

1 Probabilistic graph models for landscape genetics

2 Brook G. Milligan¹

3 ¹Department of Biology, New Mexico State University, Las Cruces, New Mexico 88003 USA

4 Corresponding author:

5 Brook G. Milligan¹

6 Email address: brook@nmsu.edu

7 ABSTRACT

Landscape genetics combines population genetics, landscape ecology, and spatial analysis to identify landscape and genetic factors that influence genetic and genomic variation. Progress in the field depends on a strong conceptual foundation and the means of identifying mechanistic connections between environmental factors, landscape features, and genetic or genomic variation. Many existing approaches and much of the software commonly in use was developed for population genetics or statistics and is not entirely appropriate for landscape genetics. Probabilistic graph models provide a statistically rigorous and flexible means of constructing models directly applicable to landscape genetics. Probabilistic graph models also allow construction of mechanistic models, which are crucial elements in testing hypotheses. Sophisticated software exists for the analysis of graph models; however, much of it does not handle the types of data used for landscape genetics, model structures involving autoregressive spatial interaction between variables, or the scale of landscape genetics problems. Thus, an important priority for the field is to develop suitably flexible software tools for graph models that overcome these problems and allow landscape geneticists to explore meaningfully mechanistic and flexible models. We are developing such a library and applying it to examples in landscape genetics.

Keywords: landscape genetics, population genetics, graph models, Bayesian inference, open source software, software development

8 Landscape genetics combines population genetics, landscape ecology, and spatial analysis to
9 identify the mechanisms by which landscape and environmental factors influence genetic and ge-
10 nomic variation. From the outset, the field has focused on the twin ecological and evolutionary
11 processes of gene flow and adaptation (Holderegger et al., 2006; Manel et al., 2003, 2010). Involving
12 as it does quantification of both genetics and landscapes, landscape genetics is inherently interdis-
13 ciplinary (Balkenhol et al., 2009; Holderegger and Wagner, 2008). While the emphasis is often on
14 the genetics, explicit consideration of the importance of GIS and allied geospatial disciplines is
15 crucial as they can contribute to landscape genetics in many ways (Cushman et al., 2016; Storfer
16 et al., 2007). For example, experimental design in landscape genetics must be informed by such
17 factors as the spatial extent and grain of available data, and the configuration of landscape features.
18 Landscape and environmental data are inherently spatial, and must be acquired, organized, and
19 analyzed in the course of a landscape genetics study. Thus, geoscientists and geocomputation will
20 play an increasingly important role in landscape genetics.

21 Progress in landscape genetics is so far limited by available analytical methods (Balkenhol et al.,
22 2009, 2016a; Guillot et al., 2009). In part this derives from the fact that many of the available
23 analytical tools and much of the usable software were originally developed for population genetics
24 or even broader statistical applications. They often include assumptions and are applicable to data
25 that are not completely appropriate for landscape genetics studies. Because of this gap, there is
26 no consensus in the literature regarding how to approach landscape genetics analysis (Balkenhol
27 et al., 2016a). Indeed, the *ad hoc* assortment of methods currently in use lacks a unifying theory;
28 consequently, more focus must be given to a mechanistic understanding of the influence of landscapes
29 and environments on genetic and genomic variation (Balkenhol et al., 2016b). Development of
30 a more comprehensive theory will come in part from an improved foundation of open source
31 computational tools allowing explicit and flexible mechanistic modeling.

32 This brief review focuses on three themes. First, it identifies the types of models most likely to
33 advance a comprehensive theory of landscape genetics, improve mechanistic understanding, and
34 provide better predictions serving, for example, conservation policy and management. Second,
35 it considers a set of open source software that could be used for general models in landscape
36 genetics but that all have significant limitations. Finally, it also suggests how these limitations can
37 be overcome with new models and computational tools.

38 1 LANDSCAPE GENETICS AND BAYESIAN INFERENCE

39 The prevailing challenge in landscape genetics is identifying the mechanisms by which landscape
40 and environmental factors influence genetic and genomic variation. More precisely, the central
41 question is: given data on intraspecific genetic variation across landscapes (or waterscapes; Manel
42 and Holderegger (2013); Selkoe et al. (2016)), what inferences are possible regarding the functional
43 mechanisms and factors causing that variation? Framing the question in this way emphasizes the
44 inherent connection between the science of landscape genetics and the nature of Bayesian inference.

45 The natural connection between landscape genetics and Bayesian inference has led to the
46 development of a variety of widely used Bayesian analysis methods. A first set of these includes
47 STRUCTURE, which identifies putative populations and assigns individuals to them (Pritchard et al.,
48 2000). Although originally designed for population not landscape genetics, it remains the most
49 widely used. A second set of Bayesian models applied to landscape genetics includes GENELAND,
50 which seeks to identify population clusters by modeling allele frequency distributions in a spatially
51 explicit way (Chen et al., 2007; Guillot et al., 2005a,b). More recently, Bayesian models that
52 explicitly relate environmental gradients to spatially explicit allele frequency distributions have been
53 developed (Coop et al., 2010; Frichot et al., 2013).

54 One element is common to all of the available software: each program implements a narrow
55 range of possible models and provides very limited opportunity for expanding its scope. For example,
56 as discussed below, both STRUCTURE and GENELAND are essentially variants of the same model,
57 yet nothing of their implementation is shared so new variants cannot be created by exploiting their
58 commonality. Further, the published descriptions do not reveal the inherent similarity between
59 STRUCTURE and GENELAND, so conceptual connections are not evident. Consequently, landscape
60 geneticists do not recognize a continuum of possible models. Even worse, they cannot exploit the
61 continuum by incrementally modifying existing models and competing alternatives against available

62 data. This is a serious limitation for a scientific field that repeatedly asserts that more mechanistic
63 and predictive models and a stronger theoretical foundation are essential (Andrew et al., 2013;
64 Balkenhol et al., 2016b; Guillot et al., 2009; Manel and Holderegger, 2013).

65 **2 PROBABILISTIC GRAPH MODELS**

66 Mathematical graphs are widely used to represent models, including some in landscape genetics.
67 Graphs are composed of a set of vertices and a set of edges, each of which connects a pair of vertices.
68 Edges may be directed or undirected, and paths are sequences of edges connecting one vertex with
69 another, possibly with intervening vertices. A cyclic graph has at least one path starting and ending
70 at the same vertex; an acyclic graph lacks any such paths.

71 One application of graphs to landscape genetics derives from the population graph concept (Dyer
72 and Nason, 2004). Here the graph is composed of vertices representing population distributions in a
73 multilocus genetic space, and edges representing interdependencies between populations due, for
74 example, to gene flow (Excoffier et al., 1992). The primary application to landscape genetics has
75 been identification of conditional independence between populations to remove edges followed by
76 analysis of graph structure metrics such as centrality or connectness (Dyer, 2007; Murphy et al.,
77 2016).

78 Graph models can be much richer, however, and both STRUCTURE and GENELAND are examples
79 used in landscape genetics. Generally, (probabilistic) graph models are composed of vertices
80 representing any kind of random variable and edges representing dependencies between them
81 (Bishop, 2006; Koller and Friedman, 2009). They are widely used, for example, in latent factor
82 analysis (Steyvers and Griffiths, 2007), a field that now finds application broadly in machine learning,
83 artificial intelligence, and document and image processing, as well as landscape genetics (Blei et al.,
84 2003; Blei, 2012; Frichot et al., 2013; Jia et al., 2011; Pritchard et al., 2000). The population
85 graph concept of Dyer and Nason (2004) is clearly a special case where each vertex represents
86 the same quantity, a population-specific distribution, but the landscape genetics analysis involving
87 edge removal and graph metrics (Murphy et al., 2016) is unrelated to the use of graphs as formal
88 probabilistic models (Bishop, 2006; Koller and Friedman, 2009). The value of the latter for landscape
89 genetics, both conceptually and for software development, is the focus here.

