Received Date : 05-May-2016
Revised Date : 29-Aug-2016
Accepted Date : 06-Sep-2016
Article type : Special Issue

Mail id: bwringe@gmail.com

parallelnewhybrid: an R package for the parallelization of hybrid detection using NEWHYBRIDS

Brendan F. Wringe[1*], Ryan R. E. Stanley[1], Nicholas W. Jeffery[1], Eric C. Anderson[2], and Ian R. Bradbury[1]

[1] Science Branch, Department of Fisheries and Oceans Canada, 80 East White Hills Road, St. John's NL, A1C 5X1

[2] Fisheries Ecology Division, National Oceanic and Atmospheric Administration Southwest Fisheries Science Center, Santa Cruz, CA, 95060

Running head: parallelization of hybrid detection in R

Keywords: hybrid, introgression, R, newhybrids, population genomics, population structure, software

Abstract

Hybridization among populations and species is a central theme in many areas of biology, and the study of hybridization has direct applicability to testing hypotheses about evolution, speciation, and genetic recombination, as well as having conservation, legal and regulatory implications. Yet, despite being a topic of considerable interest, the identification of hybrid individuals, and quantification of the (un)certainty surrounding the identifications remains difficult. Unlike other programs that exist to identify hybrids based on genotypic information, NEWHYBRIDS is able to assign individuals to specific hybrid classes (e.g. F_1, F_2) because it makes use of patterns of gene inheritance within each locus, rather than just the proportions of gene inheritance within each individual. For each comparison and set of markers, multiple independent runs of each dataset should be used to develop an estimate of the hybrid class assignment accuracy. The necessity of analyzing multiple simulated datasets, constructed from large genome-wide datasets presents significant computational challenges. To address these challenges we present *parallelnewhybrid*, an R package designed to decrease user burden when undertaking multiple NEWHYBRIDS analyses. *parallelnewhybrid* does so by taking advantage of the parallel computational capabilities inherent in modern computers to efficiently and automatically execute separate NEWHYBRIDS runs in parallel. We show that parallelization of analyses using this package affords users several-fold reductions in time over a traditional serial analysis. *parallelnewhybrid* consists of an example dataset, a README and three operating system-specific functions to execute parallel NEWHYBRIDS analyses on each of a computer's *c* cores. *parallelnewhybrid* is freely available on the long-term software hosting site GitHub (www.github.com/bwringe/parallelnewhybrid).

Introduction

Hybridization among closely related species, and genetically distinct populations of the same species, is a topic of broad interest to many fields of biology (Abbott *et al.* 2013; Todesco *et al.* 2016; Warschefsky *et al.* 2014). Natural hybrid zones, areas in which genetically distinct populations come into contact, and interbreed, are widely known and studied, especially in terms of hybridization's impact on speciation and evolution (Barton

& Hewitt 1985; Benson *et al.* 2014; Hilbish *et al.* 2012). Hybridization can slow or reverse speciation by allowing gene flow and recombination, but it can also increase the rate at which speciation occurs through adaptive introgression, and can even lead to the near instantaneous creation of novel species via allopolyploidization [(Abbott *et al.* 2013) but see (Barton 2013)]. The study of hybridization can also have conservation, legal and policy implications as it relates to the genetic structure and integrity of populations (Benson *et al.* 2014; Fitzpatrick *et al.* 2015), or introgression of domesticated (Kidd *et al.* 2009) or transgenic (Warwick *et al.* 2003) alleles into wild populations. However, despite the importance of understanding the dynamics of hybridization, the identification of hybrids themselves can be difficult, and ascertaining to which hybrid class (e.g. F_1, F_2, backcross) an individual belongs is more so.

