nature genetics

Article

https://doi.org/10.1038/s41588-023-01487-8

Extremely sparse models of linkage disequilibrium in ancestrally diverse association studies

In the format provided by the authors and unedited

nature genetics

Article

https://doi.org/10.1038/s41588-023-01487-8

Extremely sparse models of linkage disequilibrium in ancestrally diverse association studies

In the format provided by the authors and unedited

Supplementary Note for "Extremely sparse models of linkage disequilibrium in ancestrally diverse association studies"

1. Probabilistic Graphical Model of Genotypes

In this section, we precisely define the conditional dependency and independency of SNPs. Then we define two classes of graphs that can represent conditional dependencies and independencies. We will produce a graphical model that represents all conditional dependencies that is minimal under some theoretical assumptions.

Let $S = \{1, 2, \ldots, m\}$ denote a set of SNPs and let $H \subseteq 2^S$ denote a set of haplotypes. We analyze the conditional dependencies among SNPs that arise when sampling one of these haplotypes at random. Initially, we assume that haplotypes in H are observed completely, without missing SNPs; in practice, ancestral haplotypes are only partially observed, and we consider this case at the end of this section. For a subset of SNPs A and a haplotype $h \in H$ the restriction of h to A is denoted by $A(h)$.

Let Π be the set of probability distributions with support H, let $P \in \Pi$, and let X be a random variable with distribution P. A subset of SNPs A is identified with the random variable $X_A = A(X)$ chosen under P, to A.

For three sets of SNPs A, B, C, X_A is conditionally independent of X_B given X_C if their joint probability density factors:

$\forall c\ s.t.\ P(X_C = c) > 0,\ P(X_A = a, X_B = b\,|X_c = c) = P(X_A = a|X_C = c)\ P(X_B = b|X_C = c)$.

Depending on H, certain conditional independencies are possible, and others are not:

Proposition 1. For three sets of SNPs A, B, C, X_A and X_B are conditionally dependent given X_C for some distribution $P \in \Pi$ if and only if for some pair of haplotypes $h_1$and h_2, the symmetric difference $h_1 \Delta\ h_2$ overlaps A and B but not C.

Proof: First, suppose there is no such pair of haplotypes; for any h_1, h_2, without loss of generality,

$$P(A = A(h_1)|\ C = C(h_1))\ P(B = B(h_1)|\ C = C(h_1)) = P(B = B(h_1)|\ C = C(h_1))$$
$$= P(A = A(h_1), B = B(h_1)|\ C = C(h_1))$$

Therefore A and B are conditionally independent given C.

Second, suppose that for some $h_1, h_2 \in H$, $h_1 \Delta\ h_2$ overlaps A and B but not C. Let $c = C(h_1)$, and let $P(h_1) = \frac{1}{2}$ and $P(h_2) = \frac{1}{2}$. Then:

$$P(X_A = A(h_1), X_B = B(h_1)|X_c = c) = \frac{1}{2} \neq \frac{1}{4}$$
$$= P(X_A = A(h_1)|X_C = c)\ P(X_B = B(h_1)|X_C = c)$$

■

A simple graph $G = (S, E)$ is an *independency map* (or *I-map*) of $H \subset \{0, 1\}^{|S|}$ if for all disjoint subsets A, B, C of S we have

$$A \perp_{\forall P \in \Pi} B|C \Leftarrow A \perp_G B|C$$

Similarly, G is a *dependency map (or D-map)* of Π if

$$A \perp_{\forall P \in \Pi} B|C \Rightarrow A \perp_G B|C$$

G is said to be a *perfect map* of Π if it is both a D-map and an I-map. These three types of graph were defined for a particular probability distribution in ref [1], and here the definitions are modified to include all dependencies that arise in any $P \in \Pi$.

Proposition 2: G is an I-map for S and H if and only if for all $h_1, h_2 \in H$, $h_1 \Delta\ h_2$ is a connected set in G.

Proof: Suppose that for all $h_1, h_2 \in H$, $h_1 \Delta\ h_2$ is a connected set in G. If X_A, X_B are conditionally dependent given X_C for some $P \in \Pi$, then by Proposition 1, there exist h_1, h_2 such that $h_1 \Delta\ h_2$ contains some $a \in A, b \in B$ but not any $c \in C$. $h_1 \Delta\ h_2$ is connected, so there is a path in G from $a \in A$ to $b \in B$ contained within $h_1 \Delta\ h_2$; this path does not pass through C, so G is an I-map.

Suppose not: for some $h_1, h_2 \in H$, $h_1 \Delta\ h_2$ is not a connected set in G. For some $a, b \in h_1 \Delta\ h_2$, every

path from a to b in G passes through a set C with $h_1 \cap C = h_2 \cap C$. By Proposition 1, there exists $P \in \Pi$ such that a, b are conditionally dependent given C, and G is not an I-map.

■

Not all sets of haplotypes have a perfect map. For example, suppose $S = \{a, b, c\}$ and $H = \{ABC,\ aBC,\ abc\}$. Suppose G is an I-map. By Proposition 2, $bc \in E(G)$ and $\{a, b, c\}$ must be a connected set in G. Since G is an I-map, either ab or $ac \in E(G)$ and hence G is not a D-map.

Problem 1: Which sets of haplotypes H have a perfect map?

In special cases, a perfect map exists. A set of haplotypes H is called *recombination-free* if there exists a rooted tree $T = (H, E)$ such that the root has no derived alleles and an injection $M: S \rightarrow E$ such that for all SNPs $m \in S$, the haplotypes with the derived allele of SNP m are exactly those that come after $M(m)$ with respect to T. Some edges of T have labels associated with mutations i.e., elements of $M(H)$. Define the graph $G(S, E)$ such that two SNPs are adjacent if and only if the unique path connecting them contains no labeled edge. In other words, G is the reduced graph of the line graph of T.

Theorem 1: Suppose H is recombination-free. Then G, the reduced line graph on vertices corresponding to labeled edges of T, is the perfect map.

Proof: First we show that G is an I-map. Suppose h is the most recent common ancestor of $h_1, h_2 \in H$ (possibly, $h = h_1$). In T there are unique paths P_1 and P_2 from h to h_1 and h_2 respectively. Let S_i be the set of labeled edges of P_i for $i = 1, 2$. Then $h_1 \Delta\ h_2 = S_1 \cup S_2$, because h_1 has the derived allele for SNPs S_1 and h_2 has the derived allele for SNPs S_2. By the construction of G, $S_1 \cup S_2$ is connected; therefore, by Proposition 2, G is a I-map.

Now we show that G is a D-map. Suppose $a, b \in S$ and $C \subset S$. Suppose the unique path in T between a and b does not intersect C i.e., a and b belong to the same connected component of $G \setminus C$. We will show that a and b are conditionally dependent given C. There are two cases, a and b are either on the same branch with respect to the root of T or not.

Case 1. Suppose a and b are on the same branch of T. Let h_1 be the haplotype before a and h_2 be the haplotype after b. Then $h_1 \Delta h_2$ has both a and b but not any $c \in C$ since any $c \in C$ either belong to both or neither of h_1 and h_2 depending on whether h_1 has c or not, and by Proposition 1, a and b are conditionally dependent given C.

Case 2. Suppose a and b are on different branches of T. Let h be the MRCA haplotype of a and b. Let h_1 and h_2 be direct descendant haplotypes of a and b, respectively. Then $h_1 \Delta h_2$ has both a and b but not any $c \in C$ since any $c \in C$ either belong to both or neither of h_1 and h_2 depending on whether h has c or not, and by Proposition 1, a and b are conditionally dependent given C.

■

Problem 2: For a given set of Haplotypes H with SNPs set S find a sparse simple graph $G = (S, E)$ that is an I-map.

Using Proposition 2, a sufficient condition for G is that for all pairs of haplotypes $h_1, h_2 \in H$, $h_1 \Delta\ h_2$ forms a connected set. In the next section, we define a simple graph called the LDGM and prove that the symmetric difference of any two haplotypes is a connected set of LDGM. Additionally, the LDGM is edge-minimal under a precise condition.

Missing SNPs. In practice, ancestral haplotypes are only partially observed. We generalize the definition of conditional dependency and prove Proposition 1 in the generalized setting.

Let the set of haplotypes be $H \subseteq \{0,1,-1\}^m$ where for $h \in H$, $h_i = -1$ if h is missing at position i.

First, we generalize the symmetric difference to this setting. For $h_1, h_2 \in H$ let their symmetric difference comprise the SNPs where they are different and non-missing:

$$h_1 \Delta\ h_2 = \{s \in S: \text{either } s(h_1) = 1, s(h_2) = 0 \text{ or } s(h_2) = 0, s(h_1) = 1\}.$$

We write that $h_1 \equiv h_2$ when $h_1 \Delta\ h_2 = \emptyset$, i.e. when h_1, h_2 are identical at every position where they are both non-missing.

Next, let Π be a set of probability distributions with support H, let $P \in \Pi$, and for a subset of SNPs A, let X_A be a random variable with distribution P restricted to SNPs in A. Recall that for a haplotype $h \in H$, by $A(h)$ we mean haplotype h restricted to A. For a second set of SNPs B, let

$$H_B = \{h \in H : -1 \notin B(h)\}$$

and let

$$H_B(A, a) = \{h \in H_B : A(h) \equiv a\}.$$

We define:

$$P_B(X_A \equiv a) = \frac{\sum_{h \in H_B(A,a)} P(h)}{\sum_{h \in H_B} P(h)}$$

and likewise for e.g., $P_B(X_A \equiv a \mid X_C \equiv c)$.

This notation allows us to define conditional dependency between two sets of SNPs in the presence of missingness, by effectively conditioning on non-missingness. For three disjoint sets of SNPs A, B, C, we say that A and B are conditionally dependent given C if for some $c = C(h)$, there exist subsets $A' \subset A$ and $B' \subset B$ with $a = A'(h)$ and $b = B'(h)$ such that

$$P_{A' \cup B'}(X_{A'} \equiv a, X_{B'} \equiv b \mid X_C \equiv c) \neq P_{A' \cup B'}(X_{A'} \equiv a | X_C \equiv c)\, P_{A' \cup B'}(X_{B'} \equiv b | X_C \equiv c).$$

The reason A' and B' are needed, instead of A and B, is explained by this example. Let $h_1(A) = (0)$, $h_1(B) = (0,0)$, $h_2(A) = (1)$, and $h_2(B) = (1, -1)$. With our definition, A and B can be conditionally dependent with B' comprising the first element of B; however, $h_2 \notin H_{A \cup B}$, so h_2 would not produce a conditional dependency if dependencies were defined in terms of $P_{A \cup B}$.

We generalize Proposition 1:

Proposition 3 (generalizing Proposition 1): For three sets of SNPs A, B, C, X_A and X_B are conditionally dependent given X_C for some distribution $P \in \Pi$ if and only if for some pair of haplotypes h_1 and h_2, the symmetric difference $h_1 \Delta h_2$ overlaps A and B but not C.

Proof: First, suppose that for all possible values c of $C(h)$, for all $h_1, h_2 \in H_\emptyset(C, c)$, $h_1 \Delta h_2$ is disjoint from either A or B. For any such c, without loss of generality suppose $h_1 \Delta h_2$ is disjoint from A; let $A' \subset A$ and $B' \subset B$, and let a' be a restricted to A'. Then

$$P_{A' \cup B'}(X_{A'} \equiv a' \mid X_C \equiv c) = 1$$

because $A(h) \equiv a$ implies $A'(h) \equiv a'$. It follows that

$$P_{A' \cup B'}(X_{A'} \equiv a | X_C \equiv c)\, P_{A' \cup B'}(X_{B'} \equiv b | X_C \equiv c) = P_{A' \cup B'}(X_{B'} \equiv b | X_C \equiv c)$$
$$= P_{A' \cup B'}(X_{A'} \equiv a, X_{B'} \equiv b | X_C \equiv c)$$

and A and B are conditionally independent given C.

Second, suppose that for some $h_1, h_2 \in H$, $h_1 \Delta h_2$ overlaps A and B but not C. Let $c = C(h_1)$, let $A' = A \cap (h_1 \Delta h_1) \neq \emptyset$ and $B' = B \cap (h_1 \Delta h_1) \neq \emptyset$. Let $P(h_1) = \frac{1}{2}$ and $P(h_2) = \frac{1}{2}$. Then:

$$P_{A' \cup B'}(X_{A'} \equiv A'(h_1), X_{B'} \equiv B'(h_1) | X_C \equiv c) = \frac{1}{2} \neq \frac{1}{4}$$
$$= P_{A' \cup B'}(X_{A'} \equiv A'(h_1) | X_C \equiv c)\, P_{A' \cup B'}(X_{B'} \equiv B'(h_1) | X_C \equiv c).$$

■

2. LD Graphical Models

In this section, we generalize the approach in Theorem 1 to the case of recombination. We define a tree sequence, modify the tree sequence so that the graph reduction of Theorem 1 can be applied, and we show that the resulting graph is an I-map (Theorem 2.1) satisfying a minimality condition (Theorem 2.2). We illustrate each step of the algorithm with an example (Example 1).

