BIOINFORMATICS APPLICATIONS NOTE Vol. 28 no. 24 2012, pages 3318–3319
doi:10.1093/bioinformatics/bts607

Genome analysis

Advance Access publication October 15, 2012

MULTOVL: fast multiple overlaps of genomic regions

András Aszódi*

Scientific Computing Core Facility, Campus Science Support Facilities, 1030 Vienna, Austria

Associate Editor: Alfonso Valencia

ABSTRACT

Summary: We present the MULTOVL application suite that detects and statistically analyses multiple overlaps of genomic regions in a fast and efficient manner. The package supports the detection of multiple region intersections, unions and 'solitary' genomic regions. The significance of actually observed overlaps is estimated by comparing them with empirical null distributions generated by random shuffling of the input regions.

Availability and implementation: Source code and binaries are downloadable from: http://www.csf.ac.at/facilities/scc/tools.

Contact: andras.aszodi@csf.ac.at

Received on May 30, 2012; revised on September 29, 2012; accepted on October 8, 2012

1 THE MULTOVL ALGORITHM

1.1 Rationale

Overlaps between genomic regions play an important role in bioinformatics. Most new annotations are based on finding overlaps between a given genomic region and already known features such as transcripts, introns, exons, repeat regions, etc. Overlaps are also useful for finding correlations between experimental observations, for instance various enrichment regions identified by ChIP-Seq using different antibodies.

Although excellent tools have been developed to detect pairwise overlaps between two *tracks* (sets of genomic regions) (Quinlan and Hall, 2010), to the best of our knowledge, no solution to the generalized problem of finding all $N \geq 2$-fold overlaps between genomic regions belonging to $M > 2$ tracks has been published. This task cannot be accomplished efficiently by combining the results of pairwise overlaps. We developed a generic multiple overlap algorithm and implemented it in the MULTOVL set of programs.

MULTOVL was used to combine technical replicate tracks of ChIP-chip enrichment regions (Hekimoglu-Balkan *et al.*, 2012). By keeping only the triple overlaps from three input tracks, we could reduce the number of candidate regions by 30% and achieved a robust enrichment region assignment. Another practical application was to define regulatory regions in the genome of pro-B and mature-B cells by overlapping DNA-hypersensitive site peaks with CAGE tag clusters in several technical replicates (Revilla-i-Domingo *et al.*, 2012).

*To whom correspondence should be addressed.

1.2 The basic MULTOVL algorithm

The algorithm starts by saving the first and last coordinates (the *boundaries*) of the regions on the same chromosome in a sorted list. A *multiplicity counter* is then initialized to zero in a second pass and increased or decreased whenever a first or last position is encountered in the boundary list. The multiplicity is zero outside the regions, 1 over those regions that do not overlap, and in general N for N-fold overlaps (Fig. 1).

The runtime scaling of the algorithm is dominated by the $O(N \log N)$ complexity of the boundary sorting step. The multiplicity counting step scales as $O(N)$, where N is the number of input regions. Memory requirement also scales linearly with N.

The algorithm produces several kinds of output regions. In addition to the *N-fold overlaps* (intersections), it identifies *solitary regions*, which are input regions that do not overlap with any other region in the input dataset, and *unions* of input regions as well.

1.3 Multiple overlap probabilities

When analysing multiple overlaps between experimentally detected genomic regions, the question often arises whether the observed overlaps were owing to chance alone. We implemented the MULTOVLPROB tool to answer this question. MULTOVLPROB shuffles all (or some, see below) input genomic regions many times, runs the MULTOVL algorithm on each shuffled set and estimates empirical null distributions of the sum of overlap lengths for each overlap multiplicity. Using these null distributions, *P*-values and *Z*-scores for the actually observed total overlap lengths are calculated to decide whether the regions overlap significantly more often (or less) compared with what could be expected by chance. Tests with synthetic datasets indeed gave rise to low *P*-values and large positive (or negative) *Z*-scores (data not shown).

Certain chromosomal regions (repeats, unsequenced regions around telomeres and centromeres, etc.) cannot be mapped by most experimental methodologies. In such cases, shuffling shall be performed only within user-defined '*free regions*' where the input regions could be observed experimentally. Appropriate free regions should be defined also for features that cannot occur everywhere on a chromosome such as promoter regions.

