permGWAS2: Enhanced and Accelerated Permutation-based Genome-Wide Association Studies

Maura John[1,2], Arthur Korte[3,4] and Dominik G. Grimm[1,2,5,*]

[1] *Technical University of Munich, Campus Straubing for Biotechnology and Sustainability, Bioinformatics, 94315 Straubing, Germany*
[2] *Weihenstephan-Triesdorf University of Applied Sciences, Bioinformatics, 94315 Straubing, Germany*
[3] *Center for Computational and Theoretical Biology, University of Würzburg, 97078 Würzburg, Germany*
[4] *Julius-von-Sachs-Institute, Plant Ecophysiology, University of Würzburg, 97082 Würzburg, Germany*
[5] *Technical University of Munich, TUM School of Computation, Information and Technology, 85748, Garching, Germany*

[*] Corresponding authors: dominik.grimm@hswt.de

Abstract

Motivation: Permutation-based significance thresholds have been shown to be a robust alternative to Bonferroni-based significance thresholds in genome-wide association studies (GWAS). However, the implementation of permutation-based thresholds is computationally demanding. The recently published method `permGWAS` introduced a batch-wise approach using 4D tensors to efficiently compute permutation-based GWAS. However, running multiple univariate tests in parallel leads to many repetitive computations and increased computational resources. More importantly, the previous version of `permGWAS` does not take into account the population structure when permuting the phenotype.
Results: We propose `permGWAS2`, an improved and accelerated version that uses a block matrix decomposition to optimize computations, thereby reducing redundant computations. It also introduces an alternative permutation strategy that takes into account the population structure during permutation. We show that this improved framework provides a more streamlined approach to performing permutation-based GWAS with a lower false discovery rate compared to the previous version and the traditional Bonferroni correction.
Availability: `permGWAS2` is open-source and publicly available on GitHub for download: `https://github.com/grimmlab/permGWAS`.

1 Introduction

Linear mixed models (LMMs) are a popular method for correcting for confounding factors, such as population structure and cryptic relatedness, when conducting genome-wide association studies (GWAS) (Lippert *et al.*, 2011; Loh *et al.*, 2015; Gumpinger *et al.*, 2018). Permutation-based significance thresholds provide a robust alternative to Bonferroni thresholds that can limit false-positive associations in statistical hypothesis tests (Che *et al.*, 2014; Nicod *et al.*, 2016; John *et al.*, 2022). One of the main challenges in using permutation-based approaches is the high computational cost, which was recently addressed in John *et al.* (2022). There we introduced `permGWAS`, a Python framework capable of computing efficient batch-wise LMMs using 3- and 4-dimensional tensors. This reformulation, combined with the *maxT* permutation method of Westfall and Young (1993), allowed us to compute permutation-based thresholds in a reasonable amount of time. However, running multiple univariate tests in parallel inevitably leads to increased memory usage. In addition, although `permGWAS` processes the SNPs in each batch simultaneously, it still performs each hypothesis test independently, repeating certain computations that could actually be saved and reused. In fact, when estimating the parameters of an LMM, certain calculations are independent of the marker of interest and can therefore be reused for each SNP. More importantly, `permGWAS` currently contains only a simple permutation strategy that does not take the population structure into account when permuting the phenotype.
Here we propose `permGWAS2`, an improved and accelerated version of `permGWAS` that takes advantage of a block matrix decomposition to reduce the number of redundant computations. In addition, our new framework now provides an alternative permutation approach that takes population structure into account.

2 Methods

In the following, we first review the basic framework of linear mixed models and how to estimate effect sizes and variance components, followed by two lemmas on how to efficiently compute permutation-based significance thresholds while taking into account the population structure during permutations.