2.1. Tree Sequences

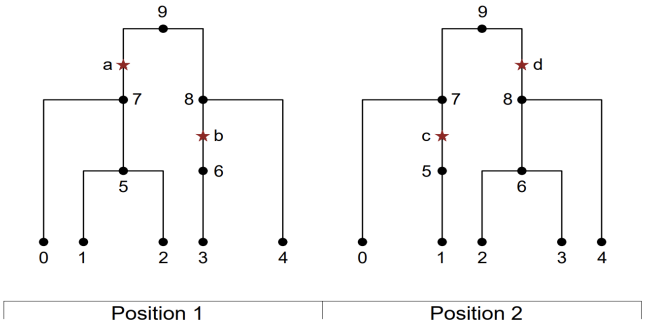
We provide a formal definition of a *tree sequence*[2]:

Definition 2.1. A tree sequence is a directed acyclic multigraph T whose vertices are a set of ancestral and sample haplotypes H, with the following properties:

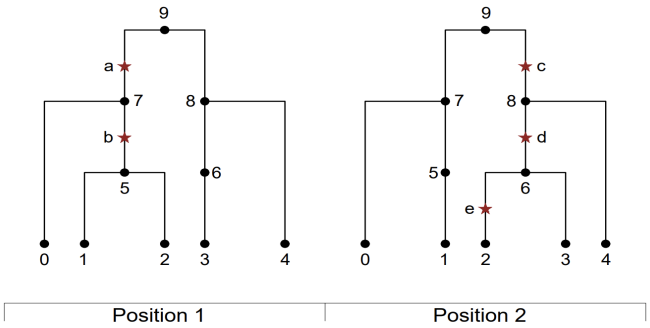
- Each edge of T is a tuple (h_1, h_2, S), where $h_1$is the "parent haplotype", h_2 is the "child haplotype", and S is the set of positions that are spanned by that edge
- For every position n, the set of haplotypes that exist at that position and the set of edges (h_1, h_2, S) with $n \in S$ form a rooted tree.
- Every mutation is identified with a position n and a clade of the rooted tree at position n; the haplotypes that have that mutation are precisely those contained in that clade, and haplotypes that do not have mutation are those that exist at that position but are not in the clade.

SNPs are encoded in the tree sequence as mutations on specific edges. Ordinarily, the carriers of a SNP may depend not only on which edge this is but also on its position within that edge.

Example 1.A. (Example from Figure 1) The tree sequence below has 5 sample haplotypes and 5 ancestral haplotypes. The tree sequence has two positions, each with two mutations. Haplotypes 5 and 6 recombined in haplotype 2.



Example 2. (Tree sequence with no perfect map) The tree sequence below has the same genealogy as Example 1.A, but different mutations. Suppose for contradiction that G is a perfect map. Haplotype 5 has mutations $\{a, b\}$ and haplotype 2 has mutations $\{a, b, c, d, e\}$. Therefore their symmetric difference is $\{c, d, e\}$ that means $\{c, d, e\}$ must form a connected set. The symmetric difference of haplotypes 6 and 9 implies that $\{c, d\}$ is connected, i.e., $cd \in E(G)$. Since the only haplotypes whose symmetric difference has both c and e are 9 and 2, c and e are independent given d which implies that $ce \notin E(G)$. Therefore de must be an edge and hence d is a cut vertex splitting c and e. On the other hand, any symmetric difference containing d abd e also intersects $\{a, b, c\}$ but $\{a, b, c\}$ is not a cut set splitting d and e which is a contradiction.



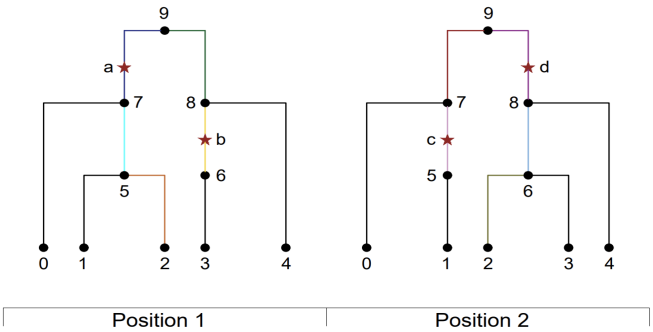
2.2. Bricked tree sequences

In an ordinary tree sequence, if a haplotype has different descendants at each position due to recombination, then a mutation occurring on a single edge would have different carriers depending on

its position. This property motivates us to modify the tree sequence, bifurcating edges when they have different descendants at different positions (see section 3.1). We refer to the resulting edges as *bricks*; bricks have the same descendant haplotypes at each position. The resulting graph, which can have multiple edges between the same two haplotypes, is referred to as a *bricked tree sequence*.

Definition 2.2 A *bricked tree sequence* is a tree sequence with the property that for every edge ("brick") (h_1, h_2, X), the descendent haplotypes of h_2 are the same at every position $x \in X$.

Example 1.B. (Example from Figure 1a) This example shows the bricked tree sequence. Note that each brick must have a unique set of descendent haplotypes. For example, the edge between haplotypes 8 and 9 has $\{0, 1, 2\}$ and $\{0, 1\}$ descendent haplotype sets in the left and right tree respectively. Therefore, this edge must create two bricks. In the following brick diagram, the corresponding two bricks are represented by assigning two colors (green and purple).



2.3 Brick diagrams and the LDGM

In this section, we introduce *brick diagrams,* which describe the symmetric difference between two haplotypes in a tree sequence. We use brick diagrams to define the LDGM, and we show that the LDGM is correct and minimal.

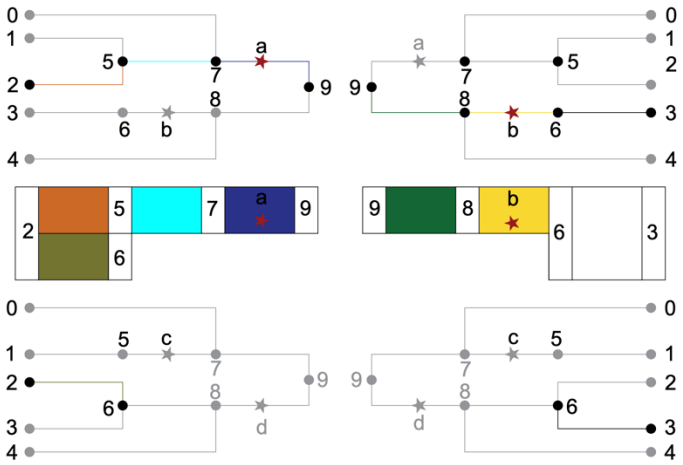
2.3.1 Brick diagrams

In the no-recombination case, the symmetric difference between two haplotypes was the SNPs along the unique path between them in the single tree. A brick diagram is a generalization of a path between two haplotypes.

Definition 2.3. The brick diagram of two haplotypes is a subset of the edges of the bricked tree sequence. A brick is in the brick diagram of haplotypes h_1, h_2 if it is an ancestor of exactly one of the two haplotypes.

A brick diagram connects a pair of haplotypes via their most recent common ancestors (MRCAs), and it can be visualized as follows.

Example 1.C. (Example from Figure 1b) The following is the brick diagram representing the symmetric difference of haplotypes 2 and 3 in Example 1. The symmetric difference between these haplotypes is $\{a, b\}$, and these are the mutations that appear on the brick diagram.

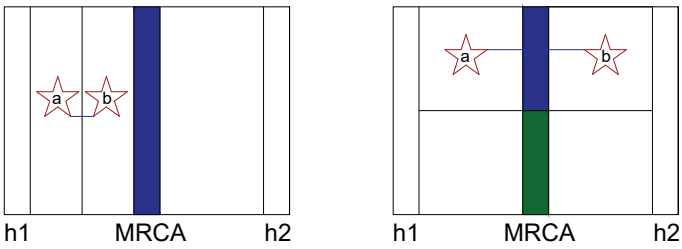


2.3.2 Paths between mutations and definition of the LDGM

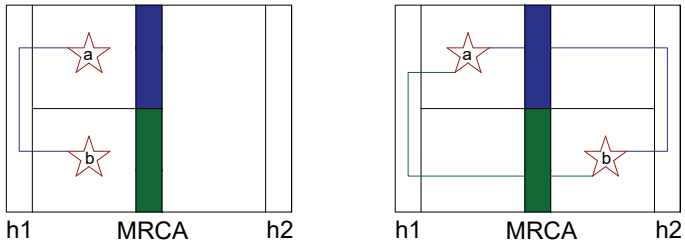
The mutations occurring on the bricks of the brick diagram are the symmetric difference between the haplotypes, since bricks have the same descendants at every position (unlike edges of a tree sequence in general). To find the LDGM that satisfies the first condition of Problem 2, we need a set of rules guaranteeing that the labeled bricks of every brick diagram form a connected set in the LDGM.

Consider two labeled bricks a and b in a brick diagram between two haplotypes h_1 and h_2. Our goal is to create a graph with a path between a and b. There are two cases:

First, suppose that $a, b \in h_1$. Both mutations occur on one side of the brick diagram (with respect to the MRCA). There are two subcases: a parent-child relationship if a and b occur on bricks with a shared position, or a recombination (parent-child-parent) relationship if not. In each subcase, we make paths between the bricks as follows:



Second, suppose that $a \in h_1, b \in h_2$. The mutations occur on opposite sides of the brick diagram, and we create paths between them that cross through an MRCA. If a and b have overlapping positions, then their MRCA is an MRCA of the haplotypes h_1 and h_2, and this is a child-parent-child relationship. If not, the path must pass through an MRCA and a child haplotype, and this is a child-parent-child-parent relationship.



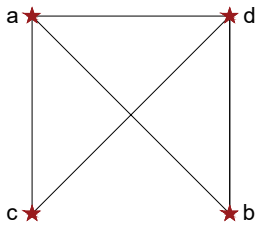
In summary, there are four types of paths in a brick diagram:
1. “I-shaped paths” between parent and child.
2. “V-shaped paths” between two parents of a shared child.
3. “A-shaped paths” between two children of a shared parent.
4. “N-shaped paths” formed by joining an A-shaped and a V-shaped path.

Similar to the recombination-free case (Theorem 1), if any path connecting SNPs a and b contains a third SNP c, then a and b are independent given c. Therefore, a path between two SNPs (labeled brick) is called *blocked* if it contains another labeled brick. We want the LDGM to have an edge between any two labeled bricks connected with an unblocked path.

Definition: The LDGM associated with a bricked tree sequence T is the graph whose vertices are the set of SNPs S and whose edges are as follows:

- If a, b are mutations whose bricks have some shared position, then they have an edge in the LDGM if they have an unblocked “I” or “A” shaped path in every tree where they both exist.
- If a, b are mutations whose bricks have no shared position, then they have an edge in the LDGM if for some haplotype h in the tree sequence, each of them has an unblocked “V” or “N” shaped path to h at every position where they exist.

Example 1.D. (Example from Figure 1c) The following graph is the LDGM of the tree sequence in Example 1.A. For instance, ab is created from an A-shaped path shown in example 1.C.



2.3.3. Correctness of the LDGM

In this subsection we will show that the LDGM of a tree sequence T is an I-Map for its set of haplotypes H.

Theorem 2.1. The LDGM is an I-map.

Proof: Suppose $h_1, h_2 \in H$ are haplotypes with $a, b \in h_1 \Delta\, h_2$. We need to show that the LDGM has a path between a and b that stays in $h_1 \Delta\, h_2$. By the definition of the brick diagram every mutation in $h_1 \Delta\, h_2$ is contained in the brick diagram D associated with $h_1,\ h_2$. The mutations a, b may have one of our four types of paths. If the path between them is unblocked, then there is an edge between a, b in the LDGM. If a third mutation c blocks the path between a and b, it is enough to show that a does not block the path between c and b, and b does not block the path between a and c (because there are finitely many mutations). Consider the four types of paths:

Case 1. There is an “I” shaped path between an ancestor a and a descendant b. If c blocks this path, then c is the descendant of a and the ancestor of b, and there are “I” shaped paths between c and both a and b.

Case 2. There is an “A” shaped path between a and b. If c blocks this path, then c is an ancestor of either a or b, and hence a does not block the path (“A” shaped path or “I” shaped path respectively) between c and b.

Case 3. There is a “V” shaped path between a and b. Therefore a and b recombined in a haplotype h. If c blocks this path, then c is a descendant of a or b and an ancestor of h. Therefore, a does not block the path between c and b.

Case 4. There is an "N" shaped path between a and b. If c blocks this path, then based on the position of c on the path between a and b this case is equivalent to either Case 2 or Case 3.

■

2.3.3. Minimality of the LDGM

In this subsection, we will show that the LDGM is minimal in the following sense. For a bricked tree sequence T with mutations S, we say that a set of haplotypes $X \subset 2^S$ satisfies the *four gametes condition* with respect to T if X contains the empty haplotype h_0 and if for all pairs of SNPs a, b:

- X contains no haplotype with ab if and only if a and b are siblings in T.
- X contains no haplotype with aB if and only if a is a parent of b.
- X contains no haplotype with Ab if and only if a is a child brick of b.

Theorem 2.2. Suppose X satisfies the four gametes condition with respect to a bricked tree sequence T, and let G be the LDGM associated with T. For every edge edge (a, b) of G, for every third SNP c, there exist h_1, h_2 in X such that $h_1 \Delta\ h_2$ contains *a* and *b* but not c.