90 Although not described as such, a probabilistic graph model represents the mathematics underlying
91 STRUCTURE (Pritchard et al., 2000). In this case, the random variables represent population-
92 specific distributions of alleles, the probabilistic assignment of alleles to populations, and prior
93 distributions that by default are uninformative (Figure 1). The STRUCTURE software supports slight
94 variations in the model depicted; for example, assignment of all alleles may be individual-specific
95 not allele-specific as shown, and priors may be informative in various ways. These variations,
96 however, are extremely limited and do not cover the continuum of related models that is possible.

97 One related model, however, is alluded to in Pritchard et al. (2000) and described in detail in
98 Falush et al. (2003); but again, the graph model itself is not presented explicitly. The main difference
99 is that in this model the population-specific allele distributions are not independent; instead, they are
100 correlated via a shared ancestral population (Figure 2).

101 A further related model, implemented in GENELAND, is described in Guillot et al. (2005a),
again without depicting the graph model (Figure 3). This model explicitly adds spatial information

103 to the model; unlike the other two, both the identity of alleles and their spatial location are observed.
104 This supports estimating additional random variables such as the inferred location of individuals and
105 spatially-explicit allele distributions.

106 A comparison of Figures 1–3 makes clear that these are all closely related models, a fact that is
107 generally not made evident by the papers describing them. Furthermore, in many ways the graph
108 models are more useful than the papers, because they make the conceptual linkages clear and enable
109 direct comparisons among them. They also make gaps in the existing models evident; for example,
110 none of these include gene flow explicitly despite its clear importance as a mechanism in landscape
111 genetics (Holderegger and Wagner, 2008; Manel and Holderegger, 2013; Storfer et al., 2007; van
112 Strien et al., 2014). Finally, probabilistic graph models invite the construction of variations by adding
113 new random variables or changing dependencies among them, because the biological structure of the
114 models is easy to reason about when presented in the form of a graph. Probabilistic graph models,
115 therefore, provide an ideal foundation for mechanistic modeling in landscape genetics that can lead
116 to an improved theoretical understanding.

117 **3 A MECHANISTIC MODELING FRAMEWORK FOR LANDSCAPE GE-** 118 **NETICS**

119 Traditional approaches to landscape genetics descriptively model either genetic characteristics
120 associated with each sampled site or individual, or derived genetic measures associated with pairs
121 of sampled sites or individuals (Joost et al., 2007). Almost all approaches model these response
122 variables using *ad hoc* distributions taken from more generic statistical literature; for example,
123 virtually the entire textbook on landscape genetics (Balkenhol et al., 2016a) follows this pattern.
124 In contrast, a mechanistic approach would construct a model of the individual observations, e.g.,
125 individual multilocus genotypes (or genomes), as a function of assumed demographic, ecological,
126 and population genetic mechanisms.

127 As described earlier and illustrated in Figures 1–3, STRUCTURE and GENELAND are examples of
128 exactly this approach; the observed alleles are modeled directly in terms of unobserved but inferable
129 populations and assignments (Guillot et al., 2005a; Pritchard et al., 2000). Viewed in this context,
130 differences between individual- and population-based approaches to landscape genetics are not
131 fundamental; rather they reduce to simple differences between the structure of the graphical models
132 in use. Individual-based models have graphs that relate observations on individuals to individual-
133 specific random variables; examples of the latter are the assignment of an individual's alleles to
134 populations (Z in Figures 1 and 3) and the inferred true location of each individual (s in Figure 3).
135 Population-based models have graphs that relate observations on individuals to population-specific
136 random variables; examples of the latter are the population-specific allele frequencies (P in Figures 1
137 and 3). By including elements of each, Figures 1 and 3 already blur the boundary between individual-
138 and population-specific models.

139 Given the power of probabilistic graph models to represent a broad spectrum of intermediate
140 cases just as well, a better framework is the set of mechanisms included. From this perspective, it is
141 evident that Figure 3 includes spatially-explicit mechanisms whereas Figure 1 does not. It is also
142 evident that neither one includes an explicit mechanism for gene flow. The power of probabilistic
143 graph models lies in their ability to cover the entire spectrum of models relevant to landscape

144 genetics and to encourage more transparent reasoning about alternative models. Using them to
145 advance landscape genetics is limited only by our ability to compare alternative models, but that in
146 turn is severely constrained by the software available to manipulate and analyze them.

147 **4 OPEN-SOURCE PROBABILISTIC GRAPH MODELS**

148 As just illustrated, the primary advantages of probabilistic graph models are that complex and
149 realistically mechanistic models can be constructed, and that their model structure can be manipulated
150 easily to explore alternatives. Thus, there is great scope for constructing general theories based upon
151 manipulating probabilistic graph models to reflect interesting biological models within landscape
152 genetics. However, software tools must exist that enable manipulation and analysis of the graphs,
153 and the types of graphs available must match those required by landscape genetics. For many
154 applications two types of graphs are enough: Bayesian networks represented by directed acyclic
155 graphs (DAGs) and Markov random fields represented by undirected graphs. Landscape genetics
156 models, however, often require more general types of graphs to accommodate, for example, spatially
157 autoregressive relationships among random variables. Additionally, landscape genetics models often
158 require distributions appropriate to a broad range of commonly encountered data types, including
159 alleles, genotypes, spatially explicit environmental data. Such a range of discrete and continuous,
160 unidimensional and multidimensional data types requires a rich array of probability distributions.

161 While the set of probabilistic graph models that has been applied to landscape genetics do not
162 harness their full flexibility, there exist modeling software that does better. The most widely used is
163 based upon the BUGS language for describing graph models, and includes WinBUGS, OpenBugs
164 (Lunn et al., 2009) and JAGS (Plummer, 2015). The BUGS language allows textual description
165 of general graph models that include a broad range of distributions. The textual description is
166 translated into executable code, a process that introduces some of the limitations common to this
167 type of modeling software. First, the flexibility of possible applications is limited by the features
168 of the BUGS language. A limited range of data types, generally scalars and vectors or matrices
169 constructed from them, is available, only data structures describable in the language may be used,
170 and algorithms are limited to those already programmed. Second, the scale of models is also limited
171 by the execution environment provided by the implementation. Despite the inherent flexibility
172 of graph models in general, both of these limitations are barriers to convenient development of
173 landscape genetics models that leverage the flexibility of graph models. While genetic data can
174 be recoded in the form of only integers or real numbers, it is tedious and error-prone to do so;
175 thus, the limited data types available create needless barriers. A landscape genetics model might
176 include thousands or millions of random variables within it; consider, for example, a model of
177 population allele frequencies and environmental factors across a landscape grid of 1000×1000 pixels.
178 This puts severe stress on models that cannot harness the full power of multithreading, distributed
179 multiprocessing, and careful memory management. Being limited by the BUGS language, these
180 programs provide restricted capacity for modelers to address these issues.

