

Appendix

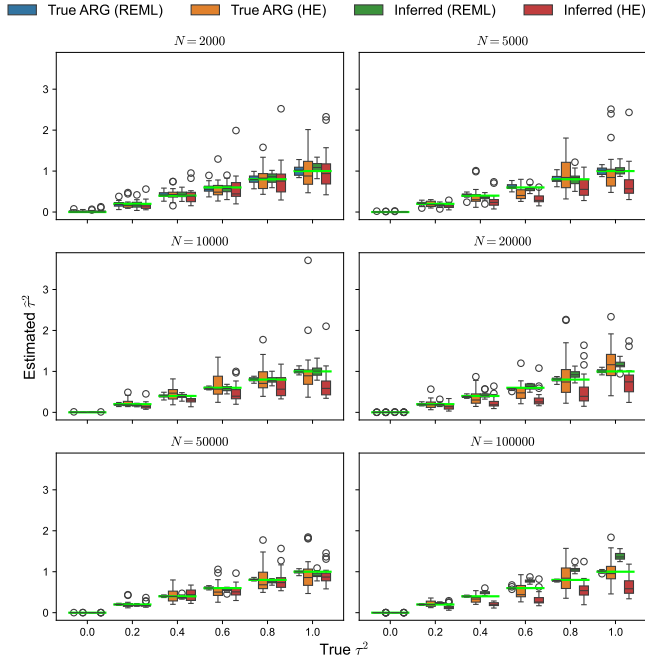


Figure A1 Statistical precision of the HE method and AI-REML. By default, the branch GRM is scaled by 1 over $4 \sum_T s_T t_{\text{root}, T}$. The values in the labels “HE (value)” refer to the additional scaling we apply to the GRM. Each boxplot shows the distribution of estimators over 100 replicates.

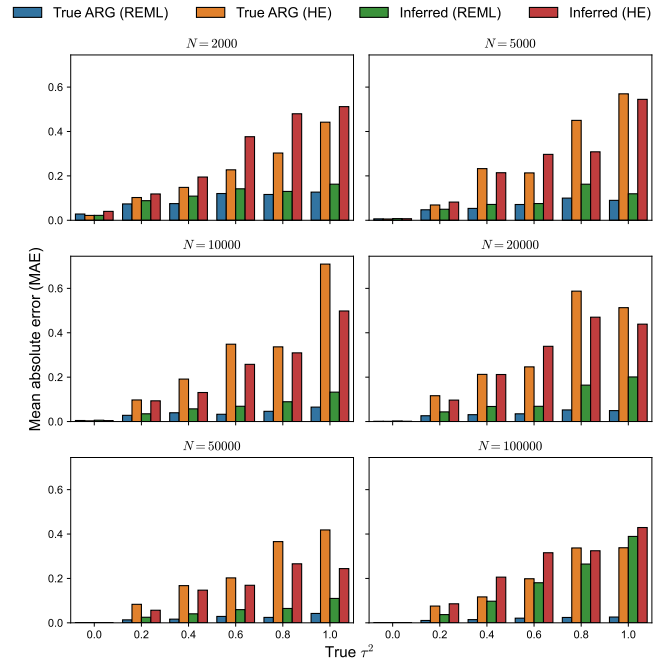


Figure A2 Statistical precision of the HE method and AI-REML. By default, the branch GRM is scaled by 1 over $4 \sum_T s_T t_{\text{root}, T}$. The values in the labels “HE (value)” refer to the additional scaling we apply to the GRM. The mean absolute error (MAE) of the estimators with respect to the true parameters $\sigma_\epsilon^2 = 1$ and $\tau^2 = 1$.

in Salehi Nowbandegani et al. (2023).) Nevertheless, this splitting is merely conceptual and never takes place in the algorithms we’ll demonstrate shortly, as it significantly degrades the storage efficiency of the tree sequence data structure.

We consider complex traits that follow a linear model. Write $\mathbf{G} \in \mathbb{R}^{N \times P}$ for the $N \times P$ genotype matrix for N genomes and P sites in the genome, whose $(i, j)^{\text{th}}$ entry is the number of derived alleles carried by the i^{th} genome at the j^{th} site. Then the trait model is:

$$\mathbf{y} = \mathbf{G}\boldsymbol{\beta} + \boldsymbol{\varepsilon} = \sum_{p=1}^P \mathbf{G}_p \boldsymbol{\beta}_p + \boldsymbol{\varepsilon}, \quad (\text{A1})$$

where $\boldsymbol{\beta} \in \mathbb{R}^P$ is a vector defining an effect size for the derived allele(s) at each site. For now, we condition on $\boldsymbol{\beta}$ so that we treat it as a fixed quantity. With infinite-sites mutation model, each site has at most two alleles, so this effect is the allele substitution effect (Falconer 1995), also called the site effect, variant effect, or mutation effect in this work. One cannot estimate $\boldsymbol{\beta}$ ’s entries separately because some variants appear on the same edge (or on the two edges pendant to the root). Columns of \mathbf{G} corresponding to these variants will be identical, leading to perfect collinearity (linkage-disequilibrium), making them statistically indistinguishable. We will group these indiscernible variants according to their edge membership. After conditioning on \mathcal{T} , the remaining variation in \mathbf{G} is solely due to the mutational process. We assume that mutations occur on edge e following a Poisson process at a rate proportional to the total number of base-pairs (span) multiplied by the amount of time the edge represents (length), $l_e s_e$. We let the per unit time per base-pair mutation rate u_{ep} vary across edges and positions. If we let $\mathbf{1}_{ep}$

