BIOINFORMATICS APPLICATIONS NOTE Vol. 29 no. 2 2013, pages 264–265
doi:10.1093/bioinformatics/bts664

Genome analysis

Advance Access publication November 17, 2012

VAGUE: a graphical user interface for the Velvet assembler

David R. Powell[1,2] and Torsten Seemann[1,2,*]

[1]Victorian Bioinformatics Consortium, Monash University, Clayton 3800, Australia and [2]Life Sciences Computation Centre, Victorian Life Sciences Computation Initiative, Parkville 3053, Australia

Associate Editor: Alfonso Valencia

ABSTRACT

Summary: Velvet is a popular open-source *de novo* genome assembly software tool, which is run from the Unix command line. Most of the problems experienced by new users of Velvet revolve around constructing syntactically and semantically correct command lines, getting input files into acceptable formats and assessing the output. Here, we present Velvet Assembler Graphical User Environment (VAGUE), a multi-platform graphical front-end for Velvet. VAGUE aims to make sequence assembly accessible to a wider audience and to facilitate better usage amongst existing users of Velvet.

Availability and implementation: VAGUE is implemented in JRuby and targets the Java Virtual Machine. It is available under an open-source GPLv2 licence from http://www.vicbioinformatics.com/.

Contact: torsten.seemann@monash.edu

Received on September 6, 2012; revised on October 10, 2012; accepted on November 8, 2012

1 INTRODUCTION

Velvet was one of the first *de novo* genome assemblers, which handled short read sequences from the original Illumina Genome Analyzer (Zerbino and Birney, 2008). It has become popular because it is relatively simple to install, has no dependencies, produces good results, runs fast and has a strong user community (Zhang *et al.*, 2011).

The main drawback of Velvet is its complex command line interface. Projects consisting of multiple sequence libraries are challenging to specify correctly. The user needs to provide the exact file format and compression type of the input sequence files. Standard paired-end read files need to be interleaved manually beforehand. To address these and other issues, we developed Velvet Assembler Graphical User Environment (VAGUE), a graphical user interface to Velvet.

2 DESCRIPTION

The purpose of the VAGUE graphical user interface is to simplify running Velvet properly. The application consists of one window with four tabs: *Setup*, *Advanced*, *Log* and *Results* (Fig. 1).

The active tab on starting VAGUE is *Setup*. It clearly lists the key parameters to be set, and Section 2.1 describes the choice of the defaults. If desired, all other Velvet parameters are accessible on the *Advanced* tab, which is populated dynamically by parsing the output of the Velvet executables, avoiding hard coding of any options and partially future-proofing VAGUE.

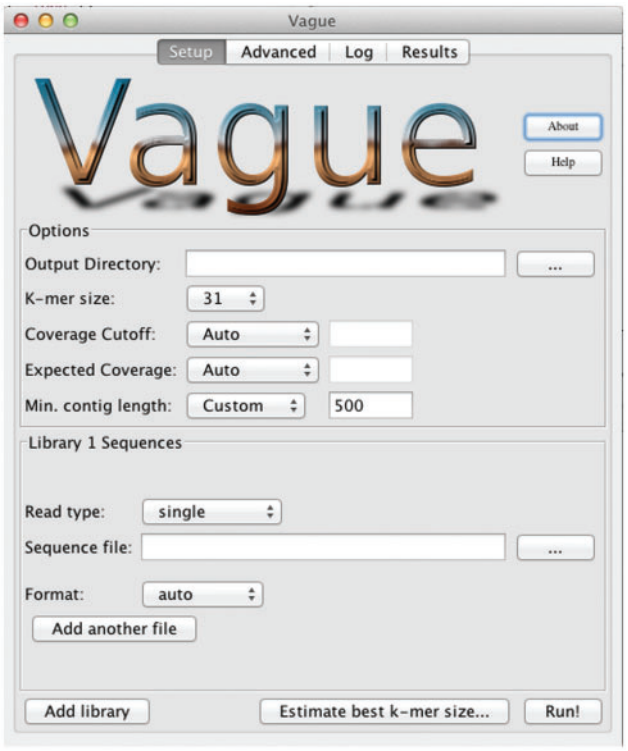

Fig. 1. Screenshot of VAGUE's Setup tab on Mac OS X

The input sequence files to assemble are also added on the *Setup* tab. If the files came from separately prepared DNA libraries, they should be placed in different *Library* sections via the *Add new library* button. As described in Section 2.2, this now supports non-interleaved paired-end reads and *bzip2* compression.

