genomalicious: serving up a smorgasbord of R functions for population genomic analyses

Joshua A. Thia[★❖] & Cynthia Riginos[★]

★ School of Biological Sciences, The University of Queensland, Brisbane, Australia.

❖ Corresponding author: josh.thia@live.com

ABSTRACT

Turning SNP data into biologically meaningful results requires considerable computational acrobatics, including importing, exporting, and manipulating data among different analytical packages and programming environments, and finding ways to visualise results for data exploration and presentation. We introduce *genomalicious*, an R package designed to provide a selection of functions for population genomicists to simply, intuitively, and flexibly, guide SNP data through their analytical pipelines, within and outside R. Moreover, researchers using pooled allele frequencies, or individually sequenced genotypes, are sure to find functions that accommodate their tastes in *genomalicious*. The source code for this package is freely available on GitHub.

INTRODUCTION

The abundance of tools available to modern population geneticists has provided means to answer questions about biodiversity, demography, and evolution in a multitude of different ways. Moreover, genomics has enriched our investigations by providing data that is abundant (in terms of number of loci) and resolute (in terms of the scale of questions we can address). The declining cost of individual sequencing, the cost efficiency of pooling, and relative ease of assembling reduced representation datasets is building towards an era where genomic data will be the norm for most population genetic studies. Specifically, in this note, we refer to "genomic data" as SNPs (single nucleotide polymorphisms) derived from reduced representation methods (e.g. RAD-seq, GBS, WGS, or other related technologies).

Choosing to apply genomic SNP information to population genetic questions comes with the challenge of understanding basic computer programming. First and foremost, a grasp of the Unix environment is essential for turning sequence reads into genotype calls. Getting beyond this first hurdle can be exceedingly difficult for first-time genomic data handling. Furthermore, many academic supervisors in population genetic fields do not possess the programming skills required to offer strong guidance to their post-graduate students. Hence, traversing the proverbial "Valley of Assembling, Calling, Filtering, and Agonising Over Parameter Choices" represents a steep learning

curve that can make the choice to use genomics daunting. However, the journey does not end there, and the next major task involves decisions on how best to analyse the potential thousands of genome-wide SNPs obtained from assembly–variant calling pipelines.

The choice of analyses and associated programs depends on an investigator's hypotheses and personal preferences. Most researchers are likely to end up in R, given its widespread use among scientists for statistical analysis and data visualisation. However, many population genetic R packages require very distinct data structures or are designed in a way that can constrain flexibility in how a researcher chooses to handle their data. Furthermore, some desired tools might not be available in R, requiring computational acrobatics to import, export and manipulate data to move among the various programs in a researcher's analysis pipeline.

We have developed *genomalicious* (think "delicious", but "genoma", *i.e.* "**genom**ic **a**nalyses") as a tool to help bridge computational challenges associated with moving genomic SNP data through an analysis pipeline that occurs both outside and within R. The primary tenet of this package is that the functions serve the user: there are no strict object classes involved, which allows a user to easily access and direct data where needed. The only major user restriction is that *genomalicious*' functions heavily utilise the `data.table` class objects, as per the *data.table* package (Dowle and Srinivasan 2018). Objects of this class have a some very useful manipulation features that make them ideal for dealing with large datasets: see the vignettes on CRAN for more details and explanations. Moreover, `data.table` objects are also innately dual `data.frame` objects, and can easily be converted into pure `data.frame` or `matrix` objects using the respective functions, `as.data.frame()` and `as.matrix()`.

In the sections that follow, we briefly highlight and describe five main degustation menus that *genomalicious* provides and how these can be incorporated into analysis pipelines of population genomic data, based on pooled or individually sequenced samples.

IMPORT AND EXPORT

The typical datatype a population genomicist will be working with are SNPs recorded as variant calls in a VCF file format. Though packages already exist to import VCF files into R, the returned objects do not necessarily have immediately intuitive structure or lack clear, straightforward ways to manipulate them. Furthermore, some analytical functions take VCF files as their direct input, but this reduces the extent to which manipulation and piping of SNP data can occur directly in R.