2 Model formulation and assumptions

3 Expressing a complex trait on an ARG

We show the correspondence between site and branch (edge) based trait expressions. We define an ARG $\mathcal{T} = (\mathcal{N}, \mathcal{E})$ as a collection of nodes \mathcal{N} and edges \mathcal{E} . Each node n has a (birth) time ago t_n , and each edge e indicates inheritance of a genomic interval $[\ell_e, r_e]$ by a child $c(e)$ from a parent $p(e)$ (Wong et al. 2024). We call these “parent” and “child” in graph terms, so the inheritance might be over more than one generation. To describe a valid ARG, the spans of all edges with the same child must be disjoint, and the time of the parent must be larger than the time of the child for each edge. We will also call the *span* of an edge $s_e = r_e - \ell_e$ (in number of base-pairs), and the *length* of an edge $l_e = t_{p(e)} - t_{c(e)}$ (in time units, such as generations). All expectations and variances are conditioned on \mathcal{T} , which accounts for the variation in observed genome data due to recombination and drift, and hence also the implicit population structure and pedigree of individuals in \mathcal{T} . Only the mutational process is left random in this setting. This setting was explored in Ralph (2019) and expanded in Ralph et al. (2020). In those papers, the utility of treating mutations as random on a fixed ARG was to understand how well a given SNP genotype (site) statistic could be expected to estimate the corresponding summary of tree shapes.

For conceptual clarity, we assume in the following that the edges are in fact *sub-edges*: each has a unique sub-topology along its span. (Although this is not true in the case of tree sequences in general, we can enforce this by appropriate splitting of edges as

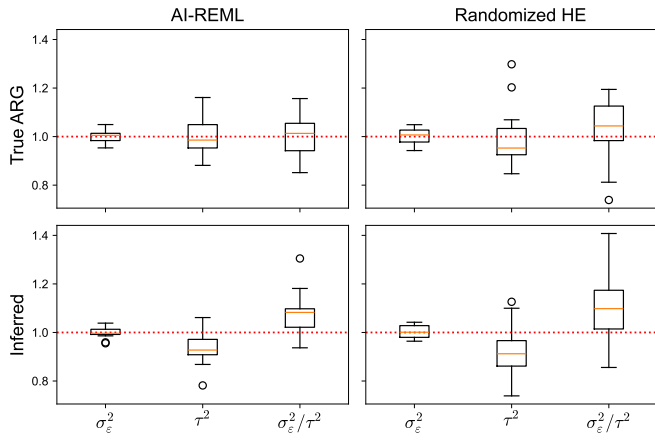


Figure A3 Statistical precision of the HE method and AI-REML in the spatial simulation benchmark. 20 replicate experiments were conducted.

1 be an indicator variable that is 1 if a mutation has arrived on
2 edge e at position p and 0 otherwise, $\mathbb{E}[\mathbf{1}_{ep}] = l_e u_{ep}$.

3 Recurrent mutations are allowed; however, mutations cannot
4 occur below an existing one. This is a slight relaxation of the
5 infinite-sites model to allow multiple recurrent mutations in
6 unrelated lineages but no back mutation. Define $\mathbf{Z} \in \mathbb{R}^{N \times E}$ so
7 that \mathbf{Z}_{ie} is the number of chromosomes of individual i that is
8 a descendant of edge e . \mathbf{Z}_{ie} ranges from 0 to 2 in the case of
9 diploids. We can express \mathbf{G} with \mathbf{Z} by:

$$\mathbf{G}_{ip} = \sum_{e:p \in e} \mathbf{Z}_{ie} \mathbf{1}_{ep}, \quad (\text{A2})$$

10 where $p \in e$ means edge e contains the position p , and define \mathbf{G}_p
11 as the p -th column of \mathbf{G} . In words, one can determine individual
12 i 's genotype at position p by first examining if edge e is ancestral
13 to i (\mathbf{Z}_{ie}) and then checking if the edge has a mutation at p
14 ($\mathbf{1}_{ep}$). Finally, we sum up such contributions across all edges
15 ($\sum_{e:p \in e}$). This also illustrates how `tstrait` computes the trait
16 values: for a position p in a local tree, it travels down the edges
17 (or equivalently branches for now) of the tree and checks if
18 there's a mutation along the path from the root to the tips. Here,
19 \mathbf{Z}_{ie} tells if an edge in $\{e : p \in e\}$ is on the path from the root to a
20 tip, which is individual i .

21 Using equation (A2), we can collapse variants corresponding
22 to perfectly correlated columns of \mathbf{G} into the edge on which they
23 reside. This is a natural choice because mutations on the same
24 edge are inherited by the same set of individuals (recall that the
25 edge's subtree is constant by assumption). Substituting (A2) in

the linear equation (A1) gives:

$$\begin{aligned} \mathbf{y} &= \sum_{p=1}^P \mathbf{G}_p \boldsymbol{\beta}_p + \boldsymbol{\varepsilon} \\ &= \sum_{p=1}^P \left(\sum_{e:p \in e} \mathbf{Z}_e \boldsymbol{\beta}_p \mathbf{1}_{ep} \right) + \boldsymbol{\varepsilon} \\ &\because \text{substitute (A2)} \\ &= \sum_{e=1}^E \sum_{p:p \in e} \mathbf{Z}_e \boldsymbol{\beta}_p \mathbf{1}_{ep} + \boldsymbol{\varepsilon} \\ &\because \text{swap the sums over } e \text{ and } p \\ &= \sum_{e=1}^E \mathbf{Z}_e \left(\sum_{p:p \in e} \boldsymbol{\beta}_p \mathbf{1}_{ep} \right) + \boldsymbol{\varepsilon} \\ &\because \mathbf{Z}_e \text{ does not depend on } p \\ &= \sum_{e=1}^E \mathbf{Z}_e \mathbf{v}_e + \boldsymbol{\varepsilon} \quad \because \mathbf{v}_e = \sum_{p:p \in e} \boldsymbol{\beta}_p \mathbf{1}_{ep} \\ &= \mathbf{Z} \mathbf{v} + \boldsymbol{\varepsilon}. \end{aligned} \quad (\text{A3})$$

The key step is at the third line which swaps the summation
over edges e and sites p . Then, we pulled out \mathbf{Z}_e out of the inner
sum over p and merged the inner sum into \mathbf{v}_e . This derivation
presents a way to convert a genotype-based expression of a trait
 $\mathbf{y} = \sum_p \mathbf{G}_p \boldsymbol{\beta}_p + \boldsymbol{\varepsilon}$ into a branch-based expression $\mathbf{y} = \sum_e \mathbf{Z}_e \mathbf{v}_e + \boldsymbol{\varepsilon}$.