Intuitively and of salience, the ease and precision with which hybridized individuals can be identified using genetic methods is inversely related to the degree of (genetic) relatedness between the groups which are (suspected of) hybridizing (Vaha & Primmer 2006). However, in many cases the degree of genetic differentiation among groups (e.g. within or among species) may be low, limiting the ability to evaluate hybridization and introgression within and among species. The identification of hybrids and the ability to determine the presence, types, and numbers of individuals of different hybrid classes can provide crucial information on the presence, magnitude, and time scale over which introgression is occurring. While several statistical approaches [reviewed by (Anderson 2009; Payseur & Rieseberg 2016)] and software programmes exist [e.g. STRUCTURE, (Hubisz *et al.* 2009); NEWHYBRIDS (Anderson & Thompson 2002); BAYESASS (Wilson & Rannala 2003); GENODIVE (Meirmans & Van Tienderen 2004), which uses the maximum likelihood method of (Buerkle 2005)] to identify hybrids, most do not assign to hybrid class, thus losing potentially important information. NEWHYBRIDS (Anderson & Thompson 2002) is unique in this respect, in the discreteness of individual assignment to specific hybrid classes it provides. This is done through evaluation of the Bayesian posterior probability of membership in each of six genotype frequency classes [i.e. pure population 1, pure population 2, F_1, F_2, back-cross to population 1, back-cross to population 2 (Anderson & Thompson 2002)], for each individual, computed using Markov chain Monte Carlo (MCMC; Anderson & Thompson

2002). In addition, NEWHYBRIDS does not require that the allele frequencies of the two populations be known *a priori*, and, as such, pure samples of the two populations need not be available to identify hybrids. This is advantageous for identifying escapees from domesticated populations without pre-impact, baseline samples from the wild populations, or in situations where the domesticated animals are derived from a local wild population. However, (as with most methods) the accuracy of NEWHYBRIDS to correctly differentiate hybrid class is highly dependent upon the number of informative markers provided, and the genetic distinctness of the two populations in question (Anderson & Thompson 2002; Vaha & Primmer 2006). Consequently, while the large genome-wide data sets produced by next-generation sequencing and genotyping methods may offer significant opportunities for improved hybrid class identification, at the same time their size and complexity present challenges for existing software. Ultimately, for each comparison and set of markers, the accuracy should be tested using the results of multiple simulated datasets, and multiple independent runs of each dataset to ensure convergence (Anderson 2003) further increasing the computational demands.

Currently, the speed at which a complete NEWHYBRIDS analysis can be completed is limited by the fact that each analysis must be initiated separately and in sequence by the user (Anderson 2003; Anderson & Thompson 2002). Since NEWHYBRIDS was published (Anderson & Thompson 2002), shared-memory multiple-computer processing unit (CPU/core) chipsets have supplanted single CPU chips, and are present in most consumer computers. If properly leveraged, this multi-core architecture can allow for more time-efficient computing by distributing tasks among cores, and allowing for parallel processing (e.g. Besnier & Glover 2013). When compared against running analyses in series, the benefits of such a parallel approach would be proportional to the size of the dataset, and thus be a boon for the analysis/use of large, next generation sequencing datasets which themselves offer significant advances in hybrid identification and classification.

It has been previously shown that scripting not only allows tasks to be reliably distributed across the available cores, but can also immediately assign a new job to a free core as soon as a core has finished a process (Besnier & Glover 2013). Although such scripting is invariably more efficient, specialized knowledge of the computer file

structure and requisite programming language is required. Here we describe an R (R Development Core Team 2015) package that takes advantage of the parallel computational capabilities inherent in modern computers to efficiently and automatically analyze lists of NEWHYBRIDS runs in parallel. We emphasize that we are not multi-threading the NEWHYBRIDS program to take advantage of multiple processors during a single run; any single run will still take just as long, but we are automating the procedure for performing multiple runs of the software in parallel. The R computing language already features many packages dedicated to the analysis and presentation of population genetics data (e.g. Goudet 2005; Jombart & Ahmed 2011; Paradis 2010). Thus, although the R language is likely to be familiar to many current and potential users of NEWHYBRIDS, our package is designed such that it should be readily usable by R novices.