MULTOVLPROB currently distributes the shuffled regions uniformly within the free regions. Non-uniform shuffling that could, for instance, preferably position shuffled regions near the 5′-ends of the free regions will be implemented in a future version of the tool.

The user may label input tracks *fixed* in which the regions will not be shuffled by MULTOVLPROB. This is useful for tracks

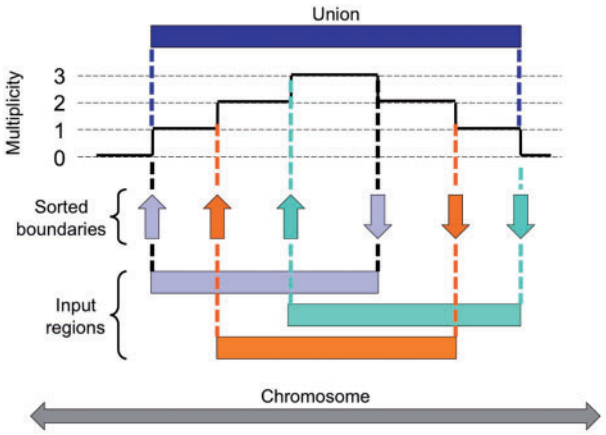

Fig. 1. The basic MULTOVL algorithm. In this example, 2- and 3-fold overlaps are identified in addition to a union of the three input regions

Table 1. Run-time performance of PARMULTOVLPROB measured on a Mac Pro (2× Xeon E5620 2.4 GHz CPUs), compiled with Apple's g++ 4.2.1, -O3 optimization

Number of threads	Average runtime (minutes:second)	Relative speedup
1	11:58.9	1.00
2	06:00.8	1.99
3	04:08.3	2.89
4	03:07.3	3.84
5	02:36.6	4.59
6	02:09.7	5.54
7	01:54.3	6.29
8	01:48.7	6.61

Note: Input: eight tracks with 20 000 simulated regions each, 1000 reshufflings. Five individual runtime measurements were averaged after subtracting the time needed for parsing the input files.

containing genomic features with well-defined positions such as transcripts.

2. IMPLEMENTATION

The MULTOVL program suite is implemented in C++ to achieve maximal I/O and execution speed and minimal memory usage. We used only standard C++ features and the Boost libraries (http://www.boost.org).

2.1 Supported data formats

MULTOVL accepts BED or GFF-formatted input by default. The output file format is GFF providing references to the input regions participating in each overlap. MULTOVL can read BAM formatted input files if compiled with the BAMtools library (https://github.com/pezmaster31/bamtools).

The MULTOVL program can save its complete internal state by using the Boost serialization library. Another MULTOVL instance can read that state file, enabling the user to run the program many times with changed parameters on the same input tracks without re-parsing them again.

Another tool in the suite, PGMULTOVL, can read input tracks from and write result tracks to PostgreSQL relational databases. The mapping of the database contents to PGMULTOVL's concept of tracks and regions is freely configurable using a JSON-formatted file that specifies the download and upload SQL queries.

2.2 Parallelization

Parallelization is an attractive option for the MULTOVLPROB program that performs thousands of independent MULTOVL calculations to estimate the overlap null distributions. We implemented the PARMULTOVLPROB tool using a simple scheme in which worker threads perform region shuffling on independent copies of the input data and observed very good scaling up to eight concurrent threads (Table 1).

ACKNOWLEDGEMENTS

The author wishes to thank Roman Stocsits for his critical comments on the manuscript and his invaluable help with testing MULTOVL, and to Markus Jaritz and Johanna Trupke for their suggestions. Special thanks to Meinrad Busslinger for his continuous support.

Funding: Part of this work was sponsored by the Austrian Genome Research Programme of The Ministry for Science and Research (BMWF), contract no 820980 (EpigeneticControl).

Conflict of Interest: none declared.

REFERENCES

Hekimoglu-Balkan,B. *et al.* (2012) Intergenic Polycomb target sites are dynamically marked by non-coding transcription during lineage commitment. *RNA Biol.*, **9**, 314–325.

Quinlan,A.R. and Hall,I.M. (2010) BEDTools: a flexible suite of utilities for comparing genomic features. *Bioinformatics*, **26**, 841–842.

Revilla-i-Domingo,R. *et al.* (2012) The B-cell identity factor Pax5 regulates distinct transcriptional programmes in early and late B lymphopoiesis. *EMBO J.*, **31**, 3130–3146.