Bioinformatics Advance Access published June 9, 2006

tuple_plot: Fast pairwise nucleotide sequence comparison with noise suppression

Karol Szafranski*, Niels Jahn and Matthias Platzer

Genome Analysis, Leibniz Institute for Age Research - Fritz Lipmann Institute, Beutenbergstr. 11, 07745 Jena, Germany

Associate Editor: Martin Bishop

ABSTRACT

Summary: The program tuple_plot identifies and visualizes local similarities between two genomic sequences, typically 100 kbp or longer, by applying the well-known dotplot principle. A dictionary of sequence words built from the input sequences serves to construct a task-specific expectancy model that is used to attribute significance values to pairwise word hits. The dictionary-based approach allows fast computation, the computation time scaling to O(N log N), depending on the size of the input sequences. The proposed scoring scheme appreciably increases the signal-to-noise ratio and may help to improve other word-based sequence comparison approaches.

Availability: tuple_plot is available at http://genome.fli-leibniz.de/software.html and may be used under GNU public license.

Contact: szafrans@fli-leibniz.de

1 INTRODUCTION

Dotplot analysis has been used for several decades as a technique for first-pass pairwise sequence comparison and its visualization (Fitch, 1969; Gibbs and McIntyre, 1970). It continues to be valuable since it principally allows for fast computations and is independent of assumptions that typically underlie alignment algorithms. Originally, a dot was drawn into a two-dimensional representation of the comparison space for any matching pair of sequence residues. In order to avoid an excess of signal in comparisons of larger sequences it has proven useful to represent rectangular sections of the comparison space by single image pixels and blacken these pixels if the number of matching residues exceeds some threshold value (Staden, 1982). The program DOTTER allows to vary such a stringency cutoff through an interactive front end (Sonnhammer and Durbin, 1995). Another strategy to reduce noise is to require consecutive matches, i.e. matches of sequence words (tuples) rather than single residues, between the two sequences (Maizel and Lenk, 1981).

In practice, dot-matrix based sequence comparison often yields undesirably low signal-to-noise ratios, especially if the input sequences are very large (>10 kb) and only moderately similar. Here, we define "noise" as the portion of hit signal (local similarity) which occurs independent of any homology relation between the sequences. We followed the intuition that noise in word-based sequence comparisons mostly arises from overabundant words. By applying a background hit model, implemented in a novel dot matrix comparison tool tuple_plot, we show that dotplot sequence comparisons can produce results with a significantly reduced noise level. A convenient aspect of our solution is that the background model is derived from the input sequences alone.

*To whom correspondence should be addressed.

2 ALGORITHM AND IMPLEMENTATION

tuple_plot performs nucleotide sequence comparisons in three steps: (i) preprocessing and parametrization of the comparison task, (ii) analysis of the sequences, and (iii) the actual dot matrix translation procedure. Initially, the program finds suitable parameters for the comparison, depending on the input sequences and program options. An adequate word size w is chosen such that approximately one random occurrence of any word t_i is expected throughout the two input sequences S_1 and S_2, with lengths l_1 and l_2, respectively.

$$E(n_i) = (l_1+l_2) / 4^w \;\wedge\; E(n_i) = 1 \qquad (1)$$

$$w = \mathrm{ceil}(\log_4(l_1+l_2)) \qquad (2)$$

The second step includes an analysis of the sequence composition of S_1 and S_2. All occurrences of a given word are stored in a list. The resulting dictionary of sequence words, organized as a hash table, serves several purposes throughout the algorithm. (i) Highly overrepresented words will be deleted from the dictionary in order to ignore these during match sampling. This way, we avoid wasting computation time on words which are uninformative for homology search (Ning *et al.*, 2001). (ii) For all the words in the dictionary the known number of word occurrences determines the number of word matches that will occur. These values will be used to score matches of any word t_i during match sampling. (iii) The actual positions of words are used to sample the word matches between sequences S_1 and S_2. These identified matches will be fed to a matrix that holds the cumulative local score sums.

In order to obtain accurate background estimates for the rectangular comparison space we analyze the sequence composition of both input sequences. The diagnostic significance of observed word matches is inferred as follows: $n_{i,1}$ and $n_{i,2}$ occurrences of word t_i found in the sequences will automatically result in $n_{i,1} \cdot n_{i,2}$ word hits. But, at most $\min(n_{i,1},n_{i,2})$ hits originate from a homology relation between the two sequences. The proposed scoring scheme uses these measures to attribute a score value to any observed hit for tuple t_i.

$$s_i = \min(n_{i,1},n_{i,2}) / (n_{i,1} \cdot n_{i,2}) = 1 / \max(n_{i,1},n_{i,2}) \qquad (3)$$

The scores are applied in the third computation step, the construction of the dot matrix.