In *genomalicious*, we provide a method to import VCF files into R as a long-format `data.table` object. This may not be the most efficient way to handle extremely large datasets, but even for thousands of loci, over hundreds of individuals, this is a useful way to store population genomic data. The function `vcf2DT()` takes VCF files, which are in in wide-format, where each sample has its own column and loci are in rows, and converts them into long-format, samples and loci are both in columns (see Figure 1).

There are several reasons why we believe this a very useful starting format for population genomic data. (1) Tabular data are easy to visualise and mentally interpret. (2) Tabular data in long-format is used in many R functions, allowing smoother integration of SNP data into R pipelines. (3)

The `data.table` class stores large volumes of information efficiently and has excellent manipulation features. (4) Therefore, users are returned an object that they are free to manipulate to their liking.

In addition to importing VCF files, *genomalicious* allows individuals to import and export SNP data in formats used by programs outside R. Current programs supported by *genomalicious* are detailed in Table 1. This list reflects our own personal research endeavours, but it is expected to grow over time.

MANIPULATE

Several tools are provided in *genomalicious* to manipulate data. There are functions available to filter SNP data based on read depth, minor allele frequency, or by independent contigs: `filter_depth()`, `filter_maf()`, `filter_unlink()`, respectively. Another functionality is the conversion of genotype or allele frequency data between long-format data tables and wide-format matrices (or vice versa): `DT2Mat_genos()` and `DT2Mat_freqs()`. Users can also convert genotypes into allele frequencies using `genos2freqs()`, or convert between genotype scoring methods using `genoscore_converter()`. Additionally, there are functions to manipulate data into more complex data structures used for programs within R (Table 1).

ANALYSE

Though the purpose of *genomalicious* is not strictly for conducting statistical tests, we provide some functions that can be used to produce analytical results and summary statistics. For example, `fstWC()` can be used to calculate F_{ST} (Weir and Cockerham 1984), or for conducting a principal component analysis (PCA) directly on a long-format genotype data table using `pca_DTgenos()`.

Another function that might be of interest is `locus_overlap()`, which can be used to estimate the null probability of observing shared loci between groups, given a specific sample size and the number of loci analysed. This approach might be of interest when a researcher has a set of outlier loci for 2+ groups and wants to determine what the random sampling probability of observing *X* number of outliers shared among these groups would be.

ILLUSTRATE

We provide some graphical functions that users might find helpful in exploring and visualising their data. Missing data can be visualised using a histogram (Figure 2) of heatmap approach (Figure 3). The function `missHist()` produces a histogram of missing data with respect to samples or loci. A heatmap indicating the presence and absence of missing data can be produced with the function `missHeatmap()`.

Users implementing PCAs can use the `pca_plot()` function to graph their results in different ways. Scatter plots of the ordination, scree plots of eigenvalues, and cumulative explained variance plots, can all be created using this function. The input can be a `prcomp` object, or a

`data.table`/`data.frame`/`matrix` for scatter plots, or a `numeric` object of eigenvalues for scree and cumulative explained variance plots. This is to provide greater flexibility. Users can choose to follow a *genomalicious* pipeline: a genotype data table is fed to `pca_DTgenos`, which utilises the `prcomp()` function to perform the PCA, and can then pass the resulting `prcomp` object directly `pca_plot()`. Alternatively, users can perform a PCA in their own fashion, then feed either the ordination or eigenvalues to `pca_plot()`.

SUMMARY

We concocted *genomalicious* to deliver the following objectives:

1. Easy import of VCF files into R as (long-format) data tables, and import/export tools for various population genetics programs that operate external or within R.

2. Easy-to-use functions for data manipulations into commonly used population genetic data structures or more complex structures required by other functions within R.