Materials and Methods

Description of the package

Though NEWHYBRIDS does not natively support multi-threading, it is possible to run independent NEWHYBRIDS analyses in parallel across all available cores. This process can, however, be tedious because NEWHYBRIDS does not have an option to change the name or location of the output files. Accordingly, to run NEWHYBRIDS in parallel, the user must perform each run in a separate directory so that the output of one instance of NEWHYBRIDS is not overwritten by the output of another instance. Doing this manually requires that the user open separate command-line terminals (e.g. Microsoft command-line, OS X Terminal) and execute NEWHYBRIDS with a specific set of parameters for each analysis folder. The status of each NEWHYBRIDS run must be individually monitored by the user, and when each analysis is completed, all the output files generated by NEWHYBRIDS must be renamed, and combined for post-processing. Such manual monitoring and (re)implementation invariably creates CPU downtime and thus inefficiency because it is difficult for the user to monitor the NEWHYBRIDS progress due to the extended duration of the analysis (many hours and potentially days).

Furthermore, manually copying, moving and renaming files introduces the potential for human error.

Our package, *parallelnewhybrid*, is designed to address these issues: it implements the parallelization of multiple NEWHYBRIDS analyses, and also automatically compiles and renames the outputs of NEWHYBRIDS to reflect the file names of the datasets that were provided to it. Differences in computer operating system architecture mean that the manner in which tasks are distributed in Windows differs from OS X and LINUX. Consequently, in addition to an example dataset, the package *parallelnewhybrid* is comprised of three operating system-specific functions to implement the parallelization of NEWHYBRIDS: *parallelnh_WIN*, *parallelnh_OSX*, and *parallelnh_LINUX*. A further consequence of the manner in which parallelization is effected by R in Windows is that most parallelization packages, *parallelnewhybrid* included, do not function correctly in graphic user interface (GUI) or embedded environment R sessions. While we have been successful in utilizing *parallelnh_xx* in the R GUI programmes RStudio (RStudio Team 2015) and R Console (R Development Core Team 2015) in both OS X (OS X Version 10.11.3, MacBook Pro, 2.3 GHz Intel Core i7, 16 GB RAM) and LINUX (Ubuntu Version 14.04, Dell Precision Tower 7190, 2X 2.3 GHz Intel Xeon, 32 GB RAM), we have found running through the terminal to offer better stability. As such, we highly recommend that *parallelnh_xx* be run in the terminal by default. It may also be necessary to run R as root or administrator when invoking *parallelnh_xx* because some anti-virus programmes and tools may prevent the manipulation of the NEWHYBRIDS executable file.

Example dataset

SimPops_NH.txt, is a simulated dataset with genotypes at 240 loci for 200 individuals in each of the six genotype frequency classes (i.e. pure1, pure2, F1, F2, BC1 and BC2; Anderson & Thompson 2002), and is intended to be analyzed in parallel using *parallelnh_xx*. The instructions for the user on how to copy the file from the R repository to another folder on the user's hard drive and to prepare the file for parallel analysis, are detailed in the README (https://github.com/bwringe/parallelnewhybrid).