181 Another general graph modeling system is Stan (Carpenter et al., 2015; Gelman et al., 2015).
182 Although more flexible in some ways than BUGS, Stan suffers from some of the same limitations that
183 reduce its applicability to landscape genetics. It has the same limited data types and the execution
184 environment is likewise limited by the Stan language. As a result, neither BUGS nor Stan are ideally

Name	Graph types	Primitive variables	Preprocessing	Implementation language	Reference
Darwin	FGs	scalars	compiled	C++	Gould (2015)
HYDRA	DAGs, MRFs, FGs, HMMs	Java classes	compiled	Java	Warmes (2013)
Infer.NET	FGs	C# classes	compiled	C#	Minka et al. (2014)
JAGS	DAGs	scalars	interpreted	C++	Plummer (2016)
JavaBayes	DAGs	scalars	interpreted	Java	Cozman (2001)
libDAI	FGs	discrete	compiled	C++	Mooij (2015)
Mocapy++	DAGs, HMMs	C++ classes	compiled	C++	Antonov et al. (2015)
Nimble	DAGs	scalar	interpreted	C++	de Valpine et al. (2016)
OpenBUGS	DAGs	scalar	interpreted	Component Pascal	Thomas (2009)
OpenGM	DAGs, MRFs, FGs	discrete	compiled	C++	OpenGM (2015)
PNL	DAGs, MRFs	C++ classes	compiled	C++	Sysoyev et al. (2013)
RISO	DAGs	Java classes	compiled	Java	Dodier (2012)
Stan		scalars	interpreted	C++	Stan Development Team (2016)
Vibes	DAGs	scalar	compiled	Java	Winn (2004)

Table 1. A selection of open source software tools for analyzing probabilistic graph models. Type of graphs include directed acyclic graphs (DAGs), Markov random fields (MRFs), factor graphs (FGs), hidden Markov models (HMMs), and Gaussian Markov models (GMMs).

185 suited for landscape genetics applications.

186 In addition to these two major classes of graph modeling software, a broad range of more
 187 specialized software systems is also available; many of these are summarized by Murphy (2014).
 188 Some are open source and may have potential for landscape genetics applications (Table 1). These
 189 tools have many of the same limitations as BUGS, JAGS, and Stan. They often handle fewer graph
 190 types than needed for landscape genetics, the data types are not well suited to landscape genetics, or
 191 their execution environments are restrictive. In addition, they are much more specialized, difficult to
 192 program, and likely well beyond the reach of typical landscape geneticists. These characteristics
 193 mean that landscape geneticists face a fundamental challenge hindering development of a strong
 194 conceptual foundation for the field based upon the expressive power, flexibility, and statistical rigor
 195 of probabilistic graph models.

196 5 DESIGNING A PROBABILISTIC GRAPH MODEL FOR LANDSCAPE 197 GENETICS

198 What then is the ideal design of a software system intended to harness the power, flexibility, and
 199 rigor of probabilistic graph models applied to landscape genetics? First and foremost, it must support
 200 a full range of relevant graph types, which in particular means not being limited to directed acyclic
 201 graphs. Second, it must support a full range of useful data types that landscape geneticists work
 202 with; in addition to simple scalars, vectors, and matrices, these include named alleles and genotypes,
 203 loci and chromosomes, geographic locations, and spatial data of various sorts. Ideally, user-defined
 204 or third-party data types should be easy to accommodate. Third, the algorithms available should be
 205 extensible to allow improved efficiency as needed. Fourth, the execution environment should not
 206 be limited to that encapsulated within a single, predefined program. This is especially important
 207 for landscape genetics models that may well encompass thousands or millions of random variables.
 208 Finally, the power and flexibility of graph models must be abstracted enough that a full spectrum

209 of landscape geneticists can create simple models easily, test alternative and biologically relevant
210 models quickly, and improve upon the models and algorithms as needed.

211 It is little surprise that existing software tools are unable to meet these stringent demands; they
212 are largely conflicting and impossible to resolve without advanced software design. The most likely
213 path forward (Lunn et al., 2009) leverages the power of C++ to present high-level abstractions
214 based upon embedded domain specific languages (de Guzman and Kaiser, 2017; Niebler, 2017)
215 assembled with expression templates (Niebler, 2017; Veldhuizen, 1995) from highly reusable generic
216 components (Stepanov and Rose, 2014). We are following these design principles to implement
217 a software library, GRAPHMODEL, intended to provide the expressive power and computational
218 performance demanded for advancing a coherent conceptual foundation for landscape genetics.

219 Design of any software library must face a fundamental tension between expressive power and
220 ease of use for a limited set of use cases. For example, a variety of statistical software packages
221 aim to make a limited range of analyses easy for newcomers, but R (R Core Team, 2017) is gaining
222 widespread use because it is a Turing-complete language that can express an expansive set of models.
223 In the case of GRAPHMODEL, we have focused initially on providing a set of generic components
224 that can be composed flexibly to develop an expansive set of models based upon probabilistic graph
225 models. Future work will provide increasingly higher levels of abstraction to simplify common use
226 cases. Note that the alternative of starting at a high level of abstraction, i.e., restricting the graph
227 models that are possible, is incompatible with the realization described here that probabilistic graph
228 models are a powerful and natural tool for landscape genetics and other fields.

229 The outcome of this work is a highly compact way of encoding probabilistic graph models
230 of relevance to landscape genetics and other fields of science. Given the expressive power of the
231 language, all of this should be readily accessible to biologists without deep knowledge of C++
232 programming. Importantly, models can be described in a formal way that removes the ambiguity
233 inherent in natural language descriptions. Finally, because models are encoded directly in C++,
234 not interpreted, they can be reused as portions of larger programs for enhanced capability; this
235 is fundamentally impossible for interpreted modeling frameworks such as OpenBUGS or JAGS.
236 The generality of this approach removes the limitations inherent to the available software and
237 characteristic of current approaches to landscape genetics data analysis, and ultimately will make it
238 easy to encode, and therefore explore, the complete space of relevant models. Some of the features
239 of the GRAPHMODEL library that make this possible are outlined in the following sections.

240 **Graph model vertices** Probabilistic graph models are of course composed of vertices and edges.
241 Each vertex represents one of several different types of concepts, including scalar and non-scalar
242 random variables, arbitrary expressions, factors that support calculation of a probability density
243 function, distributions that support sample generation, and scalar distributions that support calcula-
244 tion of a cumulative density function. For purposes of supporting Monte Carlo Markov Chains
245 (MCMCs), it is also useful if random variables can summarize a sequence of their own values. All
246 of these concepts are encapsulated within the GRAPHMODEL library as a hierarchical set of classes
247 (Figure 4). Importantly, each type of vertex also models the concepts of a vertex in an incidence
248 graph as defined by the Boost Graph library (Siek et al., 2002, 2017). Furthermore, other types of
249 graphs, e.g., a vertex and edge list graph, can be constructed from a set of vertices. As a result,
250 any appropriate graph algorithm based upon Boost Graph concepts may be used on probabilistic

Distribution	Implementation	
	PDF/CDF	Random variate generator
Bernoulli	boost/math/distributions/bernoulli.hpp	boost/random/bernoulli_distribution.hpp
Beta	boost/math/distributions/beta.hpp	boost/random/beta_distribution.hpp
Categorical		boost/random/discrete_distribution.hpp
Dirichlet	boost/math/special_functions/gamma.hpp	boost/random/gamma_distribution.hpp
Multinomial	boost/math/special_functions/binomial.hpp	boost/random/discrete_distribution.hpp
Normal	boost/math/distributions/normal.hpp	boost/random/normal_distribution.hpp
Uniform	boost/math/distributions/uniform.hpp	boost/random/uniform_01.hpp

Table 2. Probability distributions. For distributions with scalar support, both the probability density (PDF) and cumulative density (CDF) functions are implemented; otherwise, only the PDF is implemented. For all distributions, generation of random variates is implemented. Where appropriate, these are implemented as wrappers around standard functions available in the Boost.Math and Boost.Random libraries.

251 graph models described using the GRAPHMODEL library. This is one of the important abstractions
 252 illustrating the power of a generic library for probabilistic graph models.