Linear algebra formulation

We now connect this linear model to classical formulations in
quantitative genetics and selective breeding literature on the
pedigree model (Henderson 1976; Quaas 1988). While we work
with a linear model on an ARG, not pedigree, our aim here is to
highlight the inherent sparse structure of phenotype generative
models and the critical connection with sparse linear algebra
and efficient computational algorithms (Rue and Held 2005). We
start by noting that sub-edges are also edges – each has a parent
and child node and a genomic interval – but with the property
(as above) that the subtree below them in the ARG does not
change. These can be obtained as follows: iterating along the
genome, each time the local tree changes, it changes through
addition or removal of an edge. If the parent of the newly added
or removed edge is n , then we move up through the local tree,
and split each edge we encounter. For instance, if n is not a root
then there is an edge with child n and parent p , and interval
 $[\ell, r]$ that spans the current position x . By “splitting” this edge
we mean to replace it with two edges: one with interval $[\ell, x]$
and the other with interval $[x, r]$. The resulting shorter edges
will be all sub-edges.

We say that sub-edge e_1 *directly inherits* from sub-edge e_2 if
 $p(e_1) = c(e_2)$ and their genomic intervals overlap. In fact, it
then follows that the interval of e_2 is contained within that of
 e_1 : in other words, $[\ell(e_2), r(e_2)] \subseteq [\ell(e_1), r(e_1)]$. Now, define
the square matrix \mathbf{A} to have rows and columns indexed by sub-
edges, with $\mathbf{A}_{ef} = 1$ if sub-edge e directly inherits from f , and 0
otherwise. This \mathbf{A} is \mathbf{P} in Quaas (1988). The *sub-edge inheritance*
matrix is then $\mathbf{I} + \mathbf{A} + \mathbf{A}^2 + \dots = (\mathbf{I} - \mathbf{A})^{-1}$: this has (e, f) th
entry equal to 1 if $e = f$ or if e inherits from f (i.e., if the edge
represented by e carries genetic material inherited also along f),
and 0 otherwise. Note that the seemingly infinite series above
is in fact a finite sum because the depth of an ARG is finite.
Now, write \mathbf{M} for the *individual-sub-edge incidence matrix*: for an

individual i and a sub-edge e , let $\mathbf{M}_{ie} = 1$ if $c(e) \in i$ and $\mathbf{M}_{ie} = 0$ otherwise. $c(e) \in i$ means individual i contains node $c(e)$. The matrix \mathbf{Z} from the main text is thus:

$$\mathbf{Z} = \mathbf{M}(\mathbf{I} - \mathbf{A})^{-1}.$$

The directed graph whose adjacency matrix is \mathbf{A} , the *sub-edge graph*, is a multitree, and has many convenient properties. For instance, many standard ARG computations that require left-to-right iteration can be implemented as bottom-up or top-down on the sub-edge graph. This property is shared by the genome representation graph of DeHaas et al. (2024). Typically, there will be a multiplicative factor of order $\log N$ more sub-edges than edges, where N is the sample size. However, this difference can be large in practice as the constant depends on demographic history. An advantage to working with the sub-edge graph is that it allows use of more general graph-based tools, as for instance by Salehi Nowbandegani et al. (2023). However, we are not aware of a reason that such algorithms are necessarily more efficient.

This formulation is useful conceptually, but we did not find it to be useful computationally, because typical left-to-right algorithms on the ARG naturally work with sub-edges without explicitly constructing them (for instance, those in Ralph et al. (2020) or Zhu et al. (2025)). We are not aware of any argument that algorithms using a structure like the sub-edge graph are inherently more efficient than algorithms that efficiently use the tree sequence.

Sub-edges, and the “splitting” procedure are very similar to the “bricks” and the “bricking” operation used by Salehi Nowbandegani et al. (2023). However, “bricks” are defined so that they have a unique set of descendant nodes, rather than a unique subtree. Since subtrees can change but still have the same set of descendants, bricks are thus larger than sub-edges, while retaining the useful properties above. However, the brick graph is no longer a multitree. We chose the more strict definition mainly because it does not require retaining and checking the descendant nodes at each step.

The linear algebra formulation shows the sparse structure of our trait generative model on an ARG, enabling scalable algorithms as shown in Lehmann et al. (2025) and this study. For completeness, we point to Misztal (2015), who followed a similar goal – enabling inexpensive computations with the inverse of genotype GRM by avoiding inversion of a large GRM. Instead of focusing on the trait generative model, Misztal (2015) split the genotyped sample into core and non-core sets of individuals and expressed genetic values of non-core individuals conditional on core individuals. This conditioning also admits a sparse inverse of genotype GRM and is technically the same as conditioning on “inducing point” or “knots” in spatial modeling (e.g., Heaton et al. 2018). While this core conditioning approach is computationally very efficient for populations with limited genetic diversity, such as in selective breeding, it does not scale as well to more diverse settings as the work presented in this study.

