Vol. 15 no. 12 1999
Pages 1058–1059

Shufflet: shuffling sequences while conserving the *k*-let counts

Eivind Coward

Laboratoire Génome et Informatique, Université de Versailles Saint-Quentin-en-Yvelines, 45 avenue des Etats-Unis, 78035 Versailles cedex, France

Received on May 5, 1999; revised on June 17, 1999; accepted on June 28, 1999

Abstract

Summary: *Shufflet is a program and a web-application that generates fast random shufflings of sequences (DNA, protein or others), conserving the exact k-let counts for a given k. The sequences are sampled uniformly from all the valid permutations.*

Availability: *As a web application: http://www.genetique.uvsq.fr/eivind/shufflet.html. Source code is available on request for academic use.*

Contact: *coward@genetique.uvsq.fr*

Shuffled DNA or protein sequences are often used in Monte Carlo methods for obtaining statistical significance for properties of the observed sequence, such as sequence similarity (Lipman *et al.*, 1984; Comet *et al.*, 1999). It is incorporated in sequence analysis programs such as RDF (Lipman and Pearson, 1985) and Bestfit (Devereux, 1989). However, a simple shuffling of the letters obscures the frequencies of dinucleotides, trinucleotides (or -peptides) etc., which are often biased, properties that one might wish to conserve for the shuffled sequences. The program Shufflet presented here generates random shufflings that conserve the exact counts of all words equal to or shorter than a given length k. Such words will be referred to as k-lets, or singlets, doublets, triplets etc.

As input, Shufflet accepts one or more sequences in Fasta format (each sequence preceded by a header beginning with >), or a single sequence as plain text. Using the web page, the input can be given as a file or pasted directly into the form. The desired number of shuffled sequences are output in Fasta format.

For example, for $k = 2$, the sequence ACTAGT permits the doublet-preserving permutation AGTACT, but no others except the original sequence. Of course, longer sequences usually permit many permutations. Note that a high value of k severely restricts the number of valid permutations, and for short sequences one will often end up with the 'shuffled' sequence equal to the original one. Very roughly, the sequence should be longer than to the order of 4^k bp (DNA) or 20^k aa (protein), corresponding to one expected occurrence of each possible k-let in the random uniform case.

Note that the first $k - 1$ letters are the same in all the shufflings, as well as the last $k - 1$ letters. This is a consequence of the conservation of k-lets. For example, for $k = 2$, if the sequence begins with an A (and does not end with an A) it follows that the number of doublets beginning with an A is one higher than the number of doublets ending with an A. This property can only hold if the whole sequence begins with an A.

When comparing DNA sequences, the similarity between random shufflings of the respective sequences is often much larger if one conserves the dinucleotide frequencies (Fitch, 1983). Thus, using simple letter shufflings can give statistical significance to similarities that can be explained solely by the dinucleotide bias, which is not what we want. An example demonstrating this effect on alignment scores for two short DNA sequences can be found on the web page.

The choice of k, or how many neighbour constraints to put on the sequences, is not evident and will vary from case to case. Whatever choice is made, an important requirement for the Monte Carlo method to work properly is that the random shufflings are sampled uniformly from the set of all possible shufflings, that is, every valid permutation should appear with the same probability.

The need for such constrained shufflings has been pointed out by several authors, and different algorithms have been proposed. Fitch (1983) described a method for doublet-conserving shuffling, based on Euler paths on directed graphs, but it does not give a uniform sample of valid permutations. Altschul and Erickson (1985) presented a more elaborate algorithm, also based on Euler paths, which satisfies the requirement of uniform sampling. Kandel *et al.* (1996) improve a crucial step, permitting larger k and larger alphabets (protein instead of DNA), and they prove that their algorithm produces a uniform sample. A completely different approach is the iterative *swapping method* (Unger *et al.*, 1986; Kandel *et al.*, 1996), which produces a uniform sample asymptotically.

An alternative to shuffled sequences is the use of sequences generated by Markov chains of order $k-1$, whose transition probabilities are estimated from the original sequence. This method is probably more used, because it is simpler to implement, but it conserves only the *expected* frequencies of k-lets, not the exact counts. The Markov method will often give results similar to shuffling for long sequences, but Altschul and Erickson (1985) demonstrate that there may be notable differences. The shuffling method has the advantage of providing a simply defined pool for our random sequences, and it is a direct generalisation of the commonly used simple letter shuffling, while the Markov method is not. See also the discussion by Fitch (1983).