Quantification of improvement in performance

We compared the time required to analyze three independently simulated datasets, each replicated three times using *parallelnh_xx* versus a single-threaded (serial) implementation. In all cases, NEWHYBRIDS was run with an with an initial burn-in of 500 replicates, followed by 1000 sweeps (MCMC terminology of Anderson & Thompson 2002). While these numbers are lower than would be typically used in a real analysis, since time to completion scales linearly with the number of iterations (i.e. sum of burnin and sweeps), the observed fold change improvements should be invariant to number of iterations. To implement the serial analysis, we restricted the multi-threading ability of *parallelnh_xx* such that it was limited to iteratively populating a single core. We acknowledge that this is likely an overestimate of the single core speed, as the script will invariably be faster than a human operator in initiating a new run as each finishes. We chose to analyze 8 files on CPU architectures with 8 virtual (4 physical) cores (Intel Core i7; Samsung and MacBook Pro), while in the case of the Intel Xeon which has 24 virtual (12 physical) cores, scripting limited execution to a maximum of 8 cores. We also examined how the time to complete the 8 analyses scaled with the number of (virtual) cores available (1 to 8) to illustrate how differences in operating system and chip architecture influence the operation of our parallelization function. This also allowed us to observe operation under the least computationally efficient scenario: that is where the number of files to be analyzed is not a multiple of the number of (virtual) cores. Thus in cases where $n \equiv c \pmod{r} \mid r \neq 0$, where n is the number of files to be analyzed and c is the number of cores available, the function must complete $n \cdot c^{-1} + 1$ runs in parallel with jobs allocated to all cores, plus a single run in which $c - r$ cores are idle.

Results and Discussion

Not surprisingly, even when the initiation of new runs was automated, the time to complete the analysis of the nine files (three simulated datasets, each run in triplicate) was much slower without parallelization (Table 1). However, the relationship between improvement in computational speed, and the number of cores made available was not linear (Table 2). This demonstrates that the ability to automatically distribute a list of

analyses across multiple CPUs offers quantitative improvements over the native command line implementation of NEWHYBRIDS. While always quicker than running analyses sequentially, we found that the computational time did not decrease linearly with the number of cores implemented in the analyses. This relationship was seen in each operating system (i.e. Windows, OS X, LINUX), and on the different CPU architectures (Intel Core i7, Intel Xeon), and has been reported elsewhere (Besnier & Glover 2013). Like Besnierand Glover (2013), we suggest this non-linearity was not caused by *parallelnh_xx*, but instead is a function of how the operating system and processor deals with the distribution and execution of computationally intensive processes (i.e. NEWHYBRIDS) along with (operating system specific) underlying system processes. Further, monitoring of system resources during each trial indicated that performance was not limited by the availability of random access memory (RAM), as additional RAM was always available during each simulation.

We acknowledge that functionality analogous to that offered by *parallelnewhybrid* could be achieved using scripting languages (e.g. bash scrip for Unix-like systems, shell script for Windows). However, we chose to use the R programming language because we feel it offers several benefits that can help make the functionality offered by *parallelnewhybrid* accessible to, and utilized by more people. These include an existing population genetic user base which may afford most potential users with greater familiarity and comfort with R than with scripting languages, as well as the infrastructure present within the R community for the archival and distribution of packages.

In summary, we have developed an R package that provides a substantial decrease in the time required to validate and conduct hybrid detection by enabling the parallelization of analyses using NEWHYBRIDS. Furthermore, because the time to complete analyses scales with the size of the dataset provided (number of loci and individuals) and because running in parallel was always faster than in series, this package will enable the exploration of hybrid class assignment power and the utilization of larger datasets than previously feasible with NEWHYBRIDS. This should allow researchers conducting hybrid detection to generate more accurate posterior-probability thresholds for identifying individual hybrid categories by examining the accuracy with which

NEWHYBRIDS correctly identifies the hybrid class of known individuals in replicated analyses of multiple simulated datasets. Furthermore, constructing and testing multiple simulated datasets is especially important when attempting to eliminate high-grading bias through the use of simulation and training datasets (Anderson 2010), when sample sizes are small and thus gene frequencies more prone to sampling-induced alteration, and when the genetic differentiation between populations is low (Vaha & Primmer 2006). Conversely, *parallelnewhybrid* will also allow for larger datasets, both numbers of individuals and numbers of loci per individual, to be tested than are currently feasible, which may increase the ability for identification of hybridization at fine-scale levels of genetic differentiation.