3. Functions for basic analyses of population genomic data in R.

4. Functions for visualising population genomic data and results in R.

5. Functions that can be deployed on pooled or individually sequenced data.

This package is an evolving entity. We look forward to feedback and any suggestions on how *genomalicious* can be improved and made to better suit the appetites of population genomicists.

ACCESSIBILITY

The most up-to-date version of *genomalicious* is currently being served on J.A. Thia's github: https://github.com/j-a-thia/genomalicious.

PACKAGE DEPENDENCIES

data.table	Dowle and Srinivasan (2018)
ggplot2	Wickham (2011a)
gridExtra	Auguie (2017)
HierDpart	Qin (2019)
MASS	Venables and Ripley (2002)
poolfstat	Hivert et al. (2018)
plyr	Wickham (2011b)
tidyr	Wickham and Henry (2018)

REFERENCES

Auguie, B. 2017. gridExtra: Miscellaneous Functions for "Grid" Graphics.

Dowle, M., and A. Srinivasan. 2018. data.table: Extension of "data.frame."

Foll, M., and O. Gaggiotti. 2008. A genome-scan method to identify selected loci appropriate for both dominant and codominant markers: A Bayesian perspective. Genetics 180:977–993.

Gaggiotti, O. E., A. Chao, P. Peres-Neto, C. H. Chiu, C. Edwards, M. J. Fortin, L. Jost, C. M. Richards, and K. A. Selkoe. 2018. Diversity from genes to ecosystems: A unifying framework to study variation across biological metrics and scales. Evol. Appl. 11:1176–1193.

Gautier, M., J. Foucaud, K. Gharbi, T. Cézard, M. Galan, A. Loiseau, M. Thomson, P. Pudlo, C. Kerdelhué, and A. Estoup. 2013. Estimation of population allele frequencies from next-generation sequencing data: Pool- versus individual-based genotyping. Mol. Ecol. 22:3766–3779.

Gutenkunst, R. N., R. D. Hernandez, S. H. Williamson, and C. D. Bustamante. 2009. Inferring the joint demographic history of multiple populations from multidimensional SNP frequency data. PLoS Genet. 5:e1000695.

Hivert, V., R. Leblois, E. J. Petit, M. Gautier, and R. Vitalis. 2018. Measuring genetic differentiation from pool-seq data. Genetics 210.

Jombart, T. 2008. adegenet: a R package for the multivariate analysis of genetic markers. Bioinformatics 24:1403–1405.

Qin, X. 2019. HierDpart: Partitioning Hierarchical Diversity and Differentiation Across Metrics and Scales, from Genes to Ecosystems.

Venables, W. N., and B. D. Ripley. 2002. Statistics Complements to Modern Applied Statistics with S. 4th ed. Springer, New York, USA.

Weir, B. S., and C. C. Cockerham. 1984. Estimating F- statistics for the analysis of population structure. Evolution (N. Y). 38:1358–1370.

Wickham, H. 2011a. ggplot2: Elegant Graphics for Data Analysis.

Wickham, H. 2011b. The split-apply-combine strategy for data analysis. J. Stat. Softw. 40:1–29.

Citeseer.

Wickham, H., and L. Henry. 2018. tidyr: Easily Tidy Data with "spread()" and "gather()" Functions.