Proof: Suppose a and b are adjacent SNPs in the LDGM. There are two cases and several subcases:

Case 1. a and b have a shared position. Then they either have a parent-child relationship or, if not, a sibling-sibling relationship.

Case 1.1. If a and b have an unblocked "I" shaped path, with a being the ancestor of b, consider the child haplotype h_1 of b. h_1 has both a and b; suppose it also has a third mutation c. Because c does not block the path from a to b, either c is an ancestor of a, or c is not an ancestor of b. If it is an ancestor of a, then there exists a haplotype in $h_2 \in X$ with c but not a, b, and $h_1 \Delta h_2$ contains a, b but not c. Otherwise, if it is not an ancestor of b, then some $h_2 \in X$ has derived allele b but not c. h_2 must also have a; the symmetric difference between h_2 and the empty haplotype h_0 contains a and b but not c.

Case 1.2. If a and b have an unblocked "A" shaped path, consider the child haplotype h_1 of a and the child haplotype h_2 of b. Suppose that h_1 but not h_2 has mutation c. There are two possibilities, depending on whether c is an ancestor of a. Suppose c is not an ancestor of a; similar to the parent-child case, X contains a descendent haplotype h_3 of a that does not have c. The symmetric difference between h_3 and h_2 contains a and b but not c. Now suppose c is not an ancestor of a. Because it does not block the path from a to any of the MRCAs of a and b, it must be an ancestor of one of those MRCAs. This does not mean that it is an ancestor of b, which may not be a descendent of the common ancestor at that particular position (bricks have the same descendant haplotypes at every position, but not the same descendant bricks); in fact, it must not be an ancestor of b, since h_2 does not have c. Therefore, the positions of c and b must not overlap, and in particular, c and b are not siblings. By the four gametes condition, some haplotype $h_3 \in X$ has both b and c. This haplotype cannot have a, since a and b are siblings; therefore, the symmetric difference between $h_1 \supset (aBc)$ and $h_3 \supset (AbC)$ contains a and b but not c.

Case 2. Suppose that a and b do not have a shared position. Again, there are two possible relationships that might produce the edge between them: either a and b have a shared descendent h, or a has a shared descendent with a sibling of b.

Case 2.1. Suppose that a and b have an unblocked "V" shaped path. Suppose that h also has some mutation c (otherwise, its symmetric difference with the empty haplotype contains a and b but not c).

Case 2.1.1. If c is at a different position from both a and b, then it is not their ancestor, and by the four gametes condition, there is some haplotype $h_1 \in X$ that has a but not c, and there is some haplotype $h_2 \in X$ that has b but not c. If $h_1 = h_2$ then the symmetric difference between h_1 and the empty haplotype contains a and b but not c. If $h_1 \neq h_2$, then the symmetric difference between $h_1 \supset (aBC)$ and $h_2 \supset (AbC)$ contains a and b but not c.

Case 2.1.2. Suppose that c has a shared position with a. Then c is either an ancestor, a descendent, or a sibling of a. We claim that c is an ancestor of a. First, c cannot be a sibling of a, because sibling bricks never have common descendants. Second c cannot be a descendent of a, as any descendent brick of a

that is an ancestor of h would block the path between a and h. Because c is an ancestor of a, b is not an ancestor of c, as b is not an ancestor of a. By the four gametes condition, there exists some haplotype $h_1 \in X$ that has c but not b. If h_1 does not have a, then the difference between h (abc) and h_1 (ABc) contains a, b but not c. If h_1 does have a, then let h_2 be a haplotype that has c but not a. If h_2 has b, then the symmetric difference between $h_2 \supset (Abc)$ and $h_1 \supset (aBc)$ contains a and b but not c. If h_2 does not have b, then the symmetric difference of $h_2 \supset (ABc)$ and $h \supset (abc)$ contains a and b but not c.

Case 2.2. Suppose that a and b have an unblocked "N" shaped path, and in particular that b has a shared descendent h_2 with some sibling of a. Let h_1 be the child haplotype of a; h_1 has a but not b. First, suppose that h_1 but not h_2 has a third mutation c. Because c does not block the path from a to b, and because it is not an ancestor of b, it must not have a shared position with b. Therefore, it is not a sibling of b, and some haplotype $h_3 \in X$ has both b and c. If h_3 does not have a, then the symmetric difference between $h_3 \supset (Abc)$ and $h_1 \supset (aBc)$ contains a and b but not c. If h_3 does have a, then let $h_4 \in X$ be a haplotype having c but not a (which exists because c is not a descendent of a). If h_4 does not have b, then the symmetric difference between $h_3 \supset (abc)$ and $h_4 \supset (ABc)$ contains a and b but not c. If h_4 does have $b \supset (Abc)$, then its symmetric difference with $h_1 \supset (aBc)$ contains a and b but not c.

■

3. Algorithm for LDGM

To build the LDGM, first, we construct the bricked tree sequence. Second, we create two intermediate graphs called the brick-haplotype graph and the SNP-haplotype graph. Third, we remove the haplotypes and construct the LDGM.

3.1. Construction of the bricked tree sequence

Before using a tree sequence to construct an LDGM, we must first modify the tree sequence to create a *bricked tree sequence* using the efficient operations implemented in *tskit*[2–4].

To convert a tree sequence into a bricked tree sequence, we bifurcate an edge to create two bricks whenever recombination changes the edge's descendants between adjacent marginal trees. The tree sequence data structure allows this to be efficiently computed, since adjacent marginal trees in a tree sequence differ by one or more subtree-prune-and-regraft (SPR) operations. Any edges in the subtree which is regrafted in an SPR operation will retain the same descendants at both marginal trees and thus will not need to be bifurcated. Any edges *above* the subtree being regrafted may have different descendants, until the recombination is resolved at the first shared ancestral edge between the origin (at the left marginal tree) and destination (at the right marginal tree) of the subtree being regrafted. Any edges older than the MRCA will have the same descendants in the adjacent marginal trees and thus will not need to be bifurcated.

This algorithm is efficiently implemented using the *tskit.edge_diff()* function, which returns the edges that differ between marginal trees in a tree sequence. The software package *ldgm* efficiently implements this operation using the *tskit* tree sequence development toolkit.

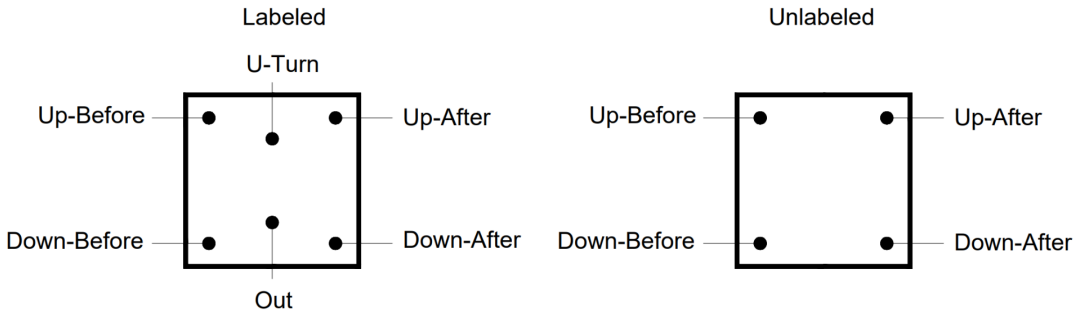
3.2. Construction of the LDGM:

Recall the definition of the LDGM: the graph with SNPs set as the set of vertices and edges as follows.

- If a, b are mutations with some shared position, then they have an edge in the LDGM if they have an unblocked path in every tree where they both exist, forming an "I" or "A" shaped path.
- If a, b are mutations with no shared position, then they have an edge in the LDGM if for some haplotype h, each of them has an unblocked path to h at every position where each brick exists, forming a "V" or "N" shaped path.

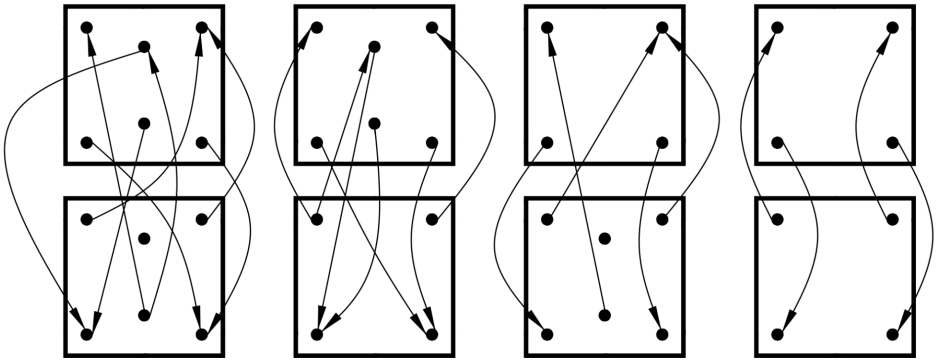
To construct the LDGM from the bricked tree sequence, we start by identifying “I” shaped paths between labeled bricks. We also identify “I” shaped paths between bricks and haplotypes, where recombination might occur. By the way the paths are constructed, we identify which paths are blocked and which ones are unblocked. Second, we identify “A” shaped paths, between pairs of labeled bricks and from labeled bricks to haplotypes, distinguishing blocked and unblocked paths. Third, we identify unblocked “V” shaped paths, which are unions of two unblocked “I” shaped paths, and “N” shaped paths, which are unions of one unblocked “I” and one unblocked “A” shaped path.
Complicating this process is that an edge is only produced between two mutations if they have an unblocked path at *every* position where both bricks exist, not at *any* position. Therefore, it is necessary to keep track of both unblocked and blocked paths, forming edges between mutations only if they have an unblocked path and no blocked path. Tracking of blocked and unblocked paths is done by using “before” and “after” vertices for each brick: before a path goes through a labeled brick, it follows “before” vertices, and afterward, it follows “after” vertices. The “before” and “after” vertices allow paths to “remember” whether they have passed through a labeled brick or not.
Similarly, it is necessary for paths to “remember” whether they are going up the tree or coming down, which is done by using “Up” and “Down” vertices for each brick. The “Up” vertex of a child brick points to the “Up” vertex of its parent brick, and the “Down” vertex of a parent points to the “Down” vertex of its child.
Finally, we use a “U-Turn” vertex to create a separate path for sibling-sibling relationships as a Child-Parent-Child path. This will allow us to choose a stronger edge weight, if possible, for sibling-sibling relationship which preforms better in practice.
The following subsections describe our algorithm in detail.

3.2.1. Brick-haplotype Graph:
Our approach to building the LDGM is first to create a graph called a brick-haplotype graph. The brick-haplotype graph contains 6 or 4 vertices for each brick and 2 vertices for each haplotype. For each “labeled” brick with a mutation, we define a cluster of six vertices: Up-Before, Down-Before, Up-After, Down-After, U-Turn, and Out. (The U-turn vertices only affect path weights, not the paths themselves; see Section 3.2.5). The associated cluster with each unlabeled brick has four vertices: Up-Before, Down-Before, Up-After, and Down-After.



Finally, for each haplotype, we define a cluster of two vertices: Before and After. By assigning multiple vertices to a brick or a haplotype, it effectively “remembers” whether a given path passed through a labeled brick or not. Moreover, considering our four desired types of paths we do not want paths to go down and up within a tree.
The union of these clusters forms the vertex set of the brick-haplotype graph.
The directed edges of the brick-haplotype graph are defined as follows in order to build all “I”, “A”, “V”, and “N” shape paths.

1. “I” shaped paths: Between parent brick (top) and child bricks (bottom) we have the following edges:



The four subcases are that both bricks are labeled (left), only the parent brick is labeled (left-middle), only the child brick is labeled (right-middle), and neither brick is labeled (right).
The outgoing edges from any labeled brick are:

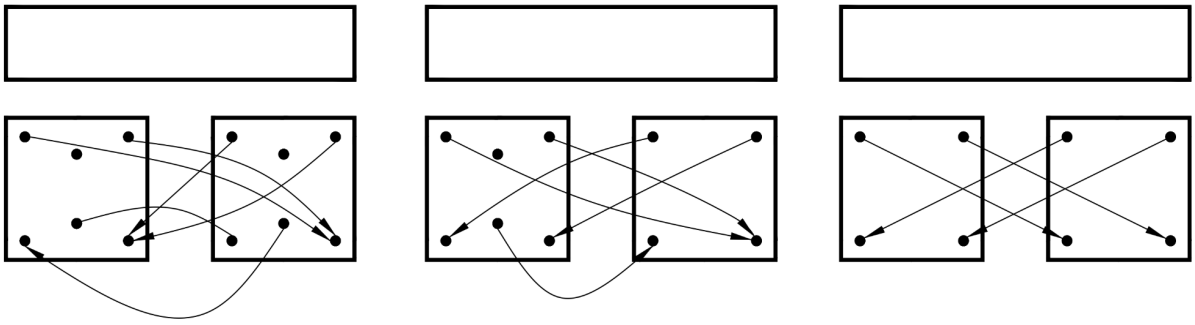
- Out to Before (Up-Before/Down-Before if the next brick is a parent/child respectively). These edges are the starting edges of paths to create the relationships, and these paths always start at before vertices because they have not been “blocked” by a labeled brick.
- Out of child to U-Turn of labeled parent, and U-Turn of labeled parent to Down-Before of child. The combination of these two types of edges creates sibling-sibling relationships with larger path weights (see section 3.4).
- Down (Before/After) of labeled parent to Down-After of a child. These edges indicate that we have already visited a labeled brick.
- Up (before/after) of labeled child to Up-After of a child. These edges indicate that we have already visited a labeled brick.