The branch-based phenotype model is a linear mixed model

The last expression we arrived at expresses a phenotype in terms of edges:

$$\mathbf{y} = \sum_e \mathbf{Z}_e \mathbf{v}_e + \boldsymbol{\varepsilon}. \quad (\text{A4})$$

We refer to \mathbf{v}_e as the edge effect and we shall characterize its probabilistic features such as its distribution. We go back to the definition of the edge effect and observe that it is a collection of indicator random variables driven by mutation:

$$\mathbf{v}_e = \sum_{p:p \in e} \beta_p \mathbf{1}_{ep}. \quad (\text{A5})$$

In turn, \mathbf{v}_e as a whole is a random variable. Since mutations occur at a low rate, \mathbf{v}_e is likely to be zero most of the time, but conditioning on the occurrence, a mutation hits position p at a probability that is proportional to the position’s mutation rate u_{ep} . The effect size of this mutation is β_p . Altogether, \mathbf{v}_e is zero or a number drawn from the pool of effect sizes $\{\beta_p\}_{p \in e}$ with probability proportional to the mutation rate u_{ep} . Hence, the distribution is determined by the collection $\{(\beta_p, u_{ep})\}_{p:p \in e}$.

This formulation closely mirrors the continuum-of-alleles (CoA) model proposed by Kimura and Crow (1964). Similar to an edge that contains many positions, a locus also stretches over a genome segment that is subject to mutation. When a mutation arrives, it occurs at a random position yielding a different allele every time, i.e., the infinite-alleles model. The CoA model describes the distribution of the effect size of these newly emerging alleles.

Our branch based expression in equation (A4) represents a phenotype by a design matrix with effect sizes that are random with respect to mutations. Hence, it can be considered as a linear random effects model. Furthermore, some features of the model can be deduced from the properties of the mutational process and the topology of the ARG. The variance of an edge can be computed as follows:

$$\begin{aligned} \text{Var}(\mathbf{v}_e) &= \text{Var}\left(\sum_{p:p \in e} \beta_p \mathbf{1}_{ep}\right) \\ &= \sum_{p:p \in e} \beta_p^2 \text{Var}(\mathbf{1}_{ep}) \\ &\because \text{mutations at different } p \text{ are independent} \\ &= \sum_{p:p \in e} \beta_p^2 l_e u_{ep} (1 - l_e u_{ep}) \\ &\because \text{Var}(\mathbf{1}_{ep}) = \mathbb{E}[\mathbf{1}_{ep}] (1 - \mathbb{E}[\mathbf{1}_{ep}]) \\ &\approx \sum_{p:p \in e} \beta_p^2 l_e u_{ep} \\ &\because l_e u_{ep} \ll 1 \text{ i.e., mutation rate is small} \\ &= \underbrace{l_e}_{\text{edge area} = A_e} \cdot \underbrace{\frac{1}{s_e} \sum_{p:p \in e} \beta_p^2 u_{ep}}_{\substack{\text{size average of squared} \\ \text{effect sizes times mutation} \\ \text{rate} =: \tau_e^2}} \\ &\because l_e \text{ does not depend on } p \\ &= A_e \tau_e^2 \end{aligned} \quad (\text{A6})$$

The variance of an edge effect is proportional to its area A_e . τ_e^2 is the average squared effect size weighted by mutation rates. In words, regions with higher functional importance (large β_p^2) and mutation rate (large u_{ep}) contribute more to branch/edge effect variation. Assuming that mutation rates are constant and the effect sizes are evenly dispersed across the genome, similar to the infinitesimal model, this recovers the ARG-LMM model we have used in the main portion of this paper.

In the main portion of the paper, the branch effects were assumed to be mean zero and mutually independent. A derivation

of these (approximate) properties helps shed light on how and when this is a good assumption. To do this, we decompose \mathbf{v}_e into two parts $\mathbf{u}_e = \mathbf{v}_e - \mathbb{E}[\mathbf{v}_e]$ and $\mathbf{f}_e = \mathbb{E}[\mathbf{v}_e]$. The vector of these quantities are defined accordingly: $\mathbf{u} = \mathbf{v} - \mathbb{E}[\mathbf{v}]$ and $\mathbf{f} = \mathbb{E}[\mathbf{v}]$. Then, we can rewrite equation (A4) as:

$$\mathbf{y} = \mathbf{Zf} + \mathbf{Zu} + \boldsymbol{\varepsilon}. \quad (\text{A7})$$

Henceforth, we refer to \mathbf{u}_e instead of \mathbf{v}_e as the edge effect. From the definition, the branch effect \mathbf{u}_e has mean zero. At first glance, enforcing zero mean by subtracting the mean looks like an unfounded assumption that sweeps the mean contributions \mathbf{Zf} under the rug. Surprisingly, if all individuals exist at the same time, this is not an issue. Under neutral evolutionary scenarios of various kinds, we may assume that the mutation rate is constant for each position (Ewens 2004; Wakeley 2008; Durrett 2008), i.e., $u_{ep} = u_p$ for all e such that $p \in e$ (edge e contains position p). In this case, \mathbf{Zf} reduces to a constant:

$$\begin{aligned} [\mathbf{Zf}]_i &= \sum_{e=1}^E \mathbf{Z}_{ie} \mathbb{E} \left[\sum_{p:p \in e} \beta_p \mathbf{1}_{ep} \right] \\ &\because \mathbf{f}_e = \mathbb{E}[\mathbf{v}_e] \\ &= \sum_{e=1}^E \sum_{p:p \in e} \mathbf{Z}_{ie} \beta_p l_e u_p \\ &\because \mathbb{E}[\mathbf{1}_{ep}] = l_e u_{ep} \text{ and } u_{ep} = u_p \\ &= \sum_{p=1}^P \sum_{e:p \in e} \mathbf{Z}_{ie} \beta_p l_e u_p \\ &\because \text{swap the sums over } e \text{ and } p \\ &= \sum_{p=1}^P \beta_p u_p \left(\sum_{e:p \in e} \mathbf{Z}_{ie} l_e \right) \\ &\because \beta_p \text{ and } u_p \text{ do not depend on } e \\ &= \sum_{p=1}^P \beta_p u_p \cdot 2t_{\text{root},p} \quad \because \text{ploidy} = 2 \\ &= \text{constant with respect to } i. \end{aligned} \quad (\text{A8})$$

Note that $\sum_{e:p \in e} \mathbf{Z}_{ie} l_e$ is the sum of lengths of all edges ancestral to individual n that adds up to the root time at position p multiplied by ploidy. This means that the fixed component of the genetic effect is fully accounted for by a model intercept.