FIUGRES & TABLES

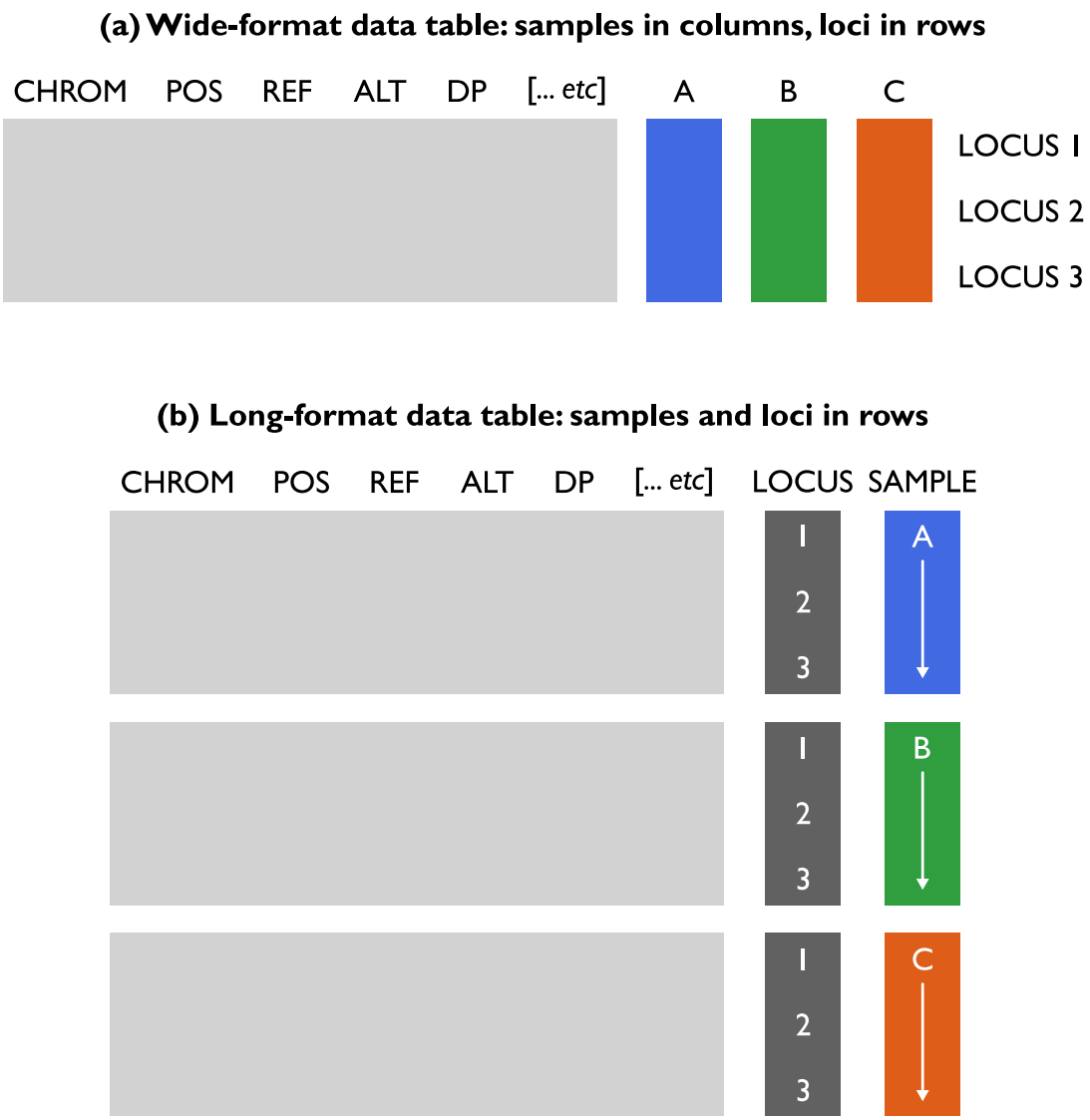
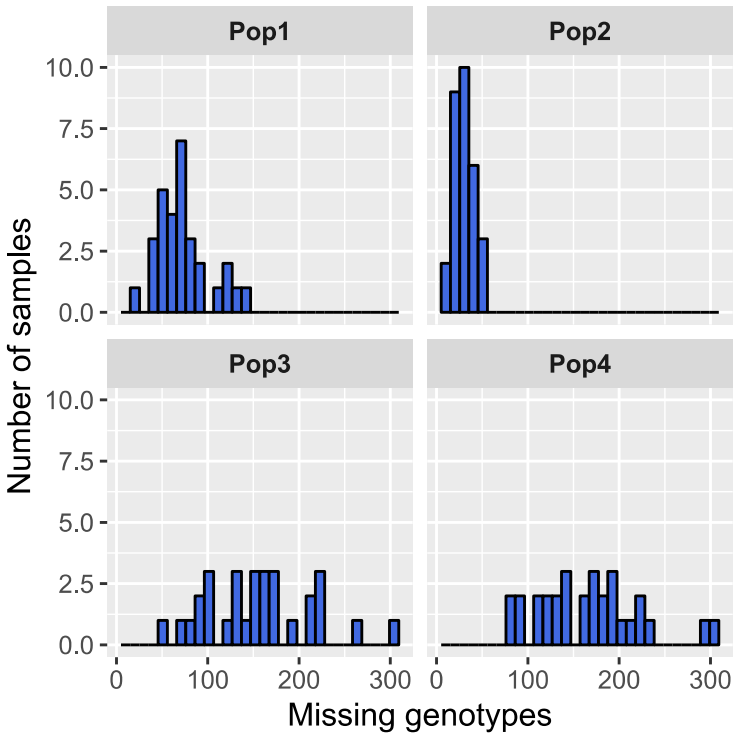