From any unlabeled brick, the outgoing edges are:

- Down-Before of an unlabeled parent to Down-Before of the child brick.
- Down-Before of unlabeled parent to Down-Before of the child brick.
- Up-Before of an unlabeled child brick to Up-Before of the parent brick and the U-turn of the parent if it is labeled.
- Up-After of an unlabeled child brick to Up-After of the parent brick.

With these edges, any path stays in the before layer until it visits a labeled brick, then switch to the after layer. Also, any path that is going down does not subsequently go up (it can go up and then down via sibling edges or U-Turn vertices). The U-Turn vertices create an alternative path, potentially with a higher edge weight, between sibling bricks passing through the shared parent (see below).

2. “A” shaped paths: Two bricks are siblings if they have a shared parent brick. Between sibling bricks, we will add the following edges:



From a labeled brick, the edges are:

- Out to Down-Before. Note that we always start the paths from out vertices; therefore, this type of edges ensures that we have not seen any labeled brick yet.
- Up vertices to Down-After to ensure that the path remembers that it already passed through a labeled brick.

From Unlabeled bricks the edges are:

- Up-Before to Down-Before
- Up-After to Down-After

With these edges we ensure that a path switches to the after level as soon as it visits a labeled brick.

3.2.2. SNP-haplotype graph:

The *SNP-haplotype graph* is the reduced graph of brick-haplotype graph such that the vertex set includes one vertex per labeled brick (the set of SNPs) and two vertices per each Haplotype (before and after). We create the SNP-haplotype graph by two-step reductions of the brick-haplotype graph. Note that if a brick has multiple mutations we pick one representative mutation since there is no distinction between mutations on the same brick with respect to conditional dependencies.

First reduction: We first create the brick-haplotype graph only using the "I" shaped paths. Then we reduce this graph to produce a "SNP-haplotype graph", as follows. We pick an out vertex of a labeled brick, say v, and create the Dijkstra paths starting from the out vertex. For each labeled brick u, if one of its before vertices is in the reach set of v but the after vertices are not, then the "I" shaped path between u and v is unblocked, and hence we add an edge between the associated SNPs in the SNP-haplotype graph. Similarly, if the before vertex of a haplotype h is reachable from v but the after vertex is not, then we add a directed edge from the associated SNP to the haplotype. After the reduction, the resulting SNP-haplotype graph contains an edge between two SNPs if there is an unblocked "I" shaped path between them. Similarly, it contains a directed edge from a SNP to a Haplotype if there is an unblocked "I" shaped path from the labeled brick to the Haplotype.

Second Reduction: Now, we add the "A" shaped paths to the brick-haplotype graph, reduce the graph to the set of labeled bricks and haplotypes similar to the previous step, and update the edges of the SNP-haplotype graph. The reason for a two-step approach is that when constructing "V" and "N" shaped paths, it is necessary to distinguish between "I" and "A" shaped paths. "V" shaped paths are the union of two "I" shaped paths, and "N" shaped paths are the union of one "I" and one "A" shaped path, but the union of two "A" shaped paths is not an allowed path. The reduction, similar to the previous step, also uses the Dijkstra algorithm. Now two SNPs are adjacent in the SNP-haplotype graph if there is an unblocked "I" shaped or "A" shaped path between them. Note that in the second reduction, we reduced the brick-Haplotype graph only to the set of SNPs; hence, we did not add any new edges to Haplotype vertices. The relationship between bricks (or SNPs) and haplotypes are essential when there is a parent-child relationship since these edges will decode recombinations.

3.3. Haplotype removal:

To create the LDGM, we only need to create the edges corresponding to "V" and "N" shaped paths. If two SNPs have edges to a haplotype in the SNP-haplotype graph, then there is a "V" shaped path or an "N" shaped path between them. Therefore, we add an edge between two SNPs in the SNP-haplotype graph if they point to the same Haplotype. Finally, by removing all Haplotype vertices from the SNP-haplotype graph, the remaining graph is the LDGM. The vertices are SNPs, and two SNPs are adjacent if:

1. They have shared positions, and an unblocked path in every tree where they both exist, forming an "I" or "A" shaped path.
2. They have no shared position, and they have an edge in the LDGM if, for some haplotype h, each has an unblocked path to h at every position where each brick exists, forming a "V" or "N" shaped path.

3.4. LDGM edge weights

In this subsection, we assign weights to the edges of the brick-haplotype graph using a well-motivated heuristic. LDGMs can have numerous weak edges due to distant genealogical relationships and rare recombination events. We eliminate weak edges by applying an edge-weight threshold, and the choice of threshold allows us to trade accuracy for sparsity.

Edge weights between adjacent bricks are defined by their respective frequencies. Suppose that a parent brick has frequency f_1, and a child brick has frequency f_2; if these bricks have mutations a, b respectively, then the frequency of observing both mutations is equal to f_2. The squared correlation between a mutation on the parent brick and a mutation on the child brick is:

$$r^2_{\text{parent}-\text{child}} = \frac{(f_2 - f_1 f_2)^2}{f_1(1-f_1)f_2(1-f_2)} = \frac{f_2(1-f_1)}{f_1(1-f_2)} = O_2/O_1$$

where $O_a = f_a/(1-f_a)$ denotes the odds of brick a. This correlation is small when $f_1 \gg f_2$.

Now, consider a sibling relationship, where the siblings have frequencies f_1, f_2. Since siblings have no shared descendants, their squared correlation is:

$$r^2_{sib-sib} = \frac{f_1 f_2}{(1-f_1)(1-f_2)} = O_1 O_2$$

which is small when $f_1 + f_2 \ll 1$. This correlation is one when $f_1 + f_2 = 1$, which occurs when every haplotype is descended from the shared parent of the two siblings, and the expression is greater than one when $f_1 + f_2 > 1$, which does not occur when the siblings have no shared descendants (in these cases, we threshold it at one).

These correlations factor across parent-child relationships. If haplotype 1 is the parent of 2, and 2 the parent of 3, then the squared correlation between 1 and 3 is $r^2_{13} = r^2_{12} r^2_{23}$. The same occurs if 1 and 2 are siblings. This allows us to interpret $d = -\log r^2$ as a distance and to sum distances across paths:

$$d_{13} = d_{12} + d_{23} = -\log r^2_{12} r^2_{23} = -\log r^2_{13}.$$

The property holds for any recombination-free ("A" or "I" shaped) path. For "V" and "N" shaped paths, which can be written as the union of two A or I shaped paths, the path distance is equal to the sum of the path distances of their component paths. This is a heuristic: the resulting path distance is no longer equal to the r^2 between the respective bricks, but it still measures the strength of their relationship.

We observed that sibling-sibling relationships between low-frequency mutations always have small weights but that sometimes, these relationships are important in the LDGM, and excluding them leads to high mean-squared error. This occurs when the two low-frequency siblings have a shared parent

mutation that also has a low frequency, such that the two mutations are weakly correlated but strongly correlated *conditional* on that parent. This problem can be prevented by picking the alternative child-parent-child path, superseding the sibling-sibling path. For labeled bricks, we add a "U-Turn" vertex at the shared parent creating a stronger edge between its descendent siblings. If the children are 1,2 and the labeled parent is 3, the U-turn distance is

$$d_{12} = d_{13} + d_{23} = -\log \frac{O_1 O_2}{O_3^2}$$

which is much smaller than $-\log O_1 O_2$ when O_3 is small. With this change, the MSE is similar across SNPs at different allele frequencies (Extended Data Figure 7).

3.5. Precision matrix inference

Our precision matrix inference method is based on the graphical lasso (GLASSO)[5], and in particular the DP-GLASSO ("dual-primal GLASSO") method of Mazumder and Hastie[6]. We compute a precision matrix whose nonzero edges correspond to the edges of the LDGM, using an L1 penalty to produce additional sparsity. Then, we remove the L1 penalty but restrict the unpenalized optimization to the edges that were found in the L1-penalized step. The motivation for this procedure is that L1 penalization causes downward bias in the nonzero entries. An L0-penalized optimization would induce sparsity without this bias, but the resulting optimization would be nonconvex. Therefore, we use the L1 penalty for edge selection, and we relax the penalty to estimate the precision matrix entries of the edges that were selected. This procedure is guaranteed to improve an L0-penalized objective function compared with the L1-penalized solution, as it improves the likelihood term without affecting the penalty term.

3.5.1 Modified DP-GLASSO

GLASSO and DP-GLASSO maximize the L1-penalized Gaussian likelihood:

$$\hat{P} = \text{argmax}_P \frac{1}{2}(\log|P| - Tr(PR)) - \lambda\left|\left|P\right|\right|_1 \quad (1)$$

where P is the estimated precision matrix, R is the sample correlation matrix, λ is the regularization parameter, $|\cdot|$ denotes the determinant, and $\left|\left|\cdot\right|\right|_1$ denotes the L1 norm. In brief, DP-GLASSO iteratively updates one row/column of P at a time, solving the following L1-penalized regression problem:

$$P_{j\neq i,i} = \text{argmin}_{w\in\mathbb{R}^{m-1}} \frac{1}{2} w^T (P^{-1})_{j\neq i, j\neq i} w - R_{i,j\neq i} w + \lambda\left|\left|w\right|\right|_1. \quad (2)$$

(This is L1-penalized ordinary least squares linear regression of the vector $R_{i,j\neq i}$ on the matrix $(P^{-1})_{j\neq i,j\neq i}$). It does so via entry-wise coordinate descend on w. DP-GLASSO differs from the original GLASSO method in that it directly estimates the sparse matrix P rather than its inverse; this difference allows us to restrict DP-GLASSO to the edges of the LDGM.

Our precision matrix inference approach differs from DP-GLASSO in three ways. First, we constrain optimization (1) such that P has nonzero entries only for edges of the LDGM, and this constraint means that we must solve a much smaller lasso regression problem in each coordinate descent update (2). The constrained optimization is:

$$\hat{P} = \text{argmax}_P \frac{1}{2}(\log|P| - Tr(PR)) - \lambda\left|\left|P\right|\right|_1$$
$$\text{subject to } P_{ij} = 0 \text{ for } (i,j) \notin E.$$

Notice that with the constraint, the objective function only depends on R_{ij} for $(i,j) \in E$.

Second, we include SNP i itself in its own set of neighbors; this differs from (2), where SNP i is excluded. In the corresponding diagonal entry of R, which would normally be one, we place a zero (see below). We

found that including SNP i as its own neighbor improves the convergence rate in practice, and we provide a heuristic justification below. With these two changes, the optimization at each step is:

$$\text{minimize}_{w\in\mathbb{R}^{|N(i)|}} \frac{1}{2} w^T (P^{-1})_{N(i),N(i)} w - (R-I)_{i,N(i)} w + \lambda \left|\left|w\right|\right|_1. \quad (3)$$

where $N(i)$ denotes the neighbors of SNP i, including i itself. After solving (3), similar to ref. [6], we update the corresponding row and column of P:

$$P_{N(i)\setminus i,i} \leftarrow w_{\setminus i}, \qquad P_{i,i} \leftarrow 1 + \hat{R}_{N(j),j} \cdot w.$$

(The formula for the diagonal entry is the same as DP-GLASSO and GLASSO.)

Third, we solve optimization (3) using an iterative proximal gradient method[7], which differs from the element-wise coordinate descent approach of DP-GLASSO and GLASSO. We made this change in order to avoid working with the dense matrix P^{-1}.

3.5.2. Justification for modified coordinate descent approach

We justify our modified approach heuristically, referring to ref. [6] for a formal discussion. Let $L = I - P$. Multiplying both sides of the equation

$$R = P^{-1}$$

by *L*, it becomes

$$RL = R - I = P^{-1}L$$

and restricting to column *j*,

$$P(R-I)_j = L_j.$$

This equation already resembles an update equation for the *j*th column of P. When we restrict the set of edges to those appearing in the LDGM, the log-likelihood (1) depends only on the corresponding elements of R. The algorithm can safely ignore non-neighbor entries not only of P but also of R. Indeed, the gradient of the log-likelihood with respect to the entries P_E is:

$$\nabla_{P_E} \log \text{likelihood} = (R_E - (P^{-1})_E)$$

such that when the likelihood is maximized, for every SNP *j*,

$$L_{N(j),j} = \left(P|_{N(j)}\right)(R-I)_{N(j),j}.$$

Without an L1 penalty, this equation suggests the following algorithm:

4. Initialize at $P = I$
5. Loop over SNPs *j* one at a time, setting

$$P_{N(j),j} = I_{N(j),j} - \left(P|_{N(j)}\right)(R-I)_{N(j),j}.$$

With an L1 penalty, the unpenalized update equation is modified to produce the SNP-wise optimization problem (3).