In spite of the importance of shuffling in many applications and the fact that these algorithms have been known for years, there are to the author's knowledge no implementations readily available to the public. The web-application presented here is an attempt to fill the gap. It is based on the Euler algorithm of Kandel *et al.* (1996), which has the advantages of being fast (essentially linear in time with respect to sequence length) and producing a uniform sample in a finite number of steps. The algorithm is based on constructing Euler paths on a directed graph, whose vertices represent the distinct $(k-1)$-lets, and whose edges represent all the k-lets. The graph is represented in memory simply as a table of k-let counts, which is much more space-efficient (for $k > 2$) than the representation proposed by Kandel *et al.* A description can be found on the web pages. See also Kandel *et al.* (1996) for more details and proofs.

Usually, conservation of k-lets implies conservation of shorter words. However, this may not hold if the sequence begins and ends with the same $(k-1)$-let. Such sequences are called cyclic by Kandel *et al.* For example (for $k = 3$) the sequences ATAT and TATA have the same triplet counts, but different doublet counts. The original algorithm by Kandel *et al.* (1996) only requires conservation of k-lets, thus permitting such shufflings. I have chosen the more restrictive conservation of words of length k and shorter (in accordance with Altschul and Erickson, 1985), believing that this is usually wanted.

For a fixed k, and assuming a fixed number of different k-lets, the algorithm is linear in time and space with respect to sequence length. Because of the graph size, it is exponential with respect to k, but this is not a problem for practical values of k. On our web server, a Digital DEC/Alpha 2100, 100 shufflings of a 10 kb DNA sequence (1 Mb output) with $k = 6$, take about 2.5 s, and 100 shufflings of a 1000 aa protein sequence with $k = 3$ take less than 1 s.

An essential but often neglected part of stochastic simulation is the random number generator. A bad pseudo-random number generator can severely bias the results, in spite of theoretical guarantees of uniform sampling. Rather than relying on an implementation-dependent generator (some of them have turned out to be unsatisfying), I have used the one of Ripley (1987), a standard linear congruential generator that is documented to behave well to statistical tests. The current seed is saved in a file between each run, effectively making one long series of pseudo-random numbers.

Shufflet is written in C and can be run independently of the web interface. For academic users, the source code is available for compilation on any C platform or incorporation into other programs.

Acknowledgements

I would like to thank Jean-Loup Risler and Alex Grossmann for helpful comments. This work was supported by grant 125679/410 from the Norwegian Research Council.

References

Altschul,S. and Erickson,B. (1985) Significance of nucleotide sequence alignments: a method for random sequence permutation that preserves dinucleotide and codon usage. *Mol. Biol. Evol.*, **2**, 526–538.

Comet,J.P., Aude,J.C., Glémet,E., Risler,J.L., Hénaut,A., Slonimski,P.P. and Codani,J.J. (1999) Significance of Z-value statistics of Smith-Waterman scores for protein alignment. *Comput. Chem.*, in press.

Devereux,J. (1989) *The GCG sequence analysis software package*. Version 6.0. Genetics Computer Group, Inc., Madison, WI.

Fitch,W.M. (1983) Random sequences. *J. Mol. Biol.*, **163**, 171–176.

Kandel,D., Matias,Y., Unger,R. and Winkler,P. (1996) Shuffling biological sequences. *Discrete Appl. Math.*, **71**, 171–185.

Lipman,D.J. and Pearson,W.R. (1985) Rapid and sensitive protein similarity searches. *Science*, **227**, 1435–1441.

Lipman,D.J., Wilbur,W.J., Smith,T.F. and Waterman,M.S. (1984) On the statistical significance of nucleic acid similarities. *Nucleic Acids Res.*, **12**, 215–226.

Ripley,B.D. (1987) *Stochastic Simulation*. Wiley, New York.

Unger,R., Avrahami,G., Harel,D. and Sussmann,J.L. (1986) Simple general shuffling scheme which preserves fragment frequencies up to any required length. *Proceedings of the Macromolecules, Genes, and Computers Conference*.