Supplementary Material

METHODS

Overview

In a preprocessing step, one must first build an index structure for the alignment computation. This is done by translating the protein reference sequence database into pDNA and then building an index for these sequences. In our implementation of PAUDA, this step is performed by a script called `pauda-build`, which uses a program called `protein2pna` to compute the pDNA sequences and then runs the program `bowtie2-build` (Langmead and Salzberg, 2012) to construct the index.

Now, assume that we are given a file of DNA sequencing reads. Using a script called `pauda-run`, first the reads are translated into protein sequences and converted into pDNA using a program called `dna2pna`. The program `bowtie2-align` (Langmead and Salzberg, 2012) is then employed to align the pDNA query sequences against the pDNA reference database. Finally, all pDNA alignments are translated back into protein-protein alignments, the scores of the alignments are computed, and then all high-scoring protein-protein alignments are written to a file in BLASTX format, using a program called `pna2blastx`, see Figure **??**.

We would like to emphasize that, from a user perspective, PAUDA consists only of two commands, `pauda-build` and `pauda-run`. Under the hood, however, these two commands invoke a number of different scripts and programs, as described in the following.

Details of pauda-build

In our studies we used the KEGG database (Kanehisa and Goto, 2000) as protein reference database. The version used was downloaded from the KEGG website in June 2011 and contains just under 7 million protein sequences.

Using the program `proteins2pna`, each of the sequences is converted into pDNA using the alphabet reduction to four letters described in (Murphy *et al.*, 2000). To achieve this, the 20 amino acids are partitioned into four classes $\{L, V, I, M, C\}$, $\{A, G, S, T, P\}$, $\{F, Y, W\}$ and $\{E, D, N, Q, K, R, H\}$ and these are mapped onto the four letters A, C, G and T, respectively. All other characters are mapped to the letter N. This is called the *BLOSUM50* reduction. Our implementation also supports the GBMR4 alphabet reduction described in (Peterson *et al.*, 2009).

After reduction of the reference sequences, all duplicate or contained pDNA sequences are removed, obtaining a database that has about 25% fewer entries, which is a crucial step toward accelerating the analysis and also ensuring that conserved sequences are not deemed "repetitive" by the employed DNA aligner. The pDNA reference sequences are written to a file in FASTA format. The program also creates an auxiliary file to facilitate the expansion of the computed matches back to the full reference proteins. The program requires about three hours to process the KEGG database.

After preprocessing the protein reference database as described above, the tool `bowtie2-build` is used to construct a Bowtie2 reference index, as described in the user manual of Bowtie2 (Langmead and Salzberg, 2012).

Details of pauda-run

The task of translating DNA sequencing reads into pDNA sequences is performed by the program `dna2pna`. It first translates each DNA sequencing read using all six reading frames into protein sequences using the standard bacterial genetic code. As an option, one can restrict the output of the program to include only those translations of a given DNA sequence that have the longest run of non-stop codons among all six reading frames. Our implementation uses a standard sequence complexity filter to avoid matches involving low complexity sequences (Wootton and Federhen, 1993; Mount, 2007). Finally, all protein sequences are processed using an alphabet reduction as described above and written as pDNA to a file in FASTA format. The header line of each FASTA record contains information identifying which original read and reading frame were used to generate the pDNA sequence. The program `dna2pna` is written in Java and it takes about one minute to process one million reads of length 100.

We then run the program `bowtie2-align` to compare a file of pDNA sequences against the PNA index.

The choice of program parameters has a significant impact on the performance of the PAUDA approach, allowing a trade-off between speed and sensitivity. In this study, we used the following Bowtie2 parameters: `--reorder --sam-no-hd --mm -t -f -p 4 -k 5 -L 18 -N 0 --mp 3 --score-min C,40 -i C,1 --local --norc`. The most important parameters here are `L=18` and `N=0`, which instruct Bowtie2 to use a seed-length of 18 and allow 0 mismatches within a seed; `mp=3` and `score-min=40`, which specify that mismatches have penalty 3 and the minimum score of an alignment is 40, respectively; and `k=5`, which specifies that up to 5 alignments should be reported for each read. We call this set of parameters the *fast* setting. It is the default setting for `pauda-run` and call also be explicitly invoked using the flag `--fast`.

Our software currently supports a second setting, `--slow`, which aims at achieving a higher sensitivity, albeit at a slower assignment rate. In this case, we set `N=1`, `min-score=35` and `k=30`, while keeping all other parameters the same. PAUDA-slow is invoked by passing the flag `--slow` to the script `pauda-run`. On the data presented in Table **??**, PAUDA-slow achieved a read assignment rate of 48% of that of BLASTX, with a speed-up of 750.

Once the Bowtie2 analysis has completed, the program `pna2blastx` is run on the output of Bowtie2. For each pDNA alignment in the input file, the program constructs the corresponding protein-protein alignments, compute their scores and then outputs all protein alignments in BLASTX format that are deemed significant, that is, whose bit-score exceeds a specified value (30, by default). Scoring is performed by first computing a raw score using the BLOSUM62 matrix (Henikoff and Henikoff, 1996), a default gap score of 11 and a default extension score of 1, and then determining a bit score and expected value as in BLASTX (Altschul *et al.*, 1990). The program `pna2blastx` is written in Java. It requires a few minutes to process the output of Bowtie2 for an original input dataset of one million Illumina reads. The choice of scoring matrix and gap scores can be changed by editing the `pna2blastx` file.

Optionally, one can request that the `pauda-run` script also generates an RMA file for direct input into the metagenome analysis

program MEGAN (Huson *et al.*, 2011), version 5. This feature uses a Java program called `blastx2rma`.

It has not escaped our notice that an alternative approach to determining the significance of a match can be based on the observation that any pDNA match involving the reverse complement of either matched sequence is a false positive match because the reverse complement of a pDNA sequence is nonsense. Hence, such matches provide a distribution of random matches that can be used to compute a p-value and also an expected value.

REFERENCES

Altschul, S. F., Gish, W., Miller, W., Myers, E. W., and Lipman, D. J. (1990). Basic local alignment search tool. *Journal of Molecular Biology*, **215**, 403–410.

Henikoff, J. G. and Henikoff, S. (1996). Blocks database and its applications. *Methods Enzymol*, **266**, 88–105.

Huson, D. H., Mitra, S., Weber, N., Ruscheweyh, H.-J., and Schuster, S. C. (2011). Integrative analysis of environmental sequences using MEGAN 4. *Genome Research*, **21**, 1552–1560.

Kanehisa, M. and Goto, S. (2000). KEGG: Kyoto encyclopedia of genes and genomes. *Nucleic Acids Res*, **28**(1), 27–30.

Langmead, B. and Salzberg, S. (2012). Fast gapped-read alignment with Bowtie 2. *Nat Meth*, **9**(4), 357–359.

Mount, D. W. (2007). Using the basic local alignment search tool (BLAST). *Cold Spring Harbor Protocols*, **2007**(7), pdb.top17.

Murphy, L. R., Wallqvist, A., and Levy, R. M. (2000). Simplified amino acid alphabets for protein fold recognition and implications for folding. *Protein Engineering*, **13**, 149–152(4).

Peterson, E. L., Kondev, J., Theriot, J. A., and Phillips, R. (2009). Reduced amino acid alphabets exhibit an improved sensitivity and selectivity in fold assignment. *Bioinformatics*, **25**(11), 1356–1362.

Wootton, J. and Federhen, S. (1993). Statistics of local complexity in amino acid sequences and sequence databases. *Computers & Chemistry*, **17**(2), 149–163.