Bioinformatics, 2017, 1–3
doi: 10.1093/bioinformatics/btx366
Advance Access Publication Date: 9 June 2017
Applications Note

OXFORD

Sequence analysis

runBNG: a software package for BioNano genomic analysis on the command line

Yuxuan Yuan[1], Philipp E. Bayer[1], Huey-Tyng Lee[1,2] and David Edwards[1,*]

[1]School of Biological Sciences, University of Western Australia, Perth, WA, Australia and [2]School of Agriculture and Food Sciences, University of Queensland, Brisbane, QLD, Australia

*To whom correspondence should be addressed.

Associate Editor: John Hancock

Received on February 12, 2017; revised on May 18, 2017; editorial decision on June 2, 2017; accepted on June 6, 2017

Abstract

Summary: We developed runBNG, an open-source software package which wraps BioNano genomic analysis tools into a single script that can be run on the command line. runBNG can complete analyses, including quality control of single molecule maps, optical map *de novo* assembly, comparisons between different optical maps, super-scaffolding and structural variation detection. Compared to existing software BioNano IrysView and the KSU scripts, the major advantages of runBNG are that the whole pipeline runs on one single platform and it has a high customizability.
Availability and implementation: runBNG is written in bash, with the requirement of BioNano IrysSolve packages, GCC, Perl and Python software. It is freely available at https://github.com/appliedbioinformatics/runBNG.
Contact: dave.edwards@uwa.edu.au
Supplementary information: Supplementary data are available at *Bioinformatics* online.

1 Introduction

Next generation sequencing (NGS) technologies have advanced our understanding of biology. Most high-throughput sequencing technologies produce reads shorter than 300 bp (Goodwin *et al.*, 2016), yet these are not sufficient in resolving large and complex repetitive regions resulting in collapsed and fragmented genome assemblies. Long read sequencing technologies, such as those produced by Pacific Biosciences and Oxford Nanopore have much longer read lengths, however, they suffer from high error rates, relatively low throughput and high costs. Optical mapping produces contiguous information spanning long and complex repeat regions, facilitating genome assembly and structural variation detection (Tang *et al.*, 2015).

Optical mapping uses restriction cut sites to physically map genomic regions. High error rates and low throughput initially hindered the uptake of optical mapping. However, with recent improvements in technology and computational algorithms, particularly those implemented by BioNano Genomics, optical mapping has become increasingly applied in genomic analysis. Currently, BioNano optical mapping has been applied in humans (Lam *et al.*, 2012; Mostovoy *et al.*, 2016; Seo *et al.*, 2016; Shi *et al.*, 2016), plants (Hastie *et al.*, 2013; Stankova *et al.*, 2016; Yang *et al.*, 2016), vertebrates (Howe and Wood, 2015), insects (Rosenfeld *et al.*, 2016) and fish (Xiao *et al.*, 2015).

IrysView is the software bundled with the BioNano Irys system and as such has dominated BioNano data analysis. However, some features of this software have limited its use for large-scale projects. IrysView can only run on Windows platforms. For large-scale analysis such as *de novo* assembly and hybrid assembly, an additional remote server is required. The establishment of a remote server from IrysView is not a trivial task. Without installing sun grid engine (SGE), it is not easy to submit a computational job from IrysView. Installing SGE needs a superuser or root permission and it is impossible to install SGE by normal users. If users do not use SGE, IrysView will directly connect to a login node if there are no extra settings, in where heavy workload jobs are prohibited. Furthermore, users cannot run IrysView in the background and close the window if a local job is running. It does not allow multiple jobs at the same time. These limitations restrict the use of BioNano optical mapping for different projects through IrysView.

KSU Lab has released its pipeline and scripts to standardize the length of each captured BioNano single molecule map and assist scaffolding (Shelton *et al.*, 2015). In this pipeline, they applied

different parameter settings and statistical methods to improve scaffolding quality. However, the limited customizability, unique server requirements, specific folder structure requirements and hard coding for different variables limit the use of the KSU pipeline.