Acknowledgements

The authors wish to thank Marion Sinclair-Waters and Mallory Van Wynegaarden for their help bug-checking the code. We also thank Thierry Gosselin for encouraging us to publish this package. This work was supported by a Natural Sciences and Engineering Research Council Strategic project Grant and Fisheries and Oceans Canada funding (International Governance Strategy; Program for Aquaculture Regulatory Research; Genomics research and Development Initiative) to I.R.B.

Author contributions

B.F.W. wrote the manuscript and the package code, and developed the supporting documentation and example data files hosted on GitHub. R.R.E.S, N.F.W., E.C.A., and I.R.B. all contributed to the initial concept, development of the code, and associated documentation, as well as assisting in the writing of the manuscript.

Data Accessibility

The package, user manual, README, and example data set are all available online from https://github.com/bwringe/parallelnewhybrid

References

Abbott R, Albach D, Ansell S*, et al.* (2013) Hybridization and speciation. *Journal of Evolutionary Biology* **26**, 229-246.

Anderson EC (2003) User's guide to the program NewHybrids Version 1.1 beta. Department of Integrative Biology, University of California, Berkeley, Berkeley, California.

Anderson EC (2009) Statistical methods for identifying hybrids and groups. In: *Population genetics for animal conservation* (eds. Bertorelle G, Bruford MW, Hauff HC, Rizzoli A, Vernesi C), pp. 25-41. Cambridge University Press, New York.

Anderson EC (2010) Assessing the power of informative subsets of loci for population assignment: standard methods are upwardly biased. *Molecular Ecology Resources* **10**, 701-710.

Anderson EC, Thompson EA (2002) A model-based method for identifying species hybrids using multilocus genetic data. *Genetics* **160**, 1217-1229.

Barton NH (2013) Does hybridization influence speciation? *Journal of Evolutionary Biology* **26**, 267-269.

Barton NH, Hewitt GM (1985) Analysis of hybrid zones. *Annual Review of Ecology and Systematics* **16**, 113-148.

Benson JF, Patterson BR, Mahoney PJ (2014) A protected area influences genotype-specific survival and the structure of a *Canis* hybrid zone. *Ecology* **95**, 254-264.

Besnier F, Glover KA (2013) ParallelStructure: A R package to distribute parallel runs of the population genetics program STRUCTURE on multi-core computers. *PLoS One* **8**.

Buerkle CA (2005) Maximum-likelihood estimation of a hybrid index based on molecular markers. *Molecular Ecology Notes* **5**, 684-687.

Fitzpatrick BM, Ryan ME, Johnson JR, Corush J, Carter ET (2015) Hybridization and the species problem in conservation. *Current Zoology* **61**, 206-216.

Goudet J (2005) HIERFSTAT, a package for R to compute and test hierarchical F-statistics. *Molecular Ecology Notes* **5**, 184-186.

Hilbish TJ, Lima FP, Brannock PM*, et al.* (2012) Change and stasis in marine hybrid zones in response to climate warming. *Journal of Biogeography* **39**, 676-687.

Hubisz MJ, Falush D, Stephens M, Pritchard JK (2009) Inferring weak population structure with the assistance of sample group information. *Molecular Ecology Resources* **9**, 1322-1332.

Jombart T, Ahmed I (2011) adegenet 1.3-1: new tools for the analysis of genome-wide SNP data. *Bioinformatics*.

Kidd AG, Bowman J, Lesbarreres D, Schulte-Hostedde AI (2009) Hybridization between escaped domestic and wild American mink (*Neovison vison*). *Molecular Ecology* **18**, 1175-1186.

Meirmans PG, Van Tienderen PH (2004) GENOTYPE and GENODIVE: two programs for the analysis of genetic diversity of asexual organisms. *Molecular Ecology Notes* **4**, 792-794.

Paradis E (2010) pegas: an R package for population genetics with an integrated–modular approach. *Bioinformatics* **26**, 419-420.