253 **Probability distributions** At the core, probability distributions are fundamental to any probabilistic
 254 graph model. Any factor, distribution, or scalar distribution vertex in a graph can be associated
 255 with an appropriate probability distribution at run-time. The most important distributions for
 256 landscape genetics are implemented in the GRAPHMODEL library, mostly as simple wrappers
 257 around the corresponding Boost distributions and generators (Table 2). Given the generic nature
 258 of the design, extending the library with new distributions based upon pre-existing mathematical
 259 and statistical libraries is straightforward. For example, the Boost Math library (Agrawal et al.,
 260 2017) implements 33 distinct statistical distributions and the Boost Random library (Maurer, 2017)
 261 implements 28 random number distributions. This represents a rich set of extensions that will be
 262 added to GRAPHMODEL. Other libraries could, of course, be used as the source of additional
 263 distributions.

264 **Expressions** Leveraging the power of probabilistic graph models for computational modeling
 265 requires construction of arbitrary expressions that, for example, represent the value of a parameter for
 266 a distribution. In C++, expression templates are a powerful mechanism of representing expression
 267 trees (Niebler, 2017; Veldhuizen, 1995). Although earlier versions used Boost.Proto (Niebler, 2017),
 268 the GRAPHMODEL library currently uses the Yap expression template library (Laine, 2016), because
 269 of its greater power, compactness, and expressiveness. This enables, for example, expressions like

270 `lit(p) + sample(normal_distribution(mean=0, standard_deviation=0.1))`

271 to represent a random walk sampler that generates samples as deviates from the current value of a
 272 random variable p , which might differ each time a sample is generated. Use of template expressions

Name	Description
<code>abs()</code>	Absolute value
<code>exp()</code>	Exponential
<code>log()</code>	Logarithm
<code>pow()</code>	Power
<code>sqrt()</code>	Square root

Table 3. Mathematical functions. These mathematical functions are implemented as expression templates and therefore can be used as primitives in mathematical expressions.

273 like this that capture natural mathematical statements as executable computations is one of the
274 powerful mechanisms for achieving flexibility and generality in the GRAPHMODEL library.

275 **Function expressions** One expectation for mathematical expressions is that they include functions
276 such as `log()` or `sqrt()`. Mathematical software libraries, of course, provide a rich set of such
277 functions, but not in a form amenable to expression templates. The GRAPHMODEL library, however,
278 already implements the most common (Table 3). More importantly, construction of new expressions
279 for mathematical (or other) functions is straightforward; all that is required is a class that wraps
280 the mathematical function of interest, a pair of functions for evaluating the function within an
281 expression tree, and a set of functions that construct the expression from its arguments (Figure 5).
282 This pattern can be easily repeated to extend the set of mathematical functions available within the
283 GRAPHMODEL library.

284 **Execution environment** The execution environment for any modeling software is crucial, as it often
285 determines the performance and therefore the set of problems that are feasible to solve. Just as
286 one design goal for the GRAPHMODEL library is to support arbitrary probabilistic graph models,
287 another is to avoid any limitations on the execution environment. Two of the performance critical
288 elements of evaluating a probabilistic graph model are calculating joint probability distributions and
289 generating an MCMC sample for a potentially large set of random variables. Both of these might
290 benefit from parallel, asynchronous computation, but especially the latter must be done in a way
291 that avoids inherent dependencies among random variables. Any practical computation will likely
292 require mixtures of sequential and asynchronous computations. Further, the choice should be in
293 the hands of the model developer, not imposed by the execution environment. These design goals
294 are addressed in the GRAPHMODEL library by allowing run-time definition of the policies used to
295 evaluate joint probability distributions and generate MCMC samples. By default the policies perform
296 calculations sequentially and can be ignored for simple models, which are unlikely to benefit from
297 asynchronous computation. However, alternative policies are possible and the library provides one
298 based upon the Boost Asynchronous library (Henry, 2015), which contains a wide range of parallel
299 asynchronous algorithms that go to great lengths to avoid any waiting for task completion. Boost
300 Asynchronous also provides threadpools that can distribute tasks across a cluster of machines. Thus,
301 composing applications for distributed asynchronous model computation is also supported with
302 no modification to the core GRAPHMODEL classes. Furthermore, run-time selection of sequential

303 or asynchronous computation may be made at the level of individual random variables or MCMC
304 generators, which provides great flexibility in the execution environment.

305 **Data sources** Any modeling software must interact with a variety of sources of data; indeed, the
306 generality of probabilistic graph models would be useless unless a diversity of data types can be
307 associated with random variables or expressions. This flexibility is supported in the GRAPHMODEL
308 library in several ways. First, all concrete types, including the variate type of random variables, the
309 result type of expressions, and the probability type for PDF calculations, are template parameters
310 for all the relevant classes. Therefore, they can be selected arbitrarily by the modeler. For example,
311 the `random_variable` class can represent a numerical scalar, a boolean value, a discrete valued
312 variable, or a vector depending on the variate template argument. Indeed, any type that can participate
313 in the expressions used in the model is a legitimate source of data, so the library can be extended in
314 arbitrary ways with user-defined types used as template arguments. Second, external data can be
315 read from a variety of data sources. One common source of data is from a file containing a dataframe,
316 one form of which is the traditional tab-delimited file created by spreadsheets or other software. The
317 GRAPHMODEL library provides support for reading dataframe files and allowing expressions to
318 reference, not copy, individual elements within a dataframe. For example, the following code reads
319 a dataframe and creates a reference of type `double` to the first sample.

```
320 auto dataframe = read_dataframe("dataframe.dat");  
321 auto dataframe_element = make_element<double>(dataframe, 0, "sample");
```

322 In the field of landscape genetics, another common source of information is georeferenced raster
323 or vector files in any of a large number of commonly used formats. The GRAPHMODEL library
324 includes an interface to the GDAL library (Open Source Geospatial Foundation, 2017), which
325 includes drivers for 142 raster formats and 84 vector formats. As with dataframes, the provided
326 interface supports associating expressions with values obtained from a dataset; a random variable
327 can, for example, represent the elevation at a particular location in space. Finally, because the
328 GRAPHMODEL library is not a separate language but is a domain-specific language within C++,
329 the full power of C++ and any possible libraries are available for interacting with data sources.
330 Arbitrary code or third-party libraries may be used to access data and associate it with random
331 variables, distributions, or expressions used in a probabilistic graph model.

332 **Probability calculations** All computationally efficient representations of real numbers are approxi-
333 mate and cover a restricted subset. This can be a serious problem when calculating joint probability
334 distributions, because they often involve products of a very large number of terms. Naïve solutions
335 based upon, for example, native data types can easily result in underflow errors, which are usually
336 silent yet yield completely erroneous results. The concrete probability type for random variables and
337 distributions is a template parameter and thus can be selected by the modeler to avoid these problems.
338 One option provided by the GRAPHMODEL library is a numeric type storing its value internally
339 on a logarithmic scale, but implementing the normal arithmetic operators (e.g., $+$, $-$, $*$, $/$, $\%$) and
340 functions as efficiently as possible. For example, the default lognumeric type (which is a base e
341 `double`) and a base 10 `float` alternative are declared as follows:

```
342 using probability_type = lognumeric<>;  
343 using probability_type = lognumeric<float, base::ten>;
```

344 Another option is to use one of many available arbitrary precision numeric libraries that are readily
345 available. Because all of the code in the GRAPHMODEL library is generic, any type that implements
346 arithmetic operators and functions appropriately can be used for probability calculations.