We can further generalize equation (A8) to incorporate time varying mutation rates. In this setting, we stipulate that a mutation rate at a position only depends on time, i.e., the mutation

rate is a function of time $u_p(t)$:

23

$$\begin{aligned} [\mathbf{Zf}]_i &= \sum_{e=1}^E \mathbf{Z}_{ie} \mathbb{E} \left[\sum_{p:p \in e} \beta_p \mathbf{1}_{ep} \right] \\ &= \sum_{e=1}^E \sum_{p:p \in e} \mathbf{Z}_{ie} \beta_p \mathbb{E}[\mathbf{1}_{ep}] \\ &\because \mathbf{Z}_{ie} \text{ does not depend on } p \\ &= \sum_{e=1}^E \sum_{p:p \in e} \mathbf{Z}_{ie} \beta_p \int_{t(e(c))}^{t(e(p))} u_p(t) dt \\ &\because \mathbb{E}[\mathbf{1}_{ep}] = \int_{t(e(c))}^{t(e(p))} u_p(t) dt \\ &= \sum_{p=1}^P \sum_{e:p \in e} \mathbf{Z}_{ie} \beta_p \int_{t(e(c))}^{t(e(p))} u_p(t) dt \\ &\because \text{swap the sums over } e \text{ and } p \\ &= \sum_{p=1}^P \beta_p \sum_{e:p \in e} \mathbf{Z}_{ie} \int_{t(e(c))}^{t(e(p))} u_p(t) dt \\ &\because \beta_p \text{ does not depend on } e \\ &= \sum_{p=1}^P \beta_p \left(2 \int_0^{t_{\text{root},p}} u_p(t) dt \right) \quad \because \text{ploidy} = 2 \\ &= \text{constant. w.r.t. } i. \end{aligned} \quad (\text{A9})$$

This confirms that the fixed portion \mathbf{Zf} of equation (A7) is constant in a fairly general setting with time varying mutation rates.

24

25

In order to demonstrate independence, we show that $\text{Cov}(\mathbf{u}_e, \mathbf{u}_{e'}) = 0$ for all pairs $e \neq e'$. This implies independence because indicator variables are independent if and only if they are uncorrelated, similar to Gaussian random variables. We first expand $\text{Cov}(\mathbf{u}_e, \mathbf{u}_{e'})$ and reduce it to a sum of covariances of indicators:

26

27

28

29

30

31

$$\begin{aligned} \text{Cov}(\mathbf{u}_e, \mathbf{u}_{e'}) &= \text{Cov} \left(\sum_{p:p \in e} \beta_p \mathbf{1}_{ep}, \sum_{p':p' \in e'} \beta_{p'} \mathbf{1}_{e'p'} \right) \\ &= \sum_{p \in e, p' \in e'} \beta_p \beta_{p'} \text{Cov}(\mathbf{1}_{ep}, \mathbf{1}_{e'p'}) \\ &\because \text{Cov is bilinear} \\ &= \sum_{p \in e, e'} \beta_p^2 \text{Cov}(\mathbf{1}_{ep}, \mathbf{1}_{e'p}) \\ &\because \text{mutations at different } p \text{ are independent.} \end{aligned} \quad (\text{A10})$$

The last line's simplification follows from the independence of indicator variables $\mathbf{1}_{ep}$ across different positions. When e and e' have no overlapping positions, the summation is trivially zero as $\{p : p \in e, e'\}$ will be an empty set. It's also zero if the two edges are not ancestral to each other, as mutations running on

32

33

34

35

36

1 separate edges are also independent. Otherwise:

$$\begin{aligned}
& \text{Cov}(\mathbf{1}_{ep}, \mathbf{1}_{e'p}) \\
&= \mathbb{E}[\mathbf{1}_{ep}\mathbf{1}_{e'p}] - \mathbb{E}[\mathbf{1}_{ep}] \mathbb{E}[\mathbf{1}_{e'p}] \\
&\because \text{Cov}(X, Y) = \mathbb{E}[XY] - \mathbb{E}[X] \mathbb{E}[Y] \\
&= \mathbb{E}[\mathbb{E}[\mathbf{1}_{ep}\mathbf{1}_{e'p} \mid \mathbf{1}_{ep}]] - l_e l_{e'} u_{e'p} \\
&\because \mathbb{E}[X] = \mathbb{E}[\mathbb{E}[X \mid Y]] \\
&= \mathbb{E}[\mathbf{1}_{ep} \mathbb{E}[\mathbf{1}_{e'p} \mid \mathbf{1}_{ep}]] - l_e l_{e'} u_{e'p} \\
&\because \mathbb{E}[XY \mid X] = X \mathbb{E}[Y \mid X] \\
&= \mathbb{P}(\mathbf{1}_{ep} = 0) \cdot 0 \cdot l_{e'} u_{e'p} \\
&+ \mathbb{P}(\mathbf{1}_{ep} = 1) \cdot 1 \cdot 0 - l_e l_{e'} u_{e'p} \\
&= -l_e l_{e'} u_{e'p}.
\end{aligned} \tag{A11}$$

2 This latter quantity is small because it is proportional to the
3 per base-pair mutation rate squared. Not allowing child muta-
4 tions gives $\mathbb{E}[\mathbf{1}_{e'p} \mid \mathbf{1}_{ep} = 1] = 0$, provided e is an ancestor of
5 e' without the loss of generality. Therefore, we can justify the
6 diagonal covariance matrices of random effects \mathbf{u} by setting all
7 terms involving u_{ep}^2 to zero.