To resolve these problems and enhance the flexibility of BioNano data analysis, we developed runBNG, a free software package to perform BioNano data analysis on Linux servers. runBNG was developed based on the functions implemented in IrysView. However, it does not suffer from the limitations of the Windows based IrysView system. runBNG supports commonly used cluster jobs, such as PBS, PBSPro, PBSTorque, Slurm, and SEG jobs. It has a high customizability, which allows users to customize each analysis depending on the aims of their research and the availability of their computational resource. Multiple jobs can be run simultaneously. Jobs can be run in the background.

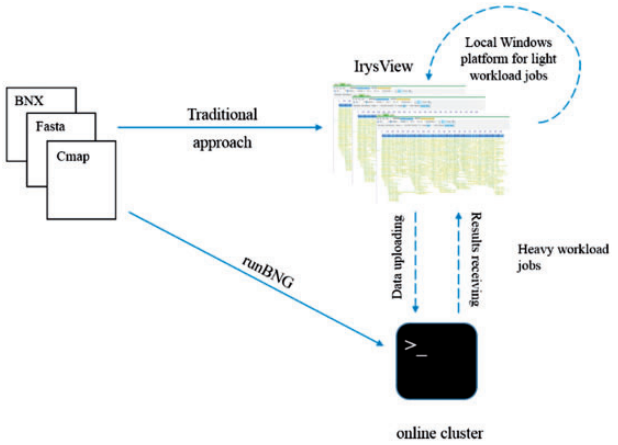

Fig. 1. IrysView based BioNano genomic analysis approaches and the approaches provided by runBNG. IrysView is a Windows based software. It provides two options to help run BioNano genomic analyses: locally or on a remote server. All commands are set from IrysView. For small workload jobs, such as 'Molecule quality report', it can be run on a local Windows platform or on a remote server. For large workload jobs, such as 'Hybrid scaffolding' it is mandatory to run on a remote server. runBNG, on the other hand, skip the use of Windows and performs all analyses directly on a Linux platform

2 Materials and Methods

runBNG is written in bash and implemented in a Linux environment. The minimum Linux system requirement is Ubuntu LTS, or CentOS6.4 or equivalent. Minimum dependencies are python v2.7.5, perl v5.10.x, v5.14.x or v5.16.x, gcc 4.4.7, and glibc 2.15. The BioNano IrysSolve tools including BioNano RefAligner, BioNano Assembler also have to be present. If users want to visualize the analysed results, a local browser tool JBrowse (Skinner *et al.*, 2009) is recommended, but it is also possible to import the results into IrysView (Fig. 1).

3 Results

runBNG provides the same functions that are implemented in BioNano IrysView. A comparison between IrysView and runBNG is given in Table 1. A test using runBNG and IrysView is presented in the Supplementary Material. In Table 1, runBNG demonstrates advantages in data handling, customizability and running feasibility. Compared to KSU scripts, although users can modify KSU scripts to make them work on their own server, it is not a trivial task for normal users. Firstly, users need knowledge of Perl. Secondly, users need to find the key variables in the KSU scripts and modify them. Thirdly, most of the variables in KSU scripts are hard coded, which means it is not easy to modify them. Furthermore, users cannot use their customized folder structure if they do not change the variable settings in KSU scripts, such as the path to BioNano scripts and tools. Users need to find the corresponding script in the KSU scripts directory and run each analysis step. The function integration in KSU scripts is relatively low. By contrast, runBNG is highly customizable, can be easily modified, integrates different functions in one script and can be run on diverse Linux machines.