3.5.3. Detailed description of the algorithm

In detail, the modified coordinate descent outer loop is as follows. We initialize the precision matrix at $P = I$. We loop through the SNPs $j = 1, \ldots, M$ five times with an L1 penalty of 0.1, restricting the optimization to the edges of the LDGM that have an edge-weight distance of at most 4 (heuristically equivalent to $r^2 \geq 1/e^4$). At each iteration, we solve optimization (2) (see below for details on the inner loop), we update row/column *j* of the P, and we check that P is still positive definite (by attempting to compute its Cholesky factor). Occasionally it is not, indicating that the lasso regression procedure failed to obtain a good solution at the previous step. Typically this is caused by a too-large step size, so when it occurs, we reduce the minimum step size by half and try the previous lasso regression step again; if reducing the minimum step size three times does not fix the problem, we simply use $w^{(0)}$ as the solution (i.e. the non-L1-penalized solution). We have not observed instances where sporadic

convergence issues for individual lasso problems lead to low-quality solutions globally, probably because the unpenalized fallback solution is acceptable within the larger optimization routine.
After iterating through every SNP five times, we repeat the procedure without an L1 penalty, restricting to the edges that were found in the previous step (precision matrix entries smaller than 1e-12 are considered non-edges) and initializing at the previous solution. We iterate through every SNP 25 times; each iteration is faster, because the inner loop is no longer required. This approach of removing the L1 penalty after initially using to find a smaller set of edges is designed to approximate the effect of an L0 penalty (which would lead to a nonconvex optimization).
Details of the inner loop are as follows. We initialize

$$w^{(0)} = \left(P|_{N(i)}\right)R_{N(i),i}$$

where

$$P|_A \coloneqq P_{AA} - P_{AA^C}\left(P_{A^CA^C}\right)^{-1}P_{A^CA}$$

is the Schur complement of P with respect to the SNPs not in A. The inverse of $P|_A$ is equal to $(P^{-1})_A$. $w^{(0)}$ is the solution to optimization (3) when $\lambda = 0$. We update $w^{(0)}$ using the proximal gradient approach of ref[7] . Let

$$S_\gamma(x) = \operatorname{sgn}(x)\min(0, \operatorname{abs}(x) - \gamma).$$

This function maps $[-\gamma, \gamma]$ to zero and $\pm(\gamma + |x|)$ to $\pm|x|$. At iteration *k* of the inner loop, we update *w* with step size $\gamma^{(k)}$ using:

$$w^{(k+1)} = S_{\lambda\gamma^{(k)}}\left(w^{(k)} - \gamma\nabla F\right)$$

where $F(w) = \frac{1}{2}w^T\left(P|_{N(i)}\right)^{-1}w - R^T_{N(i),i}w$. It can be expensive to compute $\left(P|_{N(i)}\right)^{-1}w$ repeatedly (i.e., solving $\left(P|_{N(i)}\right)x = w$) using the approach described in Section 5.1.2. This is a sparse system of equations, but its size scales with the number of SNPs in the LD block, which is much larger than the size of $N(i)$. We solve these equations more efficiently by computing the Cholesky factor of $P|_{N(i)}$. We reorder P to put $N(i)$ last, compute the Cholesky factor A of the reordered P, and discarding the rows and columns of A besides $N(i)$. (If $N(i)$ were first instead of last, this would be equivalent to the Cholesky of $P_{N(i),N(i)}$, but here it is the Cholesky of $P|_{N(i)}$ instead). This takes roughly as long as solving $\left(P|_{N(i)}\right)x = w$ once. Then, instead of solving $\left(P|_{N(i)}\right)x = w$ at each iteration, we instead solve $AA^Tx = w$, which is fast.
We initialize the step size at $\gamma^{(0)} = 1$, and at each iteration, we check that the update succeeded in reducing the objective function; if the step size is too large, sometimes the iterative procedure will begin to diverge, causing increases in the objective function. When this occurs, we reduce the step size by half and try again, until reaching a minimum step size of 0.1 (by default) at which point we terminate the loop. Otherwise, we terminate the loop after 10 iterations.

4. Low rank approximations to the local LD matrix

Low-rank approximations for the local LD matrix have been used by H. Shi et al.[8,9] and others [10,11]. This approach is useful statistically with small reference panels, as small eigenvalues in a reference LD matrix become large and noisy when that matrix is inverted. It is also useful computationally, as the amount of data required to store a rank-*k* LD matrix is similar to the amount required to store a degree-k LDGM precision matrix.
We distinguish between two approximations: the rank-*k* representation of the LD correlation matrix, $R_k = U_k\Lambda_kU_k^T$, and the rank-*k* representation of its inverse, $P_k = U_k\Lambda_k^{-1}U_k^T$. R_k and P_k are not inverses of each other, and their product is the projection matrix $U_kU_k^T$. R_k and P_k have different statistical properties. On the one hand, R_k can be quite a good approximation to R, with an MSE similar to that of the LDGM precision matrix inverse for k around 50 (see Extended Data Figure 4). It makes sense that R_k works well by this metric: the MSE is

$$MSE_k = \frac{1}{m^2} \sum_{j=k+1,\ldots,m} \lambda_j^2$$

the sum of the squares of the small eigenvalues. On the other hand, P_k is not an especially good approximation to R^{-1}. $P_k R$ is the projection matrix onto the span of U_k, which means that $P_k R^2 = R_k$ is expected to be a good approximation for R, but $P_k R$ is a poor approximation for I.
To quantify this, we define the *alternative MSE ratio* as:

$$f(P) := \frac{Tr\big((I - PR)(I - RP)\big)}{Tr(PRRP)}.$$

The numerator quantifies how close PR and RP are to I, and the denominator ensures that when P is nearly the all-zeros matrix, the error is large. (We also report the alternative MSE without the denominator in Extended Data Figure 2.) The numerator quantifies the mean-squared difference between a random vector x and the vector PRx. If $x \sim N(0, I)$:

$$E\left(\left||x - PRx|\right|^2\right) = Tr\big((I - PR)^T (I - PR) E(xx^T)\big)$$
$$= Tr\big((I - RP)(I - PR)\big).$$

Extended Data Figure 4 shows the alternative MSE ratio for the LDGM precision matrix compared with P_k for different values of k. Even with $k = 500$, the alternative MSE ratio is much higher for P_k than for the LDGM precision matrix.
These error metrics imply that a low-rank approximation is appropriate for some applications, but not for others. In applications that involve multiplying a vector by R_k (e.g., simulating GWAS summary statistics), the approximation is satisfactory. In applications that involve P_k, which seems to include existing methods, we caution that large k might be required to obtain adequate accuracy. The best choice of *k* might depend on whether within-sample or out-of-sample LD is used, with better within-sample performance at large *k* and better out-of-sample performance at small *k*.

5. Calculations involving the LDGM precision matrix

In this section, we describe how LDGM precision matrices can be used to perform useful computations. Section 5.1 describes basic operations, and section 5.2 describes calculations involving a Gaussian model for the effect size distribution (i.e., the model underlying BLUP).

5.1 Basic matrix operations

When deriving efficient formulas involving LDGM precision matrices and GWAS summary statistics, an important consideration is that most of the time, some of the SNPs in the LDGM will be missing from the GWAS summary statistics. (It is normally less significant if SNPs in the summary statistics are missing from the LDGM; see below). Missingness cannot be handled by discarding the rows and columns of the precision matrix that are missing in the summary statistics, since the resulting matrix is not the precision matrix of the non-missing SNPs. Let R be a correlation matrix, let $P = R^{-1}$ be a precision matrix, and let the indices $0,1$ denote missing and non-missing SNPs respectively, such that

$$R = \begin{pmatrix} R_{00} & R_{01} \\ R_{10} & R_{10} \end{pmatrix}, \qquad P = \begin{pmatrix} P_{00} & P_{01} \\ P_{10} & P_{10} \end{pmatrix}.$$

$R_{11}$is not equal to P_{11}^{-1}; instead, it is equal to the Schur complement of P_{11}:

$$R_{11}^{-1} = P/P_{00} := P_{11} - P_{10} P_{00}^{-1} P_{01}.$$

In general, when performing precision matrix operations with missing data, there are two choices for how to proceed: either compute the Schur complement explicitly, or perform computations using the definition of the Schur complement that avoid computing it. When a large number of SNPs are non-missing, it is better to avoid computing the Schur complement (which might be dense), but if there are

only a handful of non-missing SNPs (which is the case, e.g., in certain computations that arise when computing the precision matrix), it is better to simply compute the Schur complement (in particular, its Cholesky decomposition).

Many operations involve solving a sparse system of linear equations, i.e., $Ax = y$ where A is a sparse, symmetric, positive definite matrix. We do so using the MATLAB backslash operator ("$x = A\backslash y$"), which computes a sparse Cholesky factor of A.[12,13]

5.1.1 Precision matrix by vector multiplication

Suppose we wish to solve

$$R_{11}x = y \tag{4}$$

given a vector or tall matrix y. We do so using the definition of P/P_{00}:

$$x = P/P_{00}y = P_{11}y - P_{10}z, \qquad P_{00}z = P_{01}y.$$

This approach is extremely fast compared with directly solving the dense system of equations (4). It is implemented in the *precisionMultiply* function (see Code Availability).

5.1.2. Precision matrix by vector division

Suppose we wish to compute

$$y_1 = R_{11}x_1 \tag{5}$$

given a vector or tall matrix x. We do so by filling in zeros for the missing SNPs to form $x = [0\ x_1]^T$ and solving

$$Py = x, \qquad y = [y_0\ y_1]^T$$

and discarding the unwanted entries y_0.

Typically, this approach is only modestly faster than evaluating equation (5), depending on the number of SNPs in an LD block (with a larger improvement when there are more SNPs). It is implemented in the *precisionDivide* function (see Code Availability).

5.1.3. Computing the determinant

Suppose we wish to compute the determinant of R_{11}, denoted $|R_{11}|$. We do so with the identity

$$|R_{11}| = |P/P_{00}|^{-1} = \frac{|P_{00}|}{|P|}.$$

To evaluate the determinant in a numerically stable manner, we actually compute the log-determinant, using the identity

$$\log|P| = 2\sum_j \log A_{jj}\,, \qquad P = A^TA$$

where A is the Cholesky factor of P. Typically, this approach is substantially faster than a similar approach involving R.

More generally, we might wish to compute the determinant of a slightly different matrix:

$$|R_{11}^{-1} + T_{11}|$$

where T_{11} is a sparse matrix (for example, see below). To do so, we fill in zeros where there are missing SNPs to form a matrix T of size equal to that of P. The Schur complement of $P + T$ is

$$(P+T)/(P+T)_{00} = P_{11} + T_{11} - P_{10}P_{00}^{-1}P_{01} = R_{11}^{-1} + T_{11}$$

so the determinant can be computed as

$$\left|R_{11}^{-1} + T_{11}\right| = \frac{|P+T|}{|P_{00}|}.$$

5.1.4. Computing the likelihood of the GWAS summary statistics under the null

For a null trait ($\beta = 0$), the GWAS *Z* scores, z, follow the following distribution[14, 15]:

$$z \sim N(0, R_{11}) \quad (6)$$

Equivalently,

$$(P/P_{00})z \sim N(0, P/P_{00}).$$

The likelihood of (6) is:

$$\log f(z) = -\frac{1}{2}(m \log 2\pi + |P/P_{00}| + z^T(P/P_{00})z).$$

5.2. Calculating the BLUP and related quantities

Let β be the vector of causal per-allele effect sizes for a trait y, such that

$$y = X\beta + \epsilon$$

where X is a row vector of genotypes and ϵ is a noise term. The best linear unbiased predictor (BLUP) is the posterior mean of β under a Gaussian model:

$$\beta \sim N(0, \Sigma).$$

This model includes as an important special case the infinitesimal model, $\Sigma \propto I$, and it can be generalized by making Σ itself a random variable. Moreover, the diagonal entries of Σ can depend on the allele frequencies in a manner consistent with known allele frequency-dependent architecture.

5.2.1. *BLUP-ldgm*

The distribution of the GWAS Z scores, z, given β is:

$$z|\beta \sim N\left(n^{\frac{1}{2}}RD\beta, R\right)$$

where D is a diagonal matrix that converts β into normalized units: under Hardy-Weinberg equilibrium,

$$D_{jj} = \left(2p_j(1-p_j)\right)^{\frac{1}{2}}.$$

The unbiased effect size estimates are:

$$\hat{\beta} = n^{-\frac{1}{2}} D^{-1}Pz$$
$$\hat{\beta}|\beta \sim N(\beta, n^{-1}D^{-1}PD^{-1}).$$

(Note that these estimates are extremely noisy; the diagonal elements of P can be greater than 10, and the diagonal elements of D^{-1} are also large for low-frequency SNPs). The posterior mean is:

$$\hat{\beta}_{\mathrm{BLUP}} \coloneqq E(\beta|z) = (\Sigma^{-1} + V^{-1})^{-1}V^{-1}\hat{\beta}, \qquad V = n^{-1}D^{-1}PD^{-1}.$$

This simplifies to:

$$\hat{\beta}_{\mathrm{BLUP}} = n^{1/2}\Sigma D(nD\Sigma D + P)^{-1}Pz$$

which can be evaluated efficiently as described in sections 5.1.1 and 5.1.2, including when data are missing. This is implemented in the function *BLUPx*.