Payseur BA, Rieseberg LH (2016) A genomic perspective on hybridization and speciation. *Molecular Ecology*.

R Development Core Team (2015) *R: A language and environment for statistical computing* R Foundation for Statistical Computing, Vienna, Austria.

RStudio Team (2015) RStudio: Integrated Development for R. RStudio, Inc.

Todesco M, Pascual MA, Owens GL*, et al.* (2016) Hybridization and extinction. *Evolutionary Applications*.

Vaha JP, Primmer CR (2006) Efficiency of model-based Bayesian methods for detecting hybrid individuals under different hybridization scenarios and with different numbers of loci. *Molecular Ecology* **15**, 63-72.

Warschefsky E, Penmetsa RV, Cook DR, von Wettberg EJB (2014) Back to the wilds: tapping evolutionary adaptations for resilient crops through systematic hybridization with crop wild relatives. *American Journal of Botany* **101**, 1791-1800.

303 Warwick SI, Simard MJ, Legere A*, et al.* (2003) Hybridization between transgenic
304 *Brassica napus* L. and its wild relatives: *Brassica rapa* L., *Raphanus*
305 *raphanistrum* L., *Sinapis arvensis* L., and *Erucastrum gallicum* (Willd.) OE
306 Schulz. *Theoretical and Applied Genetics* **107**, 528-539.
307 Wilson GA, Rannala B (2003) Bayesian inference of recent migration rates using
308 multilocus genotypes. *Genetics* **163**, 1177-1191.
309

Table 1. Comparison of the computational times required to complete the analysis of the nine files (three simulated datasets, each run in triplicate) using NEWHYBRIDS in series compared to in parallel using *parallelnewhybrid* using different operating systems and CPU architectures.

Computer	Time to complete in series (min)	Time to complete in parallel (min)	Fold improvement
MacBook Pro[1]	41.60	10.13	4.10
Samsung Windows 10[2]	72.50	22.47	3.21
Dell Precision Tower Ubuntu[3]	73.36	9.93	7.62

[1]MacBook Pro, OS X 10.11, 2.3 GHz Intel Core i7 with 16 GB RAM

[2]Samsung, Windows 10, 2.3 GHz Intel Core i7 with 12 GB RAM

[3]Dell Precision Tower 7190, Ubuntu Version 14.04, 2X 2.3 GHz Intel Xeon with 32 GB RAM

Table 2 Computational time required to analyze 8 simulated datasets each with 1200 individuals genotyped at 240 loci. The analysis was repeated using different operating system and CPU architectures. CPU cores refers to the number of cores that the analysis could access simultaneously (i.e. number of parallel executions of NEWHYBRIDS). Fold improvement is calculated relative to the time taken to conduct the analysis using a single core, which itself is analogous to running the analysis in series.

	MacBook Pro[1]		Samsung Windows 10[2]		Dell Precision Tower Ubuntu[3]	
CPU Cores	Elapsed Time (min)	Fold Improvement	Elapsed Time (min)	Fold Improvement	Elapsed Time (min)	Fold Improvement
1	41.61	NA	72.20	NA	73.31	NA
2	20.56	2.02	36.84	1.96	38.12	2.00
3	15.39	2.72	32.32	2.23	30.91	2.47
4	11.90	3.50	30.65	2.35	19.03	4.02
5	12.49	3.33	32.22	2.24	19.30	3.95
6	12.87	3.23	28.56	2.52	18.82	4.05
7	13.75	3.03	29.72	2.43	18.43	4.14
8	10.13	4.10	22.47	3.21	9.95	7.67

[1]MacBook Pro, OS X 10.11, 2.3 GHz Intel Core i7 with 16 GB RAM

[2]Samsung, Windows 10, 2.3 GHz Intel Core i7 with 12 GB RAM

[3]Dell Precision Tower 7190, Ubuntu Version 14.04, 2X 2.3 GHz Intel Xeon with 32 GB RAM