347 **Future development** The fundamental design goals focusing on supporting generic and flexible
348 probabilistic graph models have largely been accomplished in the implementation of the GRAPH-
349 MODEL library. Arbitrarily complex graph models can be composed, joint probability distributions
350 calculated, and samples generated from the distribution of random variables. While this already
351 supports a wide range of applications in landscape genetics and other fields, several important
352 advances remain for future development. Because the library purposely provides great flexibility and
353 generality so as not to limit its applicability, it necessarily presents a relatively low level of abstrac-
354 tion. Thus, an important direction for future development is to provide higher layers of software that
355 increase the level of abstraction, thereby further increasing the library's expressiveness. One example
356 is additional overloading of operators to reduce boilerplate when associating probability distributions
357 with random variables. A small addition to the library could enable the model of STRUCTURE
358 to be expressed very compactly (Figure 6), which would also enable biologists to explore related
359 models easily. A second direction for future development is to expand the range of probability
360 distributions and mathematical functions that can be used as primitives within expressions. Given the
361 idiomatic nature of the wrappers, this is a very straightforward task that could rapidly lead to scores
362 of new distributions and functions. A third direction for future development is to create a variety of
363 applications aimed at particular classes of models. For example, STRUCTURE and GENELAND could
364 be reimplemented easily; one benefit would be the large-scale parallelization inherently provided by
365 the GRAPHMODEL library, something that would require a complete redesign and reimplementation
366 to add to STRUCTURE or GENELAND.

367 The value of a generic library is evident in this list of future directions: the fact that each of
368 these is a straightforward task is a consequence of a solid foundation that can be easily extended in a
369 variety of different ways. The widespread use of class and function templates, expression templates,
370 static type safety coupled with run-time type hiding when appropriate, and clear association between
371 computational components and the concepts of probabilistic graph models has yielded highly flexible
372 software that can be composed into a variety of models. Reliance on generic programming allows
373 the library to deduce much about the types in use and combine them correctly.

374 6 CONCLUSION

375 Landscape genetics suffers greatly from the absence of an analytical foundation that encourages
376 development of a mechanistic understanding of the impact of environmental and landscape factors
377 on genetic and genomic variation (Balkenhol et al., 2016a). This stems in part from the adoption of
378 software tools and methods originally developed for other purposes. There exist well-established
379 concepts and statistical approaches associated with probabilistic graph models that are ideally
380 suited as the needed foundation for landscape genetics. Unfortunately, the associated software tools
381 cannot be borrowed directly, because they are limited in ways that do not accommodate the needs of
382 landscape geneticists. One priority that would directly advance the field and resolve these problems
383 is the development of probabilistic graph model tools that do apply generally to landscape genetics.
384 Despite the inherent difficulty of this task, we have developed a suitable library and are beginning to

385 apply it to landscape genetics.

386 7 SUPPLEMENTAL MATERIALS

387 The source code for the GRAPHMODEL library, version 0.1.2, is available as a supplemental
388 compressed tar file `graph_model-0.1.2.tgz`.

389 8 ACKNOWLEDGEMENTS

390 This paper was improved by comments offered by Stéphane Joost, Olivier Ertz, Jordan Golubov,
391 María del Carmen Mandujano, and Gregory Penn. I am indebted to discussions with Gary Roemer,
392 who continually presses me to think synthetically about the issues presented here. The U.S. Forest
393 Service and Department of State both supported projects that motivated the need for probabilistic
394 graph models in landscape genetics.

395 REFERENCES

- 396 Agrawal, N., Bikineev, A., Bristow, P. A., Holin, H., Guazzone, M., Kormanyos, C., Lalande, B.,
397 Maddock, J., Murphy, J. W., Råde, J., Sobotta, B., Sewani, G., van den Berg, T., Walker, D., and
398 Zhang, X. (2017). Math Toolkit 2.5.2. [http://www.boost.org/doc/libs/1_64_0/
399 libs/math/doc/html/index.html](http://www.boost.org/doc/libs/1_64_0/libs/math/doc/html/index.html).
- 400 Andrew, R. L., Bernatchez, L., Bonin, A. L., Buerkle, C. A., Carstens, B. C., Emerson, B. C., Garant,
401 D., Giraud, T., Kane, N. C., Rogers, S. M., Slate, J., Smith, H., Sork, V. L., Stone, G. N., Vines,
402 T. H., Waits, L., Widmer, A., and Rieseberg, L. H. (2013). A road map for molecular ecology.
403 *Molecular Ecology*, 22:2605–2626.
- 404 Antonov, L., Paluszewski, M., and Hamelrcyk, T. (2015). Mocapy++. [https://sourceforge.
405 net/projects/mocapy/](https://sourceforge.net/projects/mocapy/).
- 406 Balkenhol, N., Cushman, S. A., Storfer, A. T., and Waits, L. P., editors (2016a). *Landscape genetics:
407 concepts, methods, applications*. Wiley Blackwell, Hoboken, New Jersey.
- 408 Balkenhol, N., Cushman, S. A., Waits, L. P., and Storfer, A. (2016b). Current status, future
409 opportunities, and remaining challenges in landscape genetics. In Balkenhol, N., Cushman, S. A.,
410 Storfer, A. T., and Waits, L. P., editors, *Landscape genetics: concepts, methods, applications*,
411 chapter 14, pages 247–255. Wiley Blackwell, Hoboken, New Jersey.
- 412 Balkenhol, N., Gugerli, F., Cushman, S. A., Waits, L. P., Coulon, A., Arntzen, J. W., Holderegger,
413 R., Wagner, H. H., and Participants of the Landscape Genetics Research Agenda Workshop 2007
414 (2009). Identifying future research needs in landscape genetics: where to from here? *Landscape
415 Ecology*, 24:455–463.
- 416 Bishop, C. M. (2006). *Pattern recognition and machine learning*. Springer, New York, New York.
- 417 Blei, D. M. (2012). Probabilistic topic models. *Communications of the ACM*, 55:77–84.
- 418 Blei, D. M., Ng, A. Y., and Jordan, M. I. (2003). Latent dirichlet allocation. *Journal of Machine
419 Learning Research*, 3:993–1022.
- 420 Carpenter, B., Lee, D., Gelman, A., Goodrich, B., Guo, J., Hoffman, M., Betancourt, M., Li, P.,
421 Brubaker, M. A., and Riddell, A. (2015). Stan: a probabilistic programming language. *Journal of
422 Statistical Software*.