Figure 1. An illustration of wide- and long-format data structures in genetic data. Coloured rectangles represent different data: Metadata = light grey; Samples A, B, and C = blue, green, and orange (respectively); Loci = dark grey. **(a)** Wide-format data table, analogous to that in a typical VCF file. Each sample is stored in an individual column and loci are in rows. **(b)** Long-format data table, with a single column for the locus and sample identifiers. Rows thus contain every unique combination of locus × sample.

(a) Missing data, by samples, with "ggplot" flavour plots



(b) Missing data, by loci, with "classic" R flavour plots

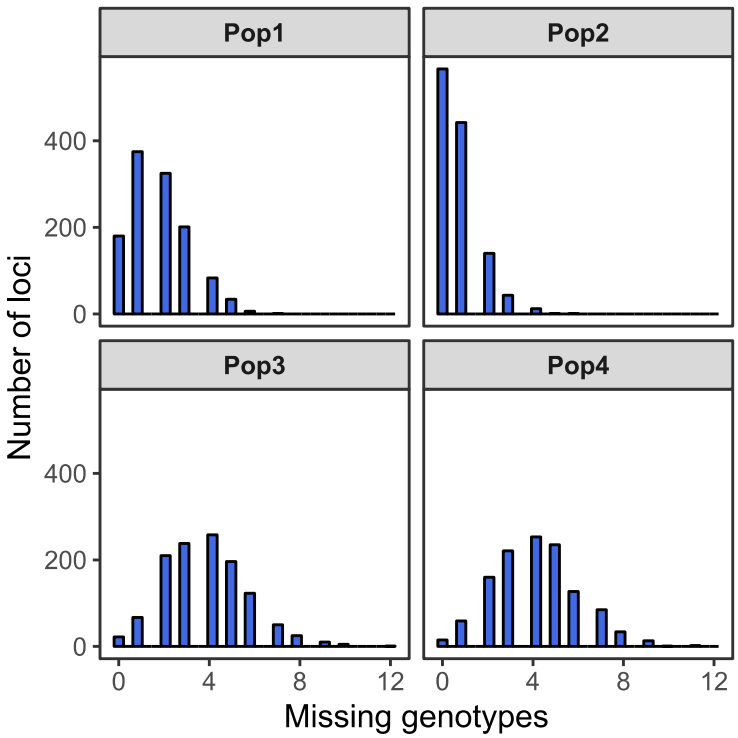


Figure 2. Histogram visualisation of missing data using *genomalicious*. The number of missing genotypes (x-axis) and their relative frequency (y-axis) within each population. Missing data can be plotted with respect to **(a)** samples, or **(b)** loci. The plot styles can be specified to exhibit **(a)** "ggplot" (Wickham 2011a), or **(b)** "classic" R appearances.