5.2.2 Simulating summary statistics

LDGMs can be used to simulate summary statistics from their asymptotic sampling distribution (equation 6). First, we choose an effect size covariance matrix Σ. In our single population simulations, this is chosen to be proportional to I, with a scalar that matches the chosen heritability:

$$h^2 = E(\beta^T DRD\beta) = Tr(DRD\Sigma)$$
$$= \sigma^2 Tr(D^2) \text{ when } \Sigma = \sigma^2 I.$$

In the multiple population setting, we concatenate the vectors of effect sizes for each population, and we choose a populations-by-populations covariance matrix, Σ_{pop}. The diagonal entries are chosen to match the heritability in each population, and the off-diagonal entries are chosen to match the desired cross-population genetic correlation, i.e.:

$$r_{pop}(a,b) = \frac{\Sigma_{pop}(a,b)}{\sqrt{\Sigma_{pop}(a,a)\Sigma_{pop}(b,b)}}.$$

Then the covariance matrix of the concatenated effect sizes is:

$$\Sigma = I \otimes \Sigma_{\text{pop}}$$

where I is the identity matrix of size equal to the number of SNPs and $\otimes$ denotes the Kronecker product. If any SNPs are missing in some of the populations, the corresponding rows and columns are dropped from Σ. More generally, the cross-population effect size distribution of each SNP can be chosen from a mixture of normal distributions.

To sample the summary statistics efficiently, first we sample the true effect sizes β, and then (for each population separately) we compute the mean of their summary statistics as:

$$E(z|\beta) = n^{1/2}P^{-1}D\beta$$

We produce noise with covariance matrix R by computing the Cholesky of P, $P = AA^T$, and solving

$$A\epsilon = x, \qquad x \sim N(0, I).$$

The summary statistics are $z = n^{1/2}P^{-1}D\beta + \epsilon$. These equations are implemented in the function *simulate_sumstats_populations.*

5.2.3. BLUP likelihood

The likelihood of the summary statistics under the BLUP model, not given β, is:

$$z \sim N(0, R + nRD\Sigma DR)$$

where $RD\Sigma DR$ is the covariance matrix of the per-s.d. marginal effect sizes, $\alpha = corr(X, y) = RD\beta$. Equivalently,

$$Pz \sim N(0, P + nD\Sigma D) \tag{7}$$

The likelihood of equation (7) is:

$$\log f(z) = -\frac{1}{2}(m \log 2\pi + |P + nD\Sigma D| + (Pz)^T(P + nD\Sigma D)Pz) \tag{8}$$

When there are missing SNPs such that we only observe z_1, we assume that their effect sizes are zero. Let $S = D\Sigma D$ and S_{11} denote the non-missing block. Let $x = (P/P_{00})z_1$. Equation (8) becomes:

$$\log f(z_1) = -\frac{1}{2}(m \log 2\pi + |P/P_{00} + nS_{11}| + x^T(P/P_{00} + nS_{11})x)$$

which can be evaluated efficiently as described in sections 5.1.1, 5.1.2, 5.1.3. This is implemented in the function *GWASLikelihood*.

The gradient of the likelihood can also be computed efficiently. Let $y = (D\Sigma D + n^{-1}P/P_{00})^{-1}x$. The partial derivative with respect to Σ_{jj} is:

$$\frac{\partial}{\partial \Sigma_{jj}} \log P(z|D) = D_{jj}^{-2}\left(\begin{bmatrix} D\Sigma D & 0 \\ 0 & 0 \end{bmatrix} + n^{-1}P\right)^{-1}_{jj} + D_{jj}^{-2}y_j^2.$$

The first term appears like it would be slow to evaluate, as it is an element of the inverse of a large sparse matrix, and inverting this matrix directly would be slow. However, efficient approaches can be used to extract the diagonal elements of the inverse without computing the entire matrix.[16] This approach is implemented in the function *GWASLikelihoodGradient*, which computes the derivatives with respect to all of the diagonal elements of Σ.

6. Evaluation of BLUP-ldgm and related methods

6.1 BLUP simulations with individual-level data

To evaluate the performance of BLUP-ldgm, we performed simulations involving individual-level genotypes and simulated phenotypes. We simulated per-s.d. causal effect sizes β from an i.i.d. normal distribution for all variants in the LDGM with MAF>0.01 on chromosome 22, with variance equal to h^2/m. We simulated phenotypes by computing $Y = X\beta + \epsilon$, where X is the matrix of standardized genotypes and $\epsilon \sim N(0,1-h^2)$. We computed marginal summary statistics $z = n^{-1/2}X'y$, LD correlation matrices $R = X'X/n$ for each LD block, and a GRM $G = XX'/m$. We ran *BLUP-ldcov* (equation 5), *BLUP-ldgm* (equation 4) and *BLUP-indiv*, which operates on y and the exact GRM:

$$\beta_{BLUP-indiv} = \frac{h^2}{m}X'(h^2\,G + (1-h^2)I)^{-1}y. \tag{6}$$

We evaluated prediction accuracy using two approaches. The within-sample prediction accuracy was defined as

$$r^2_{within-sample} = h^2 \cdot corr(X\beta, X\beta_{BLUP})^2 \tag{7}$$

where *corr* denotes the sample correlation. The pseudo-out-of-sample prediction accuracy was

$$r^2_{out-of-sample} = \frac{(\beta' R\beta_{BLUP})^2}{(\beta' R\beta)(\beta'_{BLUP}R\beta_{BLUP})}. \tag{8}$$

The difference between the two definitions is large in these simulations due to their small sample size; the difference disappears at large sample size.

We also evaluated the theoretical prediction accuracy using two different formulas, corresponding to the within-sample vs. out-of-sample prediction accuracy. For the theoretical within-sample prediction accuracy, we used the formula of ref.[17]:

$$\tilde{r}^2_{within-sample} = \frac{h^2}{1+\frac{m_{eff}}{nh^2}\left(1-\tilde{r}^2_{within-sample}\right)} \tag{9}$$

where m_{eff} was defined using the genetic relatedness matrix:

$$m_{eff} = \frac{Tr(G)^2}{Tr(G^2)} \approx 609. \tag{10}$$

(Note that with this definition, $m_{eff} \leq min(m,n)$; at large n, it agrees with the definition of ref.[18]). The theoretical out-of-sample prediction accuracy was defined using a block-diagonal approximation to R, with the formula of ref.[19]:

$$\tilde{r}^2_{out-of-sample} = Tr\left(S\left(\frac{1}{N}I+S\right)^{-1}S\right), \qquad S = \frac{h^2}{m}R. \tag{11}$$

6.1.1 Simulations involving 1000 Genomes genotypes

We performed small-scale individual-level simulations involving 1000 Genomes haploid genotypes (N~1000) on chromosome 22 (Supplementary Figure 2). We compared BLUP-ldgm and BLUP-ldcov with *BLUP-grm*, which utilizes the genetic relatedness matrix (GRM) and the individual-level phenotypes. In AFR and EUR, all three methods had similar performance; BLUP-grm performed slightly better when the heritability was high. Prediction r^2 was concordant with theoretical expectations[17,19]. In AMR, on the other hand, BLUP-grm performed significantly better, while BLUP-ldgm performed significantly worse, due to long-range admixture LD. We also tested LDGM precision matrices estimated with different parameter settings in these simulations; they produced near-identical results (Supplementary Figure 3).

6.1.2 Simulations involving UK Biobank genotypes

In order to investigate the effect of LD reference misspecification, we performed individual-level simulations involving UK Biobank European-ancestry typed and imputed genotypes (N=1k-20k) on chromosome 1. We compared BLUP-ldgm with BLUP-grm and evaluated their out-of-sample prediction accuracy, using 5k unrelated individuals as a validation set. BLUP-ldgm performed almost equally well as BLUP-grm, despite its use of a reference LD panel instead of individual-level data (Supplementary Figure 4), consistent with our analysis of real UK Biobank summary statistics (Figure 5a).

6.2 Simulations involving simulated summary statistics

Individual-level simulations become impractical computationally at a whole-biobank scale (with hundreds of thousands of individuals and millions of SNPs), and they cannot be performed at sample sizes larger than the largest available genotyping datasets. To investigate the performance of BLUP-ldgm at very large sample sizes, we simulated per-s.d. effect sizes under an infinitesimal model with heritability 0.1, at whole-genome scale, and we simulated summary statistics from their asymptotic sampling distribution (see section 5.2.2), with a range of sample sizes. For the reference LD, we used (1) the EUR LDGM precision matrices and (2) the EUR sample correlation matrices. We applied both BLUP-ldgm and BLUP-ldcov to the summary statistics in each case.

BLUP-ldgm and BLUP-ldcov produced nearly identical results at small to realistic GWAS sample sizes ($N \leq 10^6$), becoming divergent when the GWAS sample size became extremely large ($N \geq 10^7$) as the reference panel size stayed constant ($N \approx 10^3$) (Supplementary Figure 5). These estimates indicate that extremely well-powered GWAS require more accurate LD reference panels (see Discussion), consistent with what has been reported in fine mapping studies.[20] They are consistent with our analysis of UK Biobank summary statistics, where the difference between reference vs. within-sample LD was more apparent for well powered quantitative traits (like height) than for low-prevalence diseases (like diabetes) (Figure 5a).

6.3 Summary statistic imputation

An LD reference panel can be used to perform linear imputation of GWAS summary statistics using *impG*[21–23], and this approach can be applied to non-missing, held-out SNPs to assess goodness of fit for a reference LD matrix. The *impG* statistic for unobserved SNP x_j estimates the correlation between y and $\widetilde{x_j}$, where $\widetilde{x_j}$ is the linear projection of x_j onto the SNPs that are observed. Our implementation of ImpG using LDGMs, *impG-ldgm*, utilizes the following formula:

$$\tilde{Z}_J = V_J P^{-1} V_I' \left(P / P_{JJ}\right) Z_I \quad (12)$$

where $\tilde{Z}_J$ is the imputed Z score for indices J, Z_I is the observed summary statistics for indices I disjoint from J, P/P_{JJ} denotes the Schur complement of P (see Supplementary Note), and V_I, V_J are the projection matrices onto indices I, J respectively. This formula can be evaluated efficiently from right to left using sparse matrix operations (Section 5.1.1-5.12). Our correlation matrix-based implementation, *impG-ldcov*, uses the formula:

$$\tilde{Z}_J = R_{JI} R_{II}^{-1} Z_I. \quad (13)$$

As recommended by Pasaniuc et al., we used a ridge penalty of 0.1, replacing the original sample correlation matrix R with $0.9R + 0.1I$, which was approximately optimal in our cross-validation analyses as well.

We analyzed summary statistics from two traits from UK Biobank[24,25] and one trait (asthma) from the Global Biobank Meta-analysis Initiative (GBMI)[26], which provides well-powered summary statistics for several continental ancestry groups. We performed five-fold cross validation, leaving out every fifth SNP

by genome position and imputing its Z score using either impG-ldgm or impG-ldcov. Across the five iterations, we averaged the prediction r^2 and summed the runtime.

impG-ldgm had a higher cross-validated correlation with the held-out summary statistics compared with impG-ldcov (Supplementary Figure 6a). The difference was small but significant, and impG-ldgm also had smaller runtime (Supplementary Figure 6b). Consistent with our UK Biobank BLUP analysis, this analysis indicates that our precision matrices usefully regularize the correlation matrix, thereby improving out-of-sample performance in real GWAS datasets.

6.4 Summary statistic quality control

A similar cross validation-like approach is used by the *DENTIST*[27] method to perform quality control for individual SNPs and by the *SLALOM*[28] method to detect false positives in fine mapping analyses. Intuitively, if the observed association statistic of a SNP differs significantly from its value after masking and imputation, it means that there is mismatch between the LD matrix and the summary statistics for that SNP or its LD partners. We implemented a simplified version of DENTIST with LDGMs. We partition SNPs randomly into subsets I, J and apply the formula:

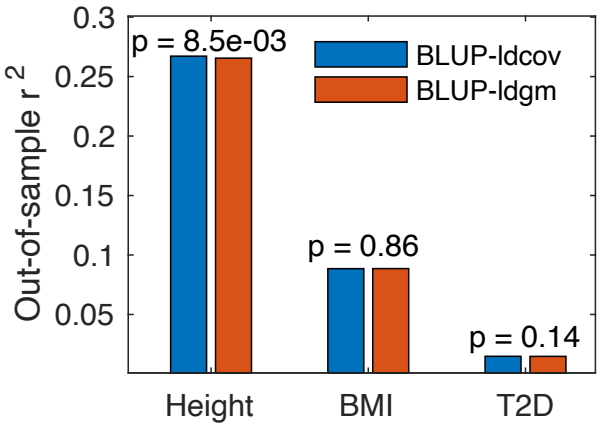
$$T_{dentist}(j) = \frac{(Z_j - \tilde{Z}_j)^2}{1 - R_{IJ}(P/P_{II})R_{JI}}$$

where SNP *j* belongs to J. $\tilde{Z}_j$ is the imputed Z score for SNP j, applying impG-ldgm to the SNPs in I. R_{IJ} (and R_{JI}) is slow to compute; we do so by direct inversion, and we are unaware of a more efficient approach.