- 423 Chen, C., Durand, E., Forbes, F., and François, O. (2007). Bayesian clustering algorithms ascertain-
424 ing spatial population structure: a new computer program and a comparison study. *Molecular*
425 *Ecology Notes*, 7:747–756.
- 426 Coop, G., Witonsky, D., Di Rienzo, A., and Pritchard, J. K. (2010). Using environmental correlations
427 to identify loci underlying local adaptation. *Genetics*, 185:1411–1423.
- 428 Cozman, F. G. (2001). JavaBayes: Bayesian networks in Java. [http://www.cs.cmu.edu/
429 ~javabayes/index.html](http://www.cs.cmu.edu/~javabayes/index.html).
- 430 Cushman, S. A., McRae, B. H., and McGarigal, K. (2016). Basics of landscape ecology: an
431 introduction to landscapes and population processes for landscape genetics. In *Landscape*
432 *Genetics: Concepts, Methods, Applications*, chapter 2, pages 11–34. Wiley Blackwell, first
433 edition.
- 434 de Guzman, J. and Kaiser, H. (2017). Spirit 2.5.2. [http://www.boost.org/doc/libs/1_
435 64_0/libs/spirit/doc/html/index.html](http://www.boost.org/doc/libs/1_64_0/libs/spirit/doc/html/index.html).
- 436 de Valpine, P., Turek, D., Paciorek, C., Temple Lang, D., and Bodik, R.
437 (2016). Nimble: numerical inference for hierarchical models using Bayesian
438 and likelihood estimation. [https://bids.berkeley.edu/research/
439 nimble-numerical-inference-hierarchical-models-using-bayesian-and-likelih](https://bids.berkeley.edu/research/nimble-numerical-inference-hierarchical-models-using-bayesian-and-likelihood)
- 440 Dodier, R. (2012). RISO: distributed belief networks. [https://sourceforge.net/
441 projects/riso/](https://sourceforge.net/projects/riso/).
- 442 Dyer, R. J. (2007). The evolution of genetic topologies. *Theoretical Population Biology*, 71:71–79.
- 443 Dyer, R. J. and Nason, J. D. (2004). Population graphs: the graph theoretic shape of genetic structure.
444 *Molecular Ecology*, 13:1713–1727.
- 445 Excoffier, L., Smouse, P. E., and Quattro, J. M. (1992). Analysis of molecular variance inferred from
446 metric distances among DNA haplotypes: application to human mitochondrial DNA restriction
447 data. *Genetics*, 131:479–491.
- 448 Falush, D., Stephens, M., and Pritchard, J. K. (2003). Inference of population structure using
449 multilocus genotype data: linked loci and correlated allele frequencies. *Genetics*, 164:1567–1587.
- 450 Fricot, E., Schoville, S. D., Bouchard, G., and François, O. (2013). Testing for associations
451 between loci and environmental gradients using latent factor mixed models. *Molecular Biology*
452 *and Evolution*, 30:1687–1699.
- 453 Gelman, A., Lee, D., and Guo, J. (2015). Stan: A probabilistic programming language for Bayesian
454 inference and optimization.
- 455 Gould, S. (2015). DARWIN: a framework for machine learning and computer vision research and
456 development. <http://drwn.anu.edu.au>.
- 457 Guillot, G., Estoup, A., Mortier, F., and Cosson, J. F. (2005a). A spatial statistical model for
458 landscape genetics. *Genetics*, 170:1261–1280.
- 459 Guillot, G., Leblois, R., Coulon, A., and Frantz, A. C. (2009). Statistical methods in spatial genetics.
460 *Molecular Ecology*, 18:4734–4756.
- 461 Guillot, G., Mortier, F., and Estoup, A. (2005b). GENELAND: a computer package for landscape
462 genetics. *Molecular Ecology Notes*, 5:712–715.
- 463 Henry, C. (2015). Boost Asynchronous. [http://htmlpreview.github.io/?https://
464 github.com/henry-ch/asynchronous/blob/master/libs/asynchronous/
465 doc/asynchronous.html](http://htmlpreview.github.io/?https://github.com/henry-ch/asynchronous/blob/master/libs/asynchronous/doc/asynchronous.html). Note: this library is not yet an official part of Boost.

- 466 Holderegger, R., Kamm, U., and Gugerli, F. (2006). Adaptive vs. neutral genetic diversity: implica-
467 tions for landscape genetics. *Landscape Ecology*, 21:797–807.
- 468 Holderegger, R. and Wagner, H. H. (2008). Landscape genetics. *BioScience*, 58:199–207.
- 469 Jia, Y., Salzmann, M., and Darrell, T. (2011). Learning cross-modality similarity for multinomial
470 data. In *Proceedings of the 2011 International Conference on Computer Vision, ICCV '11*, pages
471 2407–2414, Washington, DC. IEEE Computer Society.
- 472 Joost, S., Bonin, A., Bruford, M. W., Després, L., Conord, C., Erhardt, G., and Taberlet, P. (2007).
473 A spatial analysis method (SAM) to detect candidate loci for selection: towards a landscape
474 genomics approach to adaptation. *Molecular Ecology*, 16:3955–3969.
- 475 Koller, D. and Friedman, N. (2009). *Probabilistic graphical models: principles and techniques*.
476 MIT Press, Cambridge, MA.
- 477 Laine, T. Z. (2016). Boost.YAP (proposed). [https://tzlaine.github.io/yap/doc/
478 html/index.html](https://tzlaine.github.io/yap/doc/html/index.html). Note: this library is not yet an official part of Boost.
- 479 Lunn, D., Spiegelhalter, D., Thomas, A., and Best, N. (2009). The BUGS project: evolution, critique
480 and future directions. *Statistics in Medicine*, 28:3049–3067.
- 481 Manel, S. and Holderegger, R. (2013). Ten years of landscape genetics. *Trends in Ecology and
482 Evolution*, 28:614–621.
- 483 Manel, S., Joost, S., Epperson, B. K., Holderegger, R., Storfer, A., Rosenberg, M. S., Scribner,
484 K. T., Bonin, A., and Fortin, M.-J. (2010). Perspectives on the use of landscape genetics to detect
485 genetic adaptive variation in the field. *Molecular Ecology*, 19:3760–3772.
- 486 Manel, S., Schwartz, M. K., Luikart, G., and Taberlet, P. (2003). Landscape genetics: combining
487 landscape ecology and population genetics. *Trends in Ecology and Evolution*, 18:189–197.
- 488 Maurer, J. (2017). Boost.Random. [http://www.boost.org/doc/libs/1_64_0/doc/
489 html/boost_random.html](http://www.boost.org/doc/libs/1_64_0/doc/html/boost_random.html).
- 490 Minka, T., Winn, J., Guiver, J., Webster, S., Zaykov, Y., Yangel, B., Spengler, A., and Bronskill, J.
491 (2014). Infer.NET 2.6. [http://research.microsoft.com/en-us/um/cambridge/
492 projects/infernet/default.aspx](http://research.microsoft.com/en-us/um/cambridge/projects/infernet/default.aspx).
- 493 Mooij, J. (2015). libDAI: a free and open source C++ library for discrete approximate inference in
494 graphical models. <https://staff.fnwi.uva.nl/j.m.mooij/libdai/>.
- 495 Murphy, K. (2014). Software packages for graphical models. [https://www.cs.ubc.ca/
496 ~murphyk/Software/bnsoft.html](https://www.cs.ubc.ca/~murphyk/Software/bnsoft.html).
- 497 Murphy, M., Dyer, R., and Cushman, S. A. (2016). Graph theory and network models in landscape
498 genetics. In Balkenhol, N., Cushman, S. A., Storfer, A. T., and Waits, L. P., editors, *Landscape
499 genetics: concepts, methods, applications*, chapter 10, pages 165–179. Wiley Blackwell, Hoboken,
500 New Jersey.
- 501 Niebler, E. (2017). Boost.Proto. [http://www.boost.org/doc/libs/1_64_0/doc/
502 html/proto.html](http://www.boost.org/doc/libs/1_64_0/doc/html/proto.html).
- 503 Open Source Geospatial Foundation (2017). GDAL: Geospatial Data Abstraction Library. [http:
504 //www.gdal.org](http://www.gdal.org).
- 505 OpenGM (2015). OpenGM. <http://hciweb2.iwr.uni-heidelberg.de/opengm/>.
- 506 Plummer, M. (2015). JAGS version 4.0.0 user manual. Technical report.
- 507 Plummer, M. (2016). JAGS. <http://mcmc-jags.sourceforge.net>.
- 508 Pritchard, J. K., Stephens, M., and Donnelly, P. (2000). Inference of population structure using