8 In summary, we have the resulting generalization, that allows
9 varying mutational variance along the genome and through time:
10 The distribution of a complex trait follows a linear mixed model
11 conditioned on the sample's ARG $\mathcal{T} = (\mathcal{N}, \mathcal{E})$:

$$\mathbf{y} = \mathbf{u}_0 + \mathbf{Z}\mathbf{u} + \boldsymbol{\varepsilon}, \tag{A12}$$

12 with a constant vector \mathbf{u}_0 (all entries have the same value). The
13 random effects \mathbf{u} 's covariance matrix is:

$$\boldsymbol{\Sigma}_{\mathbf{u}} = \text{diag} \left[l_e s_e \tau_e^2 \right]_{e=1, \dots, E}, \tag{A13}$$

14 where $\tau_e^2 = \frac{1}{s_e} \sum_{p:p \in e} \beta_p^2 u_{ep}$. The residual $\boldsymbol{\varepsilon}$ contains the con-
15 tribution of non-genetic variables. We call the covariance ma-
16 trix $\text{Var}(\mathbf{Z}\mathbf{u}) = \mathbf{Z}\boldsymbol{\Sigma}_{\mathbf{u}}\mathbf{Z}^T$ as the generalized branch GRM. In
17 the model we use in the main paper, we further assume that
18 $\beta_p \sim \text{Normal}(\mu_p, \tau^2)$. Then, $\text{Var}(\mathbf{Z}\mathbf{u})$ further reduces to

$$\text{Var}(\mathbf{Z}\mathbf{u}) = \tau^2 \mathbf{B} \tag{A14}$$

19 in which \mathbf{B} is the branch GRM from [Lehmann et al. \(2025\)](#). μ_p
20 does not affect the GRM and is absorbed to the intercept as in
21 equation (A8) and (A9).

22 Variance of V_G

23 First, note that if X and Y are Gaussian, then Isserlis' Theorem
24 ([Isserlis 1918](#)) says that:

$$\mathbb{E}[X^2 Y^2] = \mathbb{E}[X^2] \mathbb{E}[Y^2] + 2\mathbb{E}[XY]^2.$$

25 Expanding out $\text{Var}(V_G)$, we will use this fact with $X = (\mathbf{P}_N \mathbf{g})_n$
26 and $Y = (\mathbf{P}_N \mathbf{g})_m$. Continuing with the notation from equa-

tion (18):

$$\begin{aligned}
& \text{Var}(V_G) \\
&= \text{Var} \left(\frac{1}{N} \sum_{n=1}^N (\mathbf{g}_n - \bar{\mathbf{g}})^2 \right) \\
&= \frac{1}{N^2} \left(\mathbb{E} \left[(\mathbf{P}_N \mathbf{g})^T (\mathbf{P}_N \mathbf{g}) (\mathbf{P}_N \mathbf{g})^T (\mathbf{P}_N \mathbf{g}) \right] - \mathbb{E} \left[(\mathbf{P}_N \mathbf{g})^T (\mathbf{P}_N \mathbf{g}) \right]^2 \right) \\
&= \frac{1}{N^2} \sum_{n=1}^N \sum_{m=1}^N \left(\mathbb{E} \left[(\mathbf{P}_N \mathbf{g})_n^2 (\mathbf{P}_N \mathbf{g})_m^2 \right] - \mathbb{E} \left[(\mathbf{P}_N \mathbf{g})_n^2 \right] \mathbb{E} \left[(\mathbf{P}_N \mathbf{g})_m^2 \right] \right) \\
&= \frac{1}{N^2} \sum_{n=1}^N \sum_{m=1}^N \mathbb{E} \left[(\mathbf{P}_N \mathbf{g})_n (\mathbf{P}_N \mathbf{g})_m \right]^2 \\
&= \tau^4 \frac{1}{N^2} \sum_{n=1}^N \sum_{m=1}^N \left(\tilde{\mathbf{B}}_{n,m} \right)^2.
\end{aligned} \tag{A15}$$

27 Since the offdiagonals of $\tilde{\mathbf{B}}$ are $\mathcal{O}(1/N)$ (where recall that N is
28 the sample size), the last line tells us that $\text{Var}(V_G) = \mathcal{O}(1/N)$,
29 and hence, $\tau^2 \text{tr}(\tilde{\mathbf{B}})/N$ is a consistent estimator of V_G , with an
30 error that is of order $1/\sqrt{N}$.
31

32 Correctness of Algorithm T

33 Here we present an algorithm for efficient simulation from the
34 ARG-LMM model. This serves several purposes: first, the struc-
35 ture of the algorithm helps to highlight the dependency struc-
36 tures within the ARG. Second, the simplest tree-by-tree trait
37 simulation algorithm for a large ARG could be computationally
38 very costly, and this algorithm is much more efficient. Third, to
39 illustrate how a key part of the matrix-vector product algorithm
40 in [Lehmann et al. \(2025\)](#) works.

41 We begin with a precise explanation of what we'd like to sim-
42 ulate. Consider a given tree T that extends over some span s_T
43 of the genome, and for an edge or sub-edge e and a node n write
44 $e \geq_T n$ if e is ancestral to n in T . Each edge $e \in T$ connects a child
45 $c(e)$ to a parent $p(e)$, and so has area $s_T(t_{p(e)} - t_{c(e)})$ in this tree.
46 For each edge in each tree, let the edge effect $\mathbf{u}_{e,T}$ be a Gaussian
47 random variable with mean zero and variance $s_T(t_{p(e)} - t_{c(e)})$.
48 Then, the genetic value of a node n – the quantity we seek to sim-
49 ulate – is equal to $\sum_T \sum_{e \geq_T n} \mathbf{u}_{e,T}$. However, if the subtree below
50 an edge e does not change across trees T_i, T_{i+1}, \dots, T_j , then these
51 effects can be combined: $\mathbf{u}_{e,T_i} + \dots + \mathbf{u}_{e,T_j}$ is inherited together
52 by the same set of nodes. By additivity of Gaussian distributions,
53 this combined effect is itself Gaussian with variance equal to
54 the combined areas, and so we aim to simulate $\mathbf{Z}\mathbf{u}$, where \mathbf{Z}
55 is the individual-sub-edge design matrix defined above and \mathbf{u}
56 is a vector of sub-edge effects from a Gaussian distribution with
57 mean zero variance equal to sub-edge areas. (We emphasize
58 that in practice a single edge in the ARG can subtend many
59 different subtrees, so that each e denotes a sub-edge as defined
60 above.)