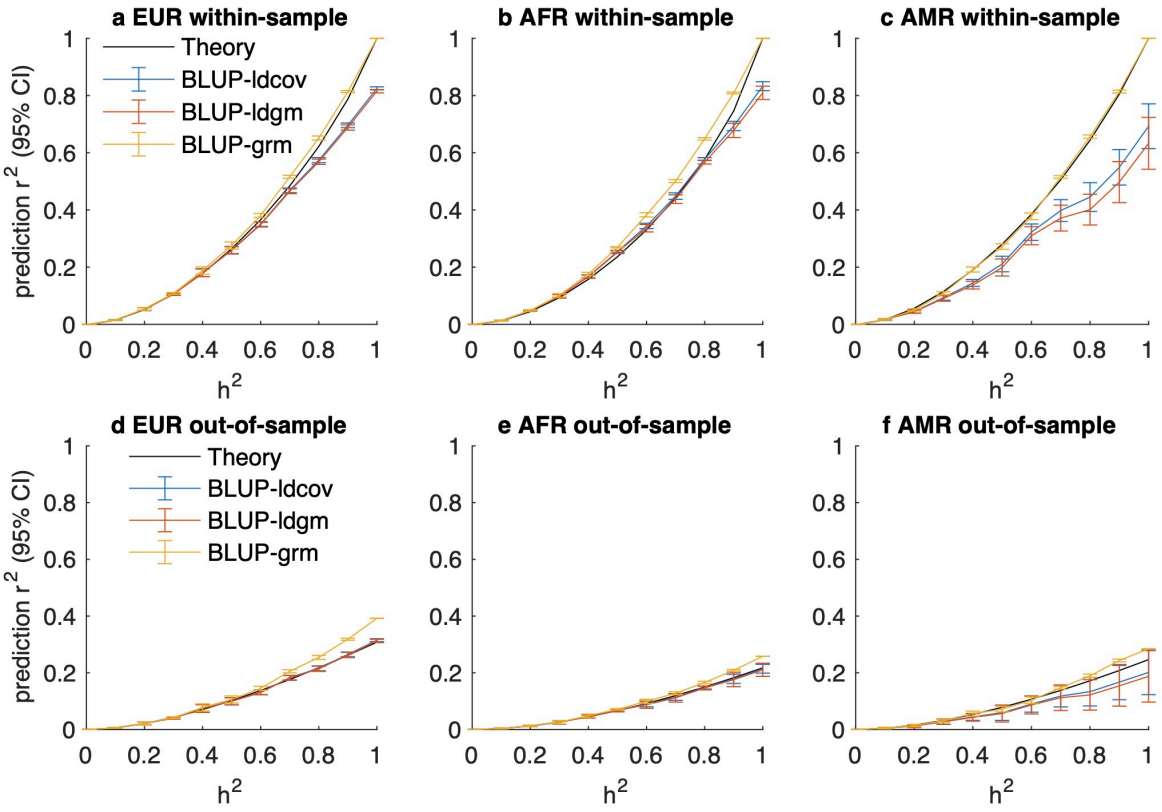
Unlike the original DENTIST implementation, which involves an iterative procedure of discarding SNPs and recomputing statistics, we calculated DENTIST statistics without iteration. This choice made it possible to compare DENTIST statistics between our two implementations, as they produce statistics for the same set of SNPs.

We applied both implementations of DENTIST to summary statistics from UK Biobank and GBMI. The mean DENTIST statistic was roughly concordant, indicating that both LD matrices provide similarly good fit for the summary statistics. The two versions of DENTIST were modestly correlated ($r \approx 0.5$), such that the set of poorly fitting SNPs is overlapping but largely distinct (Supplementary Figure 7); probably, the modest correlation reflects the fact that DENTIST is specifically designed to be sensitive to subtle LD mismatch.

7. Supplementary Figures

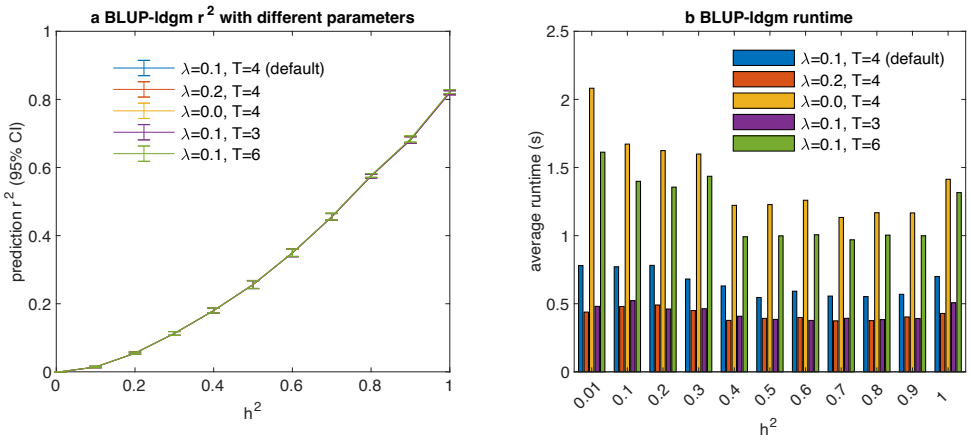


Supplementary Figure 1: out-of-sample prediction accuracy of BLUP-ldgm and BLUP-ldcov for three traits. We applied each method to European-ancestry, non-UK Biobank summary statistics for height, BMI and T2D, and we evaluated their prediction accuracy in UK Biobank unrelated, European-ancestry individuals. For height, BLUP-ldgm had a significantly greater r^2. We calculated p-values using a two-sided Binomial test across LD blocks. For numerical results, see Supplementary Table 9.

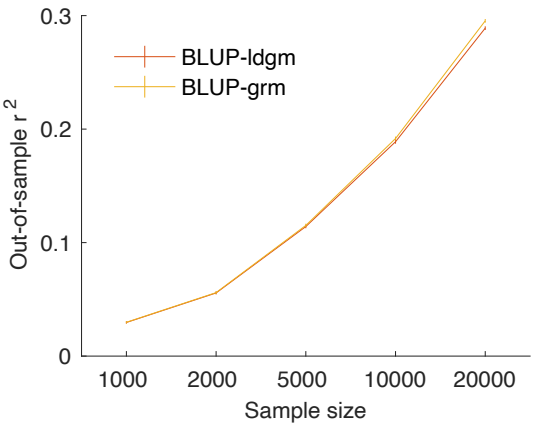


Supplementary Figure 2: **performance of BLUP-ldgm in simulations with individual-level data**. We simulated individual level phenotypes using 1000 Genomes genotypes (n=1006) for chromosome 22 (20 LD blocks), in EUR, AFR and AMR, under an infinitesimal genetic architecture. We compared BLUP-ldcov, which operates on LD correlation matrices and Z scores; BLUP-ldgm, which operates on LDGM precision matrices and Z scores; and BLUP-grm, which operates on the genetic relatedness matrix and individual-level phenotypes. We drew effect sizes from a normal distribution, varying the heritability. 10 replicates were performed. Within-sample prediction accuracy (panels a-c) was defined as the squared correlation between $X\beta_{BLUP}$ and $X\beta$ (equation 7). Pseudo-out-of-sample prediction accuracy (panels d-f) was defined using equation (8). Theoretical prediction accuracy in panels (a-c) was defined using equation (9), which is expected to approximate the accuracy of BLUP-grm. Theoretical prediction accuracy in panels (d-f) was defined using equation (11), which is expected to approximate the accuracy of BLUP-ldcov; it is expected to be exact when there is no between-block LD. The identical performance of BLUP-ldcov and BLUP-ldgm in AFR and EUR (panels a,b,d,e) indicates that the LDGM is accurate. It is not expected for BLUP-ldgm to perform better than BLUP-ldcov (Figure 5a) because BLUP-ldcov uses within-sample LD. The higher performance of BLUP-grm when the heritability is large results from the fact that conditioning on X explains a large fraction of variance in the residuals, resulting in increased effective sample size. The poor performance of BLUP-ldcov and especially BLUP-ldgm in AMR (panels c,f) reflects the presence of long-range correlations due to admixture and population structure. In panel (f), disagreement between theoretical and observed prediction accuracy for BLUP-ldcov (and BLUP-ldgm)

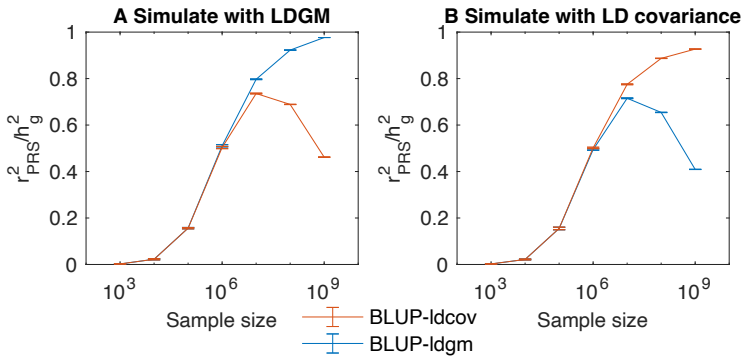
reflects the presence of between-block LD. Error bars denote 95% confidence intervals across the 10 replicates. For numerical results, see Supplementary Table 10.



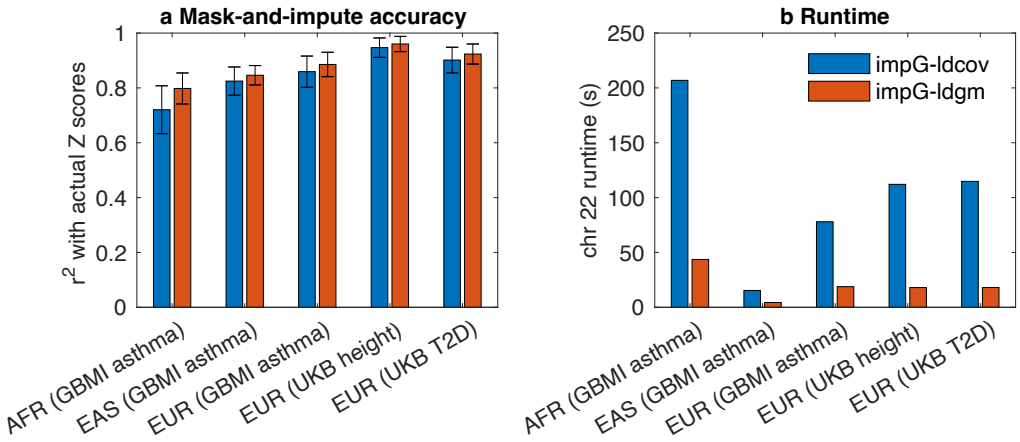
Supplementary Figure 3: performance of BLUP-ldgm with different precision matrix parameter settings. Summary statistics were simulated using individual level data (n=1006) for chromosome 22 (20 LD blocks), in EUR. We compared five parameter settings, varying the path-weight threshold T and the L1 penalty λ; the default parameter setting, used throughout the manuscript, is $T = 4, \lambda = 0.1$. We compared the within-sample prediction accuracy (equation 7) and the runtime. We report the average of 10 replicates, and error bars denote 95% confidence intervals. For numerical results, see Supplementary Table 11.



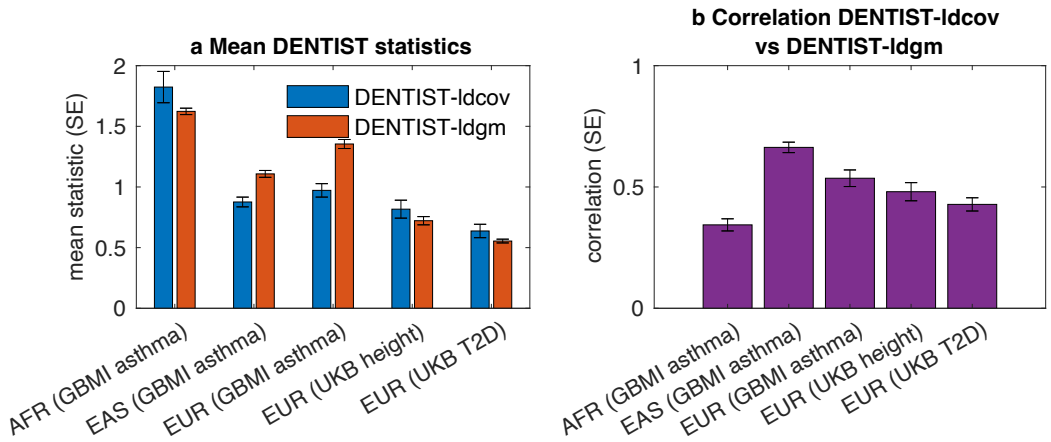
Supplementary Figure 4: performance of BLUP-ldgm in simulations with UK Biobank genotype data. We simulated individual level phenotypes using UK Biobank imputed genotypes for chromosome 1, for 25,000 unrelated British-ancestry individuals, under an infinitesimal genetic architecture with heritability 0.1. We used SNPs with minor allele frequency at least 1% in the 25,000 individuals. We designated 5,000 individuals as a validation set and used the remaining 20,000 for training, with a sample size between 1,000 and 20,000. We compared BLUP-ldgm, which operates on LDGM precision matrices and Z scores, and BLUP-grm, which operates on the genetic relatedness matrix and individual-level phenotypes. We report the average across 100 replicates, and error bars denote standard errors. For numerical results, see Supplementary Table 12.



