, **15:**809-834.
30. Paquette SM, Bak S, Feyereisen R: **Intron-exon organization and phylogeny in a large superfamily, the paralogous cytochrome P450 genes of *Arabidopsis thaliana*.** *DNA Cell Biol* 2000, **19:**307-317.
31. Iten M, Hoffmann T, Grill E: **Receptors and signalling components of plant hormones.** *J Recept Signal Transduct Res* 1999, **19:**41-58.
32. **Genome Pixelizer 2D Plotter** [http://www.atgc.org/GenoPix_2D_Plotter]