4 Conclusion

runBNG is useful for Linux-based BioNano genomic analysis since it avoids the need to switch between different platforms. It is easy to customize parameter settings, and the platform requirement to run runBNG is flexible. Users can easily run each IrysView function

Table 1. Comparison between runBNG, BioNano IrysView and KSU scripts

Subject	Item	runBNG	IrysView	KSU scripts
Data handling	Need uploading	Yes, to Linux cluster	Yes, to Linux cluster	Yes, to Linux cluster
	Upload times	Once	Always	Once
Server requirements	Job scheduler	Supports all	SGE, others need extra settings	Support all
	Hardware configuration	No special requirement	No special requirement	576 cores, 256 GB of RAM
Working on a server	Setting complexity	Low	Relatively high	Low
Functions	Digestion *in silico*	Supports	Supports, but cannot be modified	Support
	De novo assembly	Supports	Supports	Support
	Super-scaffolding	Supports, but needs to solve conflicts in settings	Supports, but needs to solve conflicts in settings	Support, but need to select the best assembly
	SV detection	Supports	Supports, but during *de novo* assembly	Support, but during *de novo* assembly
	Maps comparison	Supports	Supports	No
	Multiple jobs at once	Supports	No	Supports
Visualization	Visualization	No	Supports	No
Code remodification	Complexity	Low	N/A	High

implemented in runBNG without the limitation of the windows interface or the requirement for a dedicated custom server.

Acknowledgements

Y.Y. would like to thank the China Scholarship Council (CSC) for supporting his PhD studies at the University of Western Australia. We thank Susan Brown for permission to use her demo data. Support is also acknowledged from the Australian Genome Research Facility (AGRF) and the Queensland Cyber Infrastructure Foundation (QCIF), the Pawsey Supercomputing Centre with funding from the Australian Government and the Government of Western Australia and resources from the National Computational Infrastructure (NCI), which is supported by the Australian Government.

Funding

This work was supported by the Australia Research Council [Projects LP140100537 and LP130100925].

Conflict of Interest: none declared.

References

Goodwin,S. *et al*. (2016) Coming of age: ten years of next-generation sequencing technologies. *Nat. Rev. Genet*., **17**, 333–351.

Hastie,A.R. *et al*. (2013) Rapid genome mapping in nanochannel arrays for highly complete and accurate de novo sequence assembly of the complex Aegilops tauschii genome. *PLoS One*, **8**, e55864.

Howe,K., and Wood,J.M. (2015) Using optical mapping data for the improvement of vertebrate genome assemblies. *Gigascience*, **4**, 10.

Lam,E.T. *et al*. (2012) Genome mapping on nanochannel arrays for structural variation analysis and sequence assembly. *Nat. Biotechnol*., **30**, 771–776.

Mostovoy,Y. *et al*. (2016) A hybrid approach for de novo human genome sequence assembly and phasing. *Nat. Methods*, **13**, 587–590.

Rosenfeld,J.A. *et al*. (2016) Genome assembly and geospatial phylogenomics of the bed bug Cimex lectularius. *Nat. Commun*., **7**, 10164.

Seo,J.S. *et al*. (2016) De novo assembly and phasing of a Korean human genome. *Nature*, **538**, 243–247.

Shelton,J.M. *et al*. (2015) Tools and pipelines for BioNano data: molecule assembly pipeline and FASTA super scaffolding tool. *BMC Genomics*, **16**, 734.

Shi,L. *et al*. (2016) Long-read sequencing and de novo assembly of a Chinese genome. *Nat. Commun*., **7**, 12065.

Skinner,M.E. *et al*. (2009) JBrowse: a next-generation genome browser. *Genome Res*., **19**, 1630–1638.

Stankova,H. *et al*. (2016) BioNano genome mapping of individual chromosomes supports physical mapping and sequence assembly in complex plant genomes. *Plant Biotechnol. J*., **14**, 1523–1531.

Tang,H. *et al*. (2015) Optical mapping in plant comparative genomics. *Gigascience*, **4**, 3.

Xiao,S. *et al*. (2015) Rapid construction of genome map for large yellow croaker (Larimichthys crocea) by the whole-genome mapping in BioNano Genomics Irys system. *BMC Genomics*, **16**, 670.

Yang,J. *et al*. (2016) The genome sequence of allopolyploid Brassica juncea and analysis of differential homoeolog gene expression influencing selection. *Nat. Genet*., **48**, 1225–1232.