61 The algorithm works, roughly, as follows. The “value” $v(n)$
62 of each node caches a total contribution to genetic value that
63 is inherited by every node in the current subtree below n . The
64 “current value” of node n is obtained by summing up the values
65 of all nodes above n in the current tree – so, if the tree changes,
66 we must make adjustments to maintain a correct current value.
67 When an edge with parent p and child c is removed from the
68 tree, two things happen. First, the effect of that edge is simu-
69 lated, and added to $v(c)$. Then, the value of p and every node
70 above p must be added to $v(c)$, because c is no longer below

those nodes. Adding an edge creates a different problem: after adding the edge, c is below p in the tree, but $v(p)$ contains effects from previous parts of the genome. Therefore, we subtract the values of p and all nodes above p from $v(c)$, so that these effects cancel out when summing up the tree from c . At the end of the algorithm all edges are removed, so we end up with the empty tree, and so $v(n)$ will be equal to the genetic value of node n .

Algorithm T. (*Trait genetic value simulation*). Given a sequence of positions $0 = b_0 < b_1 < \dots < b_K = L$ that are recombination breakpoints along the genome of length L , and corresponding sequences of edges to remove (R_k) and add (A_k) at each position, sample the genetic values g_s for $1 \leq s \leq n_S$, as long as all samples are leaves in all trees. Let T be the current tree, $\ell_T(n) = t_{p(n)} - t_n$ be the length of the edge above node n in T (or zero, if n has no parent), initialize $k = 1$, $x(n) = 0$, and $v(n) = 0$ for all $n \in \mathcal{N}$. At all times, define $z(n) = \sqrt{\ell_{T_k}(n)(b_k - x(n))}$. The k -th tree T_k has span $s_{T_k} = b_k - b_{k-1}$.

T1. [Remove edges] For each edge $(c, p) \in R_k$, and for each node $n \geq_T p$, sample $\mathbf{w}_n \sim \text{Gaussian}(0, 1)$; set $v(n) += z(n)\mathbf{w}_n$; then set $v(c) += v(n)$ and $x(n) = b_k$. Then, set $x(c) = b_k$ and remove the edge.

T2. [Add edges] For each edge $(c, p) \in R_{k'}$, and for each node $n \geq_T p$, sample $\mathbf{w}_n \sim \text{Gaussian}(0, 1)$; set $v(n) += z(n)\mathbf{w}_n$; then set $v(c) -= v(n)$ and $x(n) = b_k$. Then, set $x(c) = b_k$ and add the edge.

T3. [Iteration] If $k < K$, set $k = k + 1$ and return to **T1**. Otherwise, set $g_s = v(s)$ for $1 \leq s \leq n_S$ and finish.

The algorithm proceeds by storing and updating intermediate quantities v , z , and x indexed by nodes. The genetic value is recovered at the end of the algorithm by aggregating these intermediate values efficiently. Updating the intermediate values is also efficient because it only alters the entries that are ancestral to the added or deleted edge in the local tree while leaving all other entries constant.

To prove that the algorithm works, we first reformulate the algorithm so that we have the edge effects beforehand, making the algorithm deterministic. To this end, for each node n in the k th tree, let $\mathbf{z}_{n,k}$ have a Gaussian distribution with mean zero and variance equal to the span of the tree, $b_k - b_{k-1}$. Recall that $x(n)$ is the last position at which node n was visited, and so we can define $k(n)$ so that $b_{k(n)} = x(n)$. Now, define $Z(n) = \sqrt{\ell_{T_k}(n)(\mathbf{z}_{n,k(n)} + \dots + \mathbf{z}_{n,k})}$. Comparing to Algorithm T, $Z(n)$ and $z(n)\mathbf{w}_n$ have the same distribution; the only difference is that we sample from the Gaussian here in several steps. Thus, where steps **T1** and **T2** say “set $v(n) += z(n)\mathbf{w}_n$ ”, instead replace these with the equivalent instruction “set $v(n) += Z(n)$ ”.

To show that the algorithm calculates the genetic value, we first show how to obtain the genetic value for each node based on the genome up until the current position. Then, we show that steps **T1** and **T2** in Algorithm T do not change the genetic value.

First, define the “working genetic value”, which is the genetic value of node n calculated only the genome up until position b_k , as

$$g_k(n) = \sum_{h=1}^k \sum_{u \geq_T n} \sqrt{\ell_{T_h}(u)} \mathbf{z}_{u,h}.$$

This has the correct distribution because $\sqrt{\ell_{T_h}(u)} \mathbf{z}_{u,h}$ has a Gaussian distribution with mean zero and variance equal to

$(b_h - b_{h-1})\ell_{T_h}(u)$, which is the area of the edge above u in tree h . By additivity of Gaussians, the contribution of a given edge that spans multiple trees is again Gaussian has mean zero and variance equal to its area.

Also, define

$$\hat{g}(n) = \sum_{r:r \geq_T n} (v(r) + Z(r)). \quad (\text{A16})$$

We claim that at any point in the algorithm,

$$g_k(n) = \hat{g}(n), \quad \text{for all } n. \quad (\text{A17})$$

At the end of the algorithm, the value returned is $\hat{g}(s)$, because tree is empty at termination, so the right-hand side reduces to $v(s)$. $z(s)$ is zero because $x(s) = b_K$ (recall that $z(n) = \sqrt{\ell_{T_k}(n)(b_k - x(n))}$). Therefore, if equality (A17) holds throughout the algorithm, then $v(s)$ is the desired genetic value, $g_K(s)$.