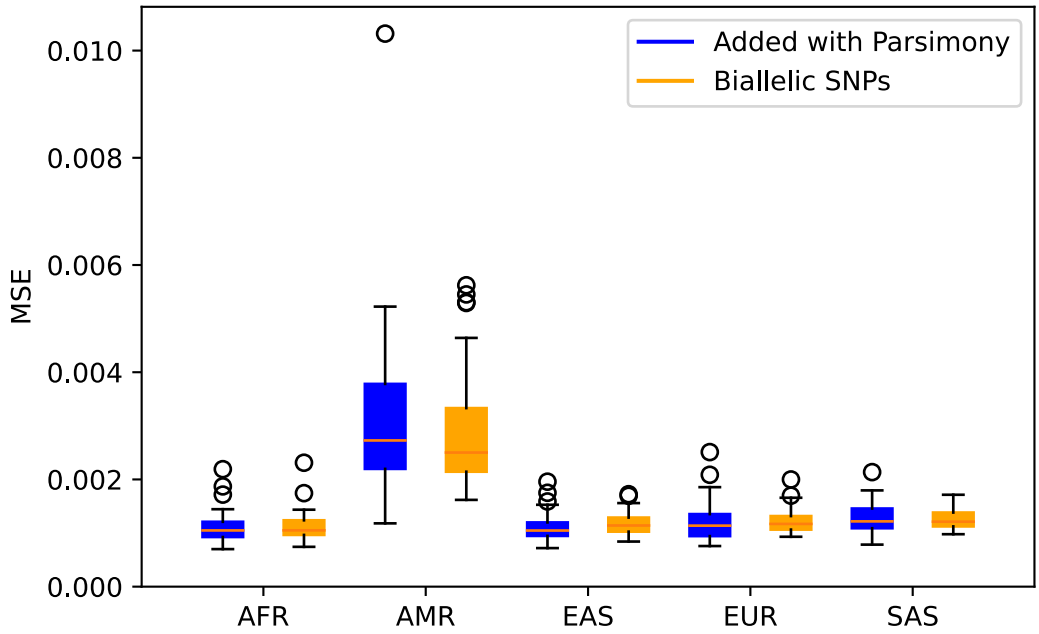
Supplementary Figure 5: comparison of BLUPx-ldgm and BLUPx-ldcov in simulations. We simulated summary association statistics from their asymptotic sampling distribution using either the LDGM precision matrix (a) or the LD covariance matrix (b), at different sample sizes, for the whole genome. BLUP-ldgm and BLUP-ldcov had similar performance (PRS r^2) at small to realistic sample sizes, but at very large sample sizes, better performance was obtained using the LD matrix that matched the way the data was simulated. We report the average across 100 replicates, and error bars denote standard errors.



Supplementary Figure 6: cross-validation analysis of linearly imputed summary statistics using impG-ldgm and impG-ldcov. For two UK Biobank traits (EUR ancestry) and one global biobank meta-analysis initiative trait (AFR, EAS and EUR ancestries), we held out 1/5 SNPs on chromosome 22, linearly imputed their Z scores, and compared the imputed vs. actual Z scores. (a) Mean cross-validation r^2, averaging across the five iterations and the 20 LD blocks, with standard errors. The difference between impG-ldcov and impG-ldgm was significant for four out of five trait-ancestry pairs (χ^2 test p < 0.05; EAS/asthma, p=0.05). (b) Total runtime, summing across the five iterations and the 20 LD blocks. For numerical results, see Supplementary Table 13.



Supplementary Figure 7: comparison of DENTIST-ldgm and DENTIST-ldcov. We implemented a simplified version of DENTIST (see Methods). For two UK Biobank traits (EUR ancestry) and one global biobank meta-analysis initiative trait (AFR, EAS and EUR ancestries), we applied both methods across chromosome 22 and calculated the mean DENTIST χ^2 statistic (a) and the correlation between the χ^2 statistics (b). The mean was reported after one round of discarding SNPs with large test statistics and re-computing the residuals; the correlation was computed without doing so, such that a correlation could be calculated without discarding additional SNPs. We report the mean and standard error across 20 LD blocks. For numerical results, see Supplementary Table 14.



Supplementary Figure 8: MSE of sites added using parsimony compared to biallelic SNPs. Following the inference approach outlined in the Methods section, 164,542 non-biallelic SNPs, sites with low-confidence or no ancestral states, and indels were added to the final LDGM precision matrices. We evaluated the MSE of the LDGM precision matrix vs sample correlation matrix restricted to these SNPs compared to the corresponding correlation matrix across chromosome 21 and 22. Each data point corresponds to an LD block. The lower whisker, lower hinge, center, upper hinge and upper whisker correspond to (lower hinge − 1.5× IQR) and the 25th percentile, median, 75th percentile, and (upper hinge + 1.5× IQR), respectively. For numerical results, see Supplementary Table 15.

8. Supplementary Table Captions

Supplementary table 1 (see Excel file): statistics for each LDGM precision matrix. For each ancestry group-LD block pair, we report the number of SNPs, the number of edges per SNP (degree) before and after L1-penalized precision matrix inference, the MSE, the alternative MSE, and the file size. We also list the parameters that were used for inference (the path distance threshold, the L1 penalty, and the recombination frequency threshold).

Supplementary table 2 (see Excel file): numerical results for Figure 3d.

Supplementary table 3 (see Excel file): numerical results for Figure 4a.

Supplementary table 4 (see Excel file): runtime per LD block to derive the LDGM and infer the LDGM precision matrix. Results are reported for chromosome 22 only because for most of the genome, we used an old version of the software that ran more slowly. For the LDGM inference step, we used 5 compute threads (1 for the precision matrix inference step). (a) Runtime for precision matrix inference for each LD block and each population; (b) runtime for LDGM inference for each LD block.

Supplementary table 5 (see Excel file): numerical results for Figure 5a. We also report the MSE between the UK Biobank LD correlation matrix and either the 1000 Genomes LDGM precision matrix or the 1000 Genomes LD correlation matrix. (a) Results for each LD block (chromosomes 21-22). We report the MSE between the UK Biobank correlation matrix and either the 1kg correlation matrix or the 1kg LDGM precision matrix. Similarly, we also report the r^2 between the BLUP PGS weights, using either the 1kg correlation matrix or the 1kg LDGM precision matrix, for each trait. (b) Averages and comparisons across LD blocks. For the MSE and the BLUP PGS weights r^2, p-values are reported for whether the 1kg precision matrix median is significantly better than the 1kg correlation matrix median, calculated using a single-tailed binomial test.

Supplementary Table 6 (see Excel file): Phenotypes summary association statistics analyzed. Summary statistics were calculated by Loh et al. (see Data Availability), who also computed heritability and effective sample size.

Supplementary table 7 (see Excel file): numerical results for Figure 5b.

Supplementary table 8 (see Excel file): numerical results for Figure 5c-d. We report the sample size in each population, the target population, and the prediction r^2. We also report cross-ancestry prediction accuracy at smaller sample sizes and for other population pairs.

Supplementary table 9 (see Excel file): numerical results for Supplementary Figure 1. We report the correlation between predicted and observed phenotypes for each chromosome and each trait.

Supplementary table 10 (see Excel file): numerical results for Supplementary Figure 2.

Supplementary table 11 (see Excel file): numerical results for Supplementary Figure 3.

Supplementary table 12 (see Excel file): numerical results for Supplementary Figure 4. We also report prediction r^2 at a larger heritability (0.5 vs. 0.1), and the runtime. For BLUP-grm, the runtime does not include the time to calculate the GRM.

Supplementary table 13 (see Excel file): numerical results for Supplementary Figure 6.

Supplementary table 14 (see Excel file): numerical results for Supplementary Figure 7.

Supplementary table 15 (see Excel file): numerical results for Extended Data Figure 9 and Supplementary Figure 8.

References

1. Scutari, M. & Strimmer, K. Introduction to Graphical Modelling. (2010).
2. Kelleher, J., Etheridge, A. M. & McVean, G. Efficient Coalescent Simulation and Genealogical Analysis for Large Sample Sizes. *PLoS Comput Biol* **12**, e1004842 (2016).
3. Kelleher, J., Thornton, K. R., Ashander, J. & Ralph, P. L. Efficient pedigree recording for fast population genetics simulation. *PLoS Comput Biol* **14**, (2018).
4. Kelleher, J. *et al.* Inferring whole-genome histories in large population datasets. *Nat Genet* **51**, 1330–1338 (2019).
5. Friedman, J., Hastie, T. & Tibshirani, R. Sparse inverse covariance estimation with the graphical lasso. *Biostatistics* **9**, 432–441 (2008).
6. Mazumder, R. & Hastie, T. The graphical lasso: New insights and alternatives. *Electron J Stat* **6**, 2125 (2012).
7. Daubechies, I., Defrise, M. & de Mol, C. An iterative thresholding algorithm for linear inverse problems with a sparsity constraint. *Commun Pure Appl Math* **57**, 1413–1457 (2004).
8. Shi, H., Kichaev, G. & Pasaniuc, B. Contrasting the genetic architecture of 30 complex traits from summary association data. *The American Journal of Human Genetics* **99**, 139–153 (2016).
9. Shi, H., Mancuso, N., Spendlove, S. & Pasaniuc, B. Local Genetic Correlation Gives Insights into the Shared Genetic Architecture of Complex Traits. *Am J Hum Genet* **101**, 737–751 (2017).
10. Ning, Z., Pawitan, Y. & Shen, X. High-definition likelihood inference of genetic correlations across human complex traits. *Nat Genet* **52**, (2020).
11. Spence, J. P., Sinnott-Armstrong, N., Assimes, T. L. & Pritchard, J. K. A flexible modeling and inference framework for estimating variant effect sizes from GWAS summary statistics. *bioRxiv* 2022.04.18.488696 (2022) doi:10.1101/2022.04.18.488696.
12. Davis, T. A. & Hager, W. W. Dynamic supernodes in sparse Cholesky update/downdate and triangular solves. *ACM Transactions on Mathematical Software (TOMS)* **35**, 1–23 (2009).

13. Davis, T. A. & Hager, W. CHOLMOD: supernodal sparse cholesky factorization and update/downdate. Preprint at (2005).
14. Conneely, K. N. & Boehnke, M. So many correlated tests, so little time! Rapid adjustment of P values for multiple correlated tests. *The American Journal of Human Genetics* **81**, 1158–1168 (2007).
15. Zhu, X. & Stephens, M. BAYESIAN LARGE-SCALE MULTIPLE REGRESSION WITH SUMMARY STATISTICS FROM GENOME-WIDE ASSOCIATION STUDIES 1. *Ann Appl Stat* **11**, 1561–1592 (2017).
16. Erisman, A. M. & Tinney, W. F. On computing certain elements of the inverse of a sparse matrix. *Commun ACM* **18**, 177–179 (1975).
17. Wray, N. R. *et al.* Pitfalls of predicting complex traits from SNPs. *Nat Rev Genet* **14**, 507–515 (2013).
18. Yang, J., Zaitlen, N. A., Goddard, M. E., Visscher, P. M. & Price, A. L. Advantages and pitfalls in the application of mixed-model association methods. *Nature Genetics* vol. 46 100–106 Preprint at https://doi.org/10.1038/ng.2876 (2014).
19. O'Connor, L. J. The distribution of common-variant effect sizes. *Nat Genet* **53**, 1243–1249 (2021).
20. Benner, C. *et al.* Prospects of fine-mapping trait-associated genomic regions by using summary statistics from genome-wide association studies. *The American Journal of Human Genetics* **101**, 539–551 (2017).
21. Pasaniuc, B. *et al.* Fast and accurate imputation of summary statistics enhances evidence of functional enrichment. *Bioinformatics* **30**, 2906–2914 (2014).
22. Wen, X. & Stephens, M. Using linear predictors to impute allele frequencies from summary or pooled genotype data. *Ann Appl Stat* **4**, 1158–1182 (2010).
23. Lee, D., Bigdeli, T. B., Riley, B. P., Fanous, A. H. & Bacanu, S.-A. DIST: direct imputation of summary statistics for unmeasured SNPs. *Bioinformatics* **29**, 2925–2927 (2013).
24. Bycroft, C. *et al.* The UK Biobank resource with deep phenotyping and genomic data. *Nature* **562**, 203–209 (2018).
25. Loh, P.-R., Kichaev, G., Gazal, S., Schoech, A. P. & Price, A. L. Mixed-model association for biobank-scale datasets. *Nat Genet* **50**, 906 (2018).
26. Zhou, W. *et al.* Global Biobank Meta-analysis Initiative: Powering genetic discovery across human disease. *Cell Genomics* **2**, 100192 (2022).
27. Chen, W. *et al.* Improved analyses of GWAS summary statistics by reducing data heterogeneity and errors. *Nat Commun* **12**, 7117 (2021).
28. Kanai, M. *et al.* Meta-analysis fine-mapping is often miscalibrated at single-variant resolution. *Cell Genomics* **2**, 100210 (2022).