- 509 multilocus genotype data. *Genetics*, 155:945–959.
- 510 R Core Team (2017). *R: A Language and Environment for Statistical Computing*. R Foundation for
511 Statistical Computing, Vienna, Austria.
- 512 Selkoe, K. A., Scribner, K. T., and Galindo, H. M. (2016). Waterscape genetics—applications of
513 landscape genetics to rivers, lakes, and seas. In Balkenhol, N., Cushman, S. A., Storfer, A. T.,
514 and Waits, L. P., editors, *Landscape genetics: concepts, methods, applications*, chapter 13, pages
515 220–246. Wiley Blackwell, Hoboken, New Jersey.
- 516 Siek, J., Lee, L.-Q., and Lumsdaine, A. (2002). *The Boost Graph Library: user guide and reference*
517 *manual*. Addison-Wesley.
- 518 Siek, J. S., Lee, L.-Q., and Lumsdaine, A. (2017). The Boost Graph library (bgl). [http://www.
519 boost.org/doc/libs/1_64_0/libs/graph/doc/index.html](http://www.boost.org/doc/libs/1_64_0/libs/graph/doc/index.html).
- 520 Stan Development Team (2016). The Stan math library, version 2.10.0. <http://mc-stan.org>.
- 521 Stepanov, A. A. and Rose, D. E. (2014). *From Mathematics to Generic Programming*. Addison-
522 Wesley, first edition.
- 523 Steyvers, M. and Griffiths, T. (2007). Probabilistic topic models. In Landauer, T., McNamara, D.,
524 Dennis, S., and Kintsch, W., editors, *Latent Semantic Analysis: A road to meaning*. Laurence
525 Erlbaum.
- 526 Storfer, A., Murphy, M. A., Evans, J. S., Goldberg, C. S., Robinson, S., Spear, S. F., Dezzani, R.,
527 Delmelle, E., Vierling, L., and Waits, L. P. (2007). Putting the ‘landscape’ in landscape genetics.
528 *Heredity*, 98:128–142.
- 529 Sysoyev, A. V., Milch, B., Bradski, G. R., and Dash, D. (2013). Probabilistic networks library.
530 <https://sourceforge.net/projects/openpnl/>.
- 531 Thomas, A. (2009). OpenBugs. <http://www.openbugs.net/w/FrontPage>.
- 532 van Strien, M. J., Keller, D., Holderegger, R., Ghazoul, J., Kienast, F., and Bolliger, J. (2014).
533 Landscape genetics as a tool for conservation planning: predicting the effects of landscape change
534 on gene flow. *Ecological Applications*, 24:327–339.
- 535 Veldhuizen, T. (1995). Expression templates. *C++ Report*, 7(5):26–31.
- 536 Warmes, G. (2013). HYDRA MCMC library. [https://sourceforge.net/projects/
537 hydra-mcmc/](https://sourceforge.net/projects/hydra-mcmc/).
- 538 Winn, J. (2004). Variational inference for Bayesian networks. [http://vibes.sourceforge.
539 net](http://vibes.sourceforge.net).

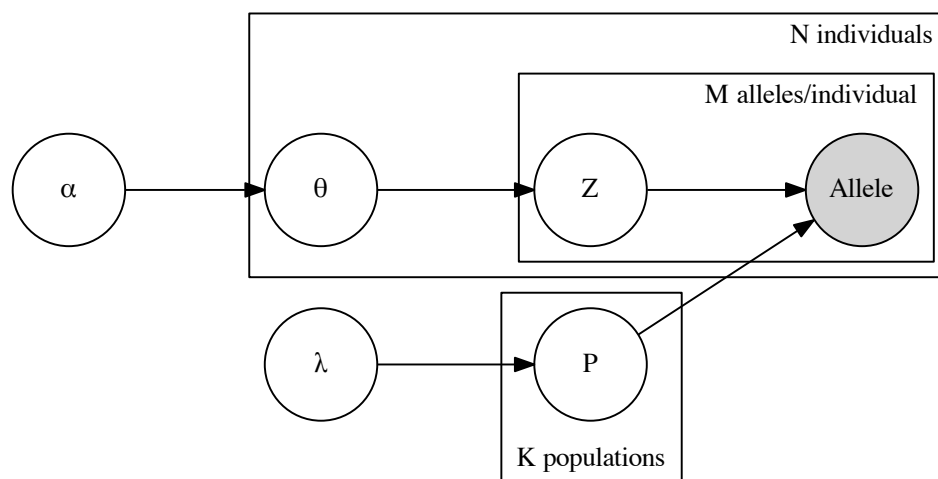


Figure 1. Plate notation (Bishop, 2006) for the locus-specific graph model used by STRUCTURE (Pritchard et al., 2000). Each circle represents a random variable (or a set of them for those enclosed within boxes) and each arrow represents a dependency of one random variable upon another. This models N individuals each sampled for M (usually two) alleles. P represents the allele frequency distribution in each of K populations and Z represents the assignment of alleles to populations. θ is the distribution of assignments and α and λ are Bayesian priors. The single filled circle indicates that among these random variables only the alleles have been observed; the rest are inferred (or fixed in the case of α and λ).

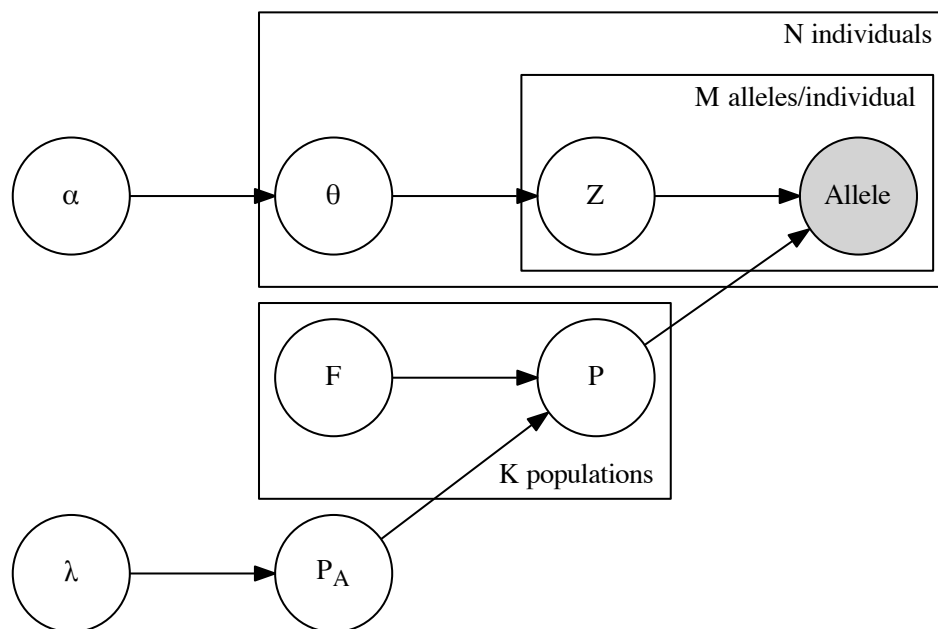


Figure 2. Plate notation for the correlated allele frequency extension (Falush et al., 2003) to the locus-specific graph model used by STRUCTURE. This models an ancestral population (P_A) from which a correlated set of extant populations (P) have been derived. The pattern of correlation between populations is governed by F .