The final parameter to be supplied is the *k*-mer size. Most first time users of Velvet find this troublesome. In Section 2.3, we describe how VAGUE can auto-detect a reasonable *k* value.

The assembly process is launched by clicking the *Run* button at the bottom right corner. The view automatically switches to the *Log* tab, where the output of the *velveth* and *velvetg* commands can be observed while they are running behind the scenes.

When the assembly is complete, the view automatically changes to the *Results* tab. Here, all the contigs are listed and can be individually viewed in FASTA format. Gross statistics

*To whom correspondence should be addressed.

such as the minimum, maximum and N50 contig sizes are also displayed.

2.1 Better Velvet defaults

The Velvet software has evolved significantly since its first release, with many new features now considered standard. However, to maintain backward compatibility, many of these features remain turned off by default.

VAGUE changes the default values of two parameters, which improve greatly the quality of assembly produced. Both *-exp_cov* and *-cov_cutoff* are set to *auto* rather than left disabled. This allows for improved repeat resolution and more reliable contig sequences (Zerbino *et al*., 2009).

2.2 Velvet command line improvements

Velvet requires the user to provide the file format (fasta, fastq, etc.) and whether it is *gzip* compressed. We considered implementing auto-detection of these settings within VAGUE but opted instead to add new functionality to the Velvet source code. This also allowed us to add support for *bzip2* compression. The new *-fmtAuto* option is now part of standard Velvet and may be used by VAGUE to load input files.

Illumina paired-end reads are typically distributed as two separate files, one for the left mates and one for the right mates. Velvet requires that paired reads be provided in a single interleaved file. This is an unnecessary waste of disk space and temporary files. To resolve this drawback, we extended Velvet by adding the *-separate* option, which allows *pairs* of filenames to be supplied. By adding this feature to Velvet itself, rather than within VAGUE, the benefits extend to the whole Velvet community.

2.3 Estimating the *k*-mer size

The key parameter for any de Bruijn graph-based assembler is the word size k. Larger values of k increase specificity but reduce depth of coverage. The author of Velvet suggests a k-coverage of 20 is a good starting point for experimentation (Zerbino, 2010).

VAGUE can optionally select k based on the read files the user has provided, and an estimate of the target genome size. The input read files are scanned, and the k-coverage calculated for all possible k. The k value closest to a k-coverage of 20 is automatically chosen, but it can be adjusted as the user wishes. The authors have found this strategy to choose k at or close to the value they would have chosen via manual assessment.

3 CONCLUSIONS

The VAGUE package is free, open-source and easy to use. It runs on any platform, which supports the Java runtime environment. It comes pre-packaged with 64-bit Velvet binaries for both Linux and Mac OS X, but will preferentially use a local copy in the user's path if it is at least version 1.2.07 or higher.

VAGUE will introduce Velvet to a new audience and give existing users a friendlier alternative that saves time, reduces trial and error and ultimately improves the quality of *de novo* assemblies produced. Additionally, the new *-separate* command line option and *bzip2* support will save significant amounts of compute time and disk space.

ACKNOWLEDGEMENTS

The authors thank Daniel Zerbino for useful discussion and accepting our source code changes into the Velvet project; Tim Stinear, Dieter Bulach and Simon Gladman for testing VAGUE; and the reviewers of this manuscript for their helpful comments.

Funding: This research was supported in part by the Victorian Life Sciences Computation Initiative, an initiative of the Victorian Government hosted by the University of Melbourne, Australia.

Conflicts of Interest: none declared.

REFERENCES

Zerbino,D.R. and Birney,E. (2008) Velvet: algorithms for de novo short read assembly using de Bruijn graphs. *Genome Res*., **18**, 821–829.

Zerbino,D.R. *et al.* (2009) Pebble and rock band: heuristic resolution of repeats and scaffolding in the velvet short-read de novo assembler. *PLoS One*, **4**, e8407.

Zerbino,D.R. (2010) Using the Velvet de novo assembler for short-read sequencing technologies. *Curr. Protoc. Bioinformatics*, **Chapter 11**, Unit 11.5.

Zhang,W. *et al.* (2011) A practical comparison of de novo genome assembly software tools for next-generation sequencing technologies. *PLoS One*, **6**, e17915.