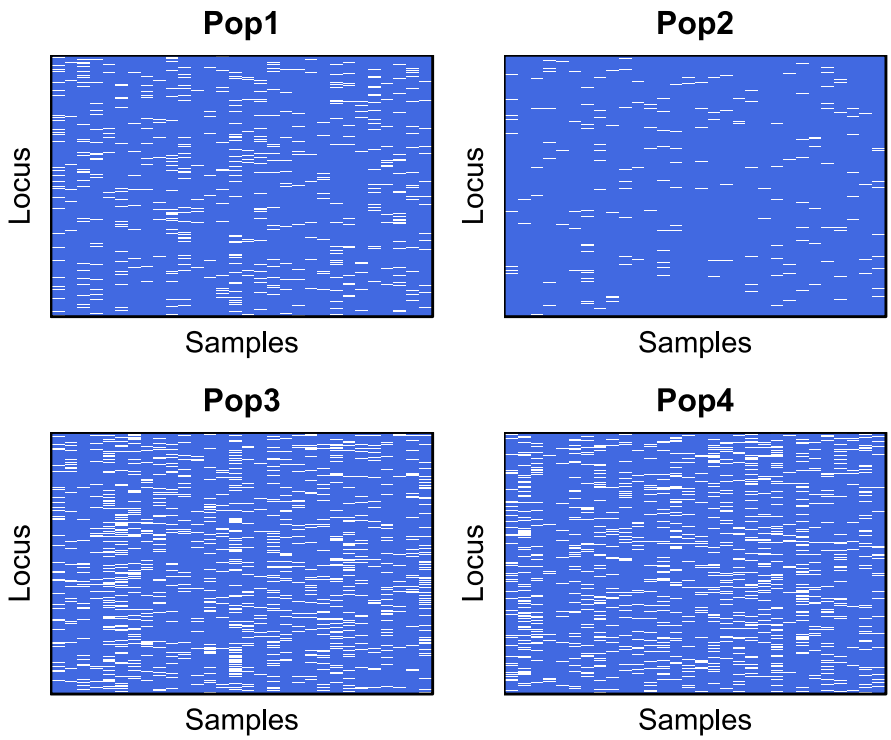


Figure 3. Heatmap visualisation of missing data using *genomalicious*. Samples (x-axis) and loci (y-axis) are plotted against each other, subset by population. Colours represent the presence (blue) and absence (white) of data for each sample × locus combination.

Table 1. Programs and data structures supported by *genomalicious* for importing into R, exporting out of R, or manipulating SNP data tables to fit specifications by other R packages. (Continued on next page)

Setting	Program	Reference	Import	Export	Manipulate	Details
Outside R	*bayescan*	Foll and Gaggiotti (2008)		✓		Input files from pooled allele frequencies: `bayescan_inputs_pools()`
	poolne_estim	Gautier et al. (2013)	✓	✓		Input and output files: `poolne_estim_input()` `poolne_estim_out()`
	δaδi	Gutenkunst et al. (2009)		✓		Input files from pooled allele frequencies: `dadi_inputs_pool()`
Within R	*adegenet*	Jombart (2008)			✓	Convert a long-format genotype data table into a `genind` or `genlight` object : `adegenet_DT2genind()` `adegenet_DT2genlight()`
	HierDpart	Gaggiotti et al. (2018) Qin (2019)			✓	Run `IDIP()` on a long-format genotype data table, across all loci: `idip_DTfreqs()` `idip_DTgenos()`

Table 1 (continued). Programs and data structures supported by *genomalicious* for importing into R, exporting out of R, or manipulating SNP data tables to fit specifications by other R packages.

Setting	Program	Reference	Import	Export	Manipulate	Details

Within R	*poolfstat*	Hivert et al. (2018)			✓	Run `computeFST()` on a long-format data table of read counts, and bootstrap resample over loci to get confidence intervals:
						`poolfstat_boot()`
						`poolfstat_DT()`