We prove equation (A17) by induction. The equality trivially holds at the beginning of the algorithm, because both sides are zero. Next we prove that each of the three steps in the algorithm preserves the equality (A17).

Step T3 is the easiest to analyze. By the induction step, we assume that we began this step with $\hat{g}(n) = g_k(n)$. The current tree T and the intermediate quantities v and x do not change. Advancing the position from k to $k' = k + 1$ changes $Z(n)$ by adding $\sqrt{\ell_{T_k}(n)} \mathbf{z}_{n,k+1}$, and so changes $\hat{g}(n)$ by

$$\sum_{r:r \geq_T n} \sqrt{\ell_{T_k}(r)} \mathbf{z}_{r,k+1}.$$

Comparing to equation (A16), this is equal to $g_{k+1}(n) - g_k(n)$, and so at the end of this step, $\hat{g}(n) = g_{k'}(n)$, as desired.

This is the only step where the working genetic value changes. In the other two steps, v , z , and x change, but $\hat{g}(n)$ remains constant, thus preserving equation (A17). We prove this only for **T2**; the argument is basically the same for **T1**.

Suppose that we added edge (c, p) where c and p are child and parent nodes of the edge, respectively. Call the tree at the start of the step T , and the tree after the step T' , and similarly for x' , v' , Z' , and \hat{g}' . Since adding an edge only increases the set of nodes above n , the difference in $\hat{g}(n)$ caused by **T2** is

$$\begin{aligned} \hat{g}'(n) - \hat{g}(n) &= \sum_{r:n \leq_T' r} (v'(r) + Z'(r)) - \sum_{r:n \leq_T r} (v(r) + Z(r)) \\ &= \sum_{r:n \leq_T' r, n \not\leq_T r} (v'(r) + Z'(r)) \\ &\quad + \sum_{r:n \leq_T r} (v'(r) + Z'(r) - v(r) - Z(r)). \end{aligned} \quad (\text{A18})$$

We can divide the nodes of \mathcal{N} into four categories as (1) the child node c , (2) nodes below c , (3) the parent node p and nodes above p , and (4) all other nodes. Since $v(r)$ and $Z(r)$ only change for nodes that are above c in T' , evidently $\hat{g}'(n) = \hat{g}(n)$ for nodes in the last category. Next, note that $v(n) + Z(n) = v'(n) + Z'(n)$ for nodes n that are above p : this is because **T2** first sets $v'(n) = v(n) + Z(n)$, then sets $x(n) = b_k$, which has the effect that $Z'(n) = 0$. Therefore, $\hat{g}'(n) = \hat{g}(n)$ for nodes $n \geq_T p$.

This leaves only nodes $n \leq_T' c$. The first term in equation (A18) sums over $\{r : n \leq_T' r, n \not\leq_T r\}$: nodes that are above n after adding the edge, but were not above n before the

1 addition. Hence, for nodes below c , this set is simply all nodes
2 above p , and the first summation is

$$\sum_{r:p \leq_T r} (v'(r) + Z'(r)). \quad (\text{A19})$$

3 The only nodes that change x are those above p , so for nodes
4 $r \leq_T c$, $Z'(r) = Z(r)$. Furthermore, $v'(r) = v(r)$ for nodes
5 strictly below c . Thus, the summand of the second summation
6 is $v'(r) + Z'(r) - v(r) - Z(r)$ is equal to 0, except for $r = c$, in
7 which case it is $v'(c) - v(c)$. Therefore, the second summation
8 is just $v'(c) - v(c)$. Now, $v(c)$ changes because this step, for
9 each $r \geq_T p$, first sets $v'(r) = v(r) + Z(r)$, and then subtracts
10 $v'(r)$ from $v(c)$. As a result, still for nodes n below c , the second
11 summation is

$$v'(c) - v(c) = - \sum_{r:p \leq_T r} (v(r) + Z(r)). \quad (\text{A20})$$

12 Putting these together, the two summations in equation (A18)
13 cancel, so $\hat{g}'(n) = \hat{g}(n)$ for all remaining nodes.

14 In summary, **T1** and **T2** modify the intermediate values v , x ,
15 and Z while leaving the working genetic value constant so that
16 **T3** correctly updates the genetic value. See **Efficient ARG-LMM**
17 **simulation** for more details of fully simulating the trait based
18 on this algorithm.

19 Submatrix computation

20 `tskit` provides multiplication by \mathbf{B} (and \mathbf{B}^{-1} via conjugate gra-
21 dient), and for its submatrices, one can multiply an incidence
22 matrix before and after multiplying \mathbf{B} to perform matrix-vector
23 multiplication. For instance, $\mathbf{B}_{n,o} \mathbf{v}$ can be done by:

$$\mathbf{B}_{n,o} \mathbf{v} = \mathbf{A}_n \cdot \mathbf{B} \cdot \mathbf{A}_o^T \mathbf{v}, \quad (\text{A21})$$

24 where \mathbf{A}_n and \mathbf{A}_o are incidence matrices with each row corre-
25 sponding to the non-phenotyped and observed individuals. For
26 each row, only one column that corresponds to the index of
27 the row in the full list of individuals are set to 1 and 0 other-
28 wise. Multiplication by incidence matrices \mathbf{A}_\bullet can be done very
29 quickly as they are very sparse.

30 The conditional variances of the predicted genetic values
31 may be useful as a measure of precision for individual BLUPs,
32 but in large-scale applications assembling the full covariance
33 matrix (equation 20) is infeasible. However, we can obtain un-
34 biased estimates of the diagonal using the XDiag algorithm in
35 [Epperly et al. \(2024\)](#), which only requires matrix-vector products
36 of equation (20) against a fixed number of random test vectors.