The overall computational effort is approximately O(l w) for the construction of the tuple dictionary, $O(4^w)$ proportional to O(l) for the background correction, and $O(l_2 w)$ for the computation of the

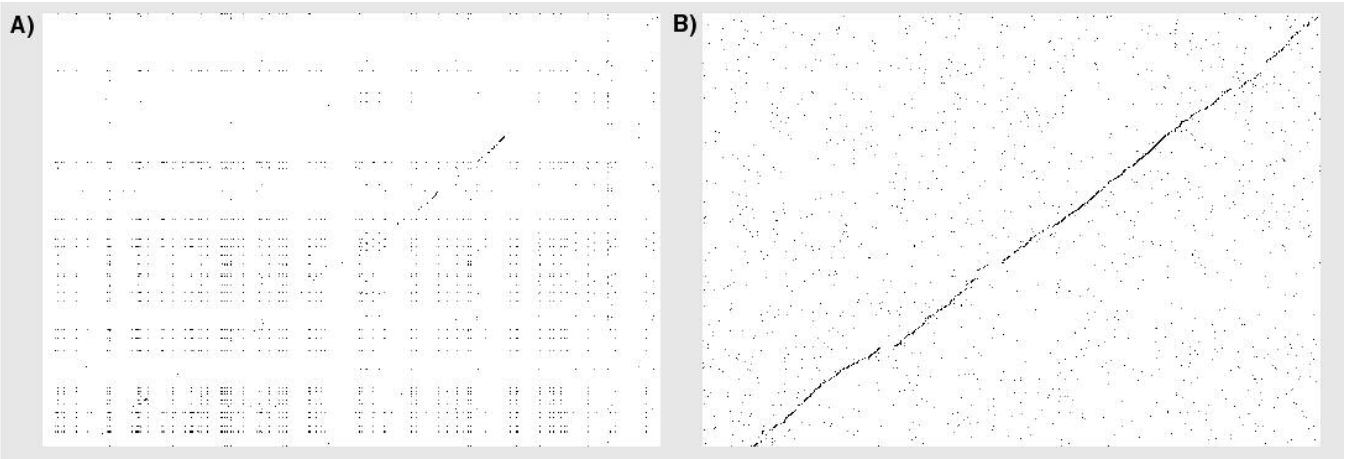

Fig. 1. Dotplots of the FOS gene loci from *H. sapiens* (acc.nos. AL691403 AF111167, 200 kb, horizontal) and *F. catus* (acc.no. AC118538, 140 kb, vertical) based on 10 nt word hits. In order to allow direct comparison of the different methods the score threshold is adjusted such that 1.0% of the image area is turned to hit state. **A**) Results obtained from the classical dot plot approach, i.e. tuple_plot without the noise suppression technique. **B**) Results obtained with tuple_plot.

dot matrix. With w being roughly proportional to log l the upper bound of the overall computational time scales to O(l log l). The program tuple_plot is written in C++ and has been extensively tested under the operating systems Solaris and Linux. It should work under any system which provides an ANSI-conforming C++ compiler. The command line interface of tuple_plot provides various options that influence the sequence comparison or the display of its results. The results are returned in PNG and HTML format. Forward and reverse matches are distinguishable by separate colors used for plotting.

We note that the proposed strategy for noise suppression can be applied to several other word-based sequence comparison approaches, like BlastZ (Schwartz *et al.*, 2003), SSAHA (Ning *et al.*, 2001), MUMmer (Delcher *et al.*, 2002), or LAGAN (Brudno *et al.*, 2003). We anticipate that it will help to shorten computation times and shift reported sequence similarities towards actual sequence homologies.

3 RESULTS

Initial trials with a naïve dotplot approach applied to large-size sequence comparison (BACs with insert sizes of 140 and 200 kb) showed that a tremendous portion of noise arises from microsatellite and other repetitive sequence elements (fig. 1A). With the proposed suppression technique, noise signal is reduced to an acceptable amount and shows a homogeneous distribution (fig. 1B). As shown, the increased specificity of the dotplot approach allows to discern similarities between intergenic sequences, well above noise level. Within a few CPU seconds, tuple_plot completes a 140x200 kb comparison of BAC sequences. tuple_plot has been tested on a variety of input sequences and, using a range of parameters, it produces highly robust reports on sequence similarities.

ACKNOWLEDGEMENTS

We are much indebted to U. Gausmann, G. Glöckner and K. Huse for testing developmental versions of tuple_plot. We also thank J. Sühnel and R. Sinha for helpful discussions. This work was supported by the German Ministry of Education and Research, grant 0312704E.

REFERENCES

Brudno,M. *et. al* (2003) LAGAN and multi-LAGAN: efficient tools for large-scale multiple alignment of genomic DNA. Genome Res. **13**, 721-731.

Delcher,A.L. (2002) Fast algorithms for large-scale genome alignment and comparison. Nucl. Acids Res. **30**, 2478-2483.

Fitch,W.M. (1969) Locating gaps in amino acid sequences to optimize the homology between two proteins. Biochem. Genet. **3**, 99-108.

Gibbs,A.J., and McIntyre,G. (1970) The diagram, a method for comparing sequences. Its use with amino acid and nucleotide sequences. Eur. J. Biochem. **16**, 1-11.

Maizel,J.V.J., and Lenk,R.P. (1981) Enhanced graphic matrix analysis of nucleic acid and protein sequences. Proc. Natl. Acad. Sci. U.S.A. 78, 7665-7669.

Ning,Z. *et. al.* (2001) SSAHA: a fast search method for large DNA databases. Genome Res. **11**, 1725-1729.

Schwartz,S. *et. al.* (2003) Human-mouse alignments with BLASTZ. Genome Res. **13**, 103-107.

Sonnhammer,E.L., and Durbin,R. (1995) A dot-matrix program with dynamic threshold control suited for genomic DNA and protein sequence analysis. Gene **167**, GC1-GC10.

Staden,R. (1982) An interactive graphics program for comparing and aligning nucleic acid and amino acid sequences. Nucl. Acids Res. **10**, 2951-2961.