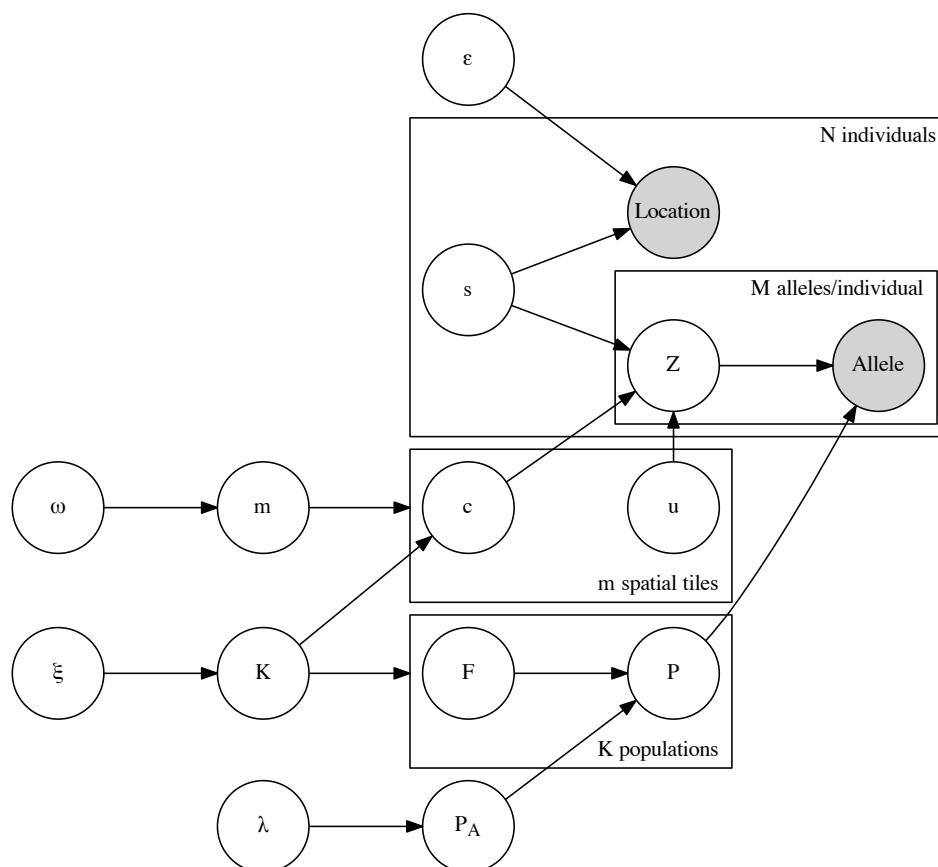


Figure 3. Plate notation for the spatially-explicit extension of STRUCTURE used by GENELAND (Guillot et al., 2005a,b). Additional random variables include the true (s) and observed (shaded) locations of sampled individuals and the error (ϵ) between them, and the locations of points defining the Voronoi tessellation (u) and their population identity (c). In this case, both the number of Voronoi cells (m) and the number of populations (K) are random variables.

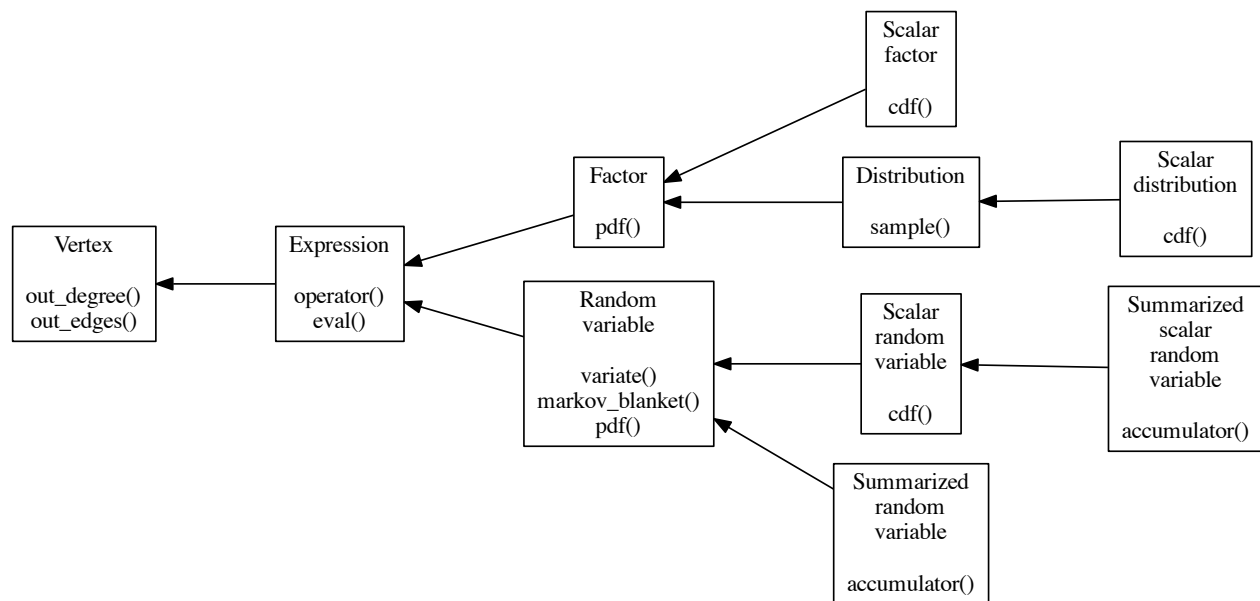


Figure 4. Conceptual class hierarchy of graph model vertices. Each vertex in a probabilistic graph model corresponds conceptually to one of these basic concepts. Additionally, the run-time behavior of the classes can be modified. For example, the specific distribution represented by a vertex can be modified at run-time by replacing its corresponding strategy.

```

// A class wrapping the log() function provided in the C library
//
template < typename Expression >
class log_function
{
    using self = log_function;
public:
    template < typename E
        , typename = mpl::enable_constructor_t<self,E>
        >
    explicit log_function (E&& expression)
        : expression_(std::forward<E>(expression))
    {}
    template < typename ... T >
    auto operator () (T&& ... t) const
    {
        using std::log;
        return log(expression_(std::forward<T>(t)...));
    }
private:
    Expression expression_;
};

// Evaluate a terminal containing a log_function value
//
template < typename ... T, typename ... Args >
auto transform_expression (expression_terminal<log_function<T...>> expr,
    Args&& ... args)
{ return evaluate_terminal(expr, std::forward<Args>(args)...); }

// Evaluate a terminal containing a reference to a log_function
//
template < typename ... T, typename ... Args >
auto transform_expression (expression_terminal<log_function<T...>&> expr,
    Args&& ... args)
{ return evaluate_terminal(expr, std::forward<Args>(args)...); }

// Construct a log expression function
//
template < typename Wrapper >
auto log (expression_function<Wrapper>&& function)
{
    using function_type = expression_function<Wrapper>;
    using log_function_type = log_function<function_type>;
    return make_expression_function(log_function_type(std::move(function)));
}

template < typename Expr >
auto log (Expr&& expr)
{ return log(make_expression_function(std::forward<Expr>(expr))); }

template < typename Expression >
auto log (log_function<Expression> const& expr) { return expr; }

template < typename Expression >
auto log (log_function<Expression>&& expr) { return std::move(expr); }

```

Figure 5. Implementation of the `log()` expression function.

```
observed_allele_type X;
allele_assignment_type Z;
individual_admixture_distribution_type theta;
population_allele_frequency_distribution_type P;
diriclet_parameter_type alpha;
diriclet_parameter_type lambda;

allele_frequency_type Pr;

for (auto population : populations)
  P(population) =~ dirichlet(lambda);
for (auto individual : individuals)
{
  theta(individual) =~ dirichlet(alpha);
  for (auto allele : alleles(individual))
  {
    Z(individual,allele) =~ multinomial(theta(individual));
    for (auto population : populations)
      Pr(individual) += Z(population,individual) * P(population);
    X(individual,allele) =~ bernoulli(Pr(individual,allele));
  }
}
```

Figure 6. Compact implementation of the STRUCTURE model with admixture (Pritchard et al., 2000). This is C++ source code for the probabilistic graph model corresponding to one of the models in STRUCTURE. A few additional lines of code transforms this into a model with correlated allele frequencies (Falush et al., 2003) or one with spatially explicit observations (Guillot et al., 2005a).