2.1 Linear mixed models

For a vector of n phenotypic observations $y \in \mathbb{R}^n$, we consider a linear mixed model (LMM) of the form

$$y = X\beta + u + \epsilon, \tag{1}$$

where $X \in \mathbb{R}^{n \times c}$ is a matrix of fixed effects including a column of ones for the overall mean, the covariates, and the SNP of interest; $\beta \in \mathbb{R}^c$ denotes the effect sizes of the fixed effects; $u \in \mathbb{R}^n$ are random effects with $u \sim \mathcal{N}\left(0, \sigma_g^2 K\right)$, for the genetic variance component $\sigma_g^2 \in \mathbb{R}$ and the genetic similarity matrix $K \in \mathbb{R}^{n \times n}$; and $\epsilon \in \mathbb{R}^n$ is a vector of residual effects with $\epsilon \sim \mathcal{N}\left(0, \sigma_e^2 I\right)$ for residual variance component $\sigma_e^2 \in \mathbb{R}$ and

identity matrix $I \in \mathbb{R}^{n \times n}$. Then y also follows a Gaussian distribution with mean $X\beta$ and covariance matrix $V := \sigma_g^2 K + \sigma_e^2 I$.

Estimation of log likelihood

We estimate the effect sizes β, and the variance components σ_e^2 and σ_g^2 by maximizing the likelihood function

$$\ell\left(\beta, \sigma_g^2, \sigma_e^2\right) = \mathcal{N}\left(y \mid X\beta, \sigma_g^2 K + \sigma_e^2 I\right). \tag{2}$$

This is equivalent to finding the maximum of the log-likelihood function.

$$\begin{aligned} \ell\ell\left(\beta, \sigma_g^2, \sigma_e^2\right) &= \log \mathcal{N}\left(y \mid X\beta, \sigma_g^2 K + \sigma_e^2 I\right) \\ &= -\frac{1}{2}\left(n \log\left(2\pi\right) + \log|V| + \left(y - X\beta\right)^\top V^{-1}\left(y - X\beta\right)\right). \end{aligned} \tag{3}$$

Let $\delta := \frac{\sigma_e^2}{\sigma_g^2}$ and $H := K + \delta I$. Then Equation 3 can be simplified to

$$\begin{aligned} \ell\ell\left(\beta, \sigma_g^2, \delta\right) &= \log \mathcal{N}\left(y \mid X\beta, \sigma_g^2\left(K + \delta I\right)\right) \\ &= -\frac{1}{2}\left(n \log\left(2\pi\right) + n \log\left(\sigma_g^2\right) + \log|H| + \frac{1}{\sigma_g^2}\left(y - X\beta\right)^\top H^{-1}\left(y - X\beta\right)\right). \end{aligned} \tag{4}$$

Computing the derivative with respect to β and setting it equal to zero yields the generalized least squares estimate:

$$\widehat{\beta} = \left(X^\top H^{-1} X\right)^{-1} X^\top H^{-1} y \tag{5}$$

Substituting β in Equation 4 with $\widehat{\beta}$, computing the derivative with respect to σ_g^2 and setting it equal to zero yields

$$\widehat{\sigma}_{g\,ML}^2 = \frac{1}{n}\left(y - X\widehat{\beta}\right)^\top H^{-1}\left(y - X\widehat{\beta}\right) \tag{6}$$

Finally we can substitute β and σ_g^2 in Equation 4 with $\widehat{\beta}$ and $\widehat{\sigma}_{g\,ML}^2$, and the log likelihood becomes a function only dependent on δ:

$$\ell\ell\left(\widehat{\beta}, \widehat{\sigma}_{g\,ML}^2, \delta\right) = \ell\ell\left(\delta\right) = -\frac{1}{2}\left(n \log\left(2\pi\widehat{\sigma}_{g\,ML}^2\right) + \log|H| + n\right). \tag{7}$$

Efficient evaluation of log likelihood

Since the kinship matrix K is symmetric, we can compute the spectral decomposition $K = UDU^\top$, where U is an orthogonal matrix of eigenvectors of K, i.e., $UU^\top = I$, and D is a diagonal matrix containing the corresponding eigenvalues λ_i for $i \in \{1, \ldots, n\}$.

Then $H = U\left(D + \delta I\right)U^\top$ and $H^{-1} = U\left(D + \delta I\right)^{-1}U^\top$. Hence, we get

$$\widehat{\beta} = \left(\left(U^\top X\right)^\top \left(D + \delta I\right)^{-1} U^\top X\right)^{-1} \left(U^\top X\right)^\top \left(D + \delta I\right)^{-1} U^\top y \tag{8}$$

$$\widehat{\sigma}^2_{g\,ML} = \frac{1}{n}\left(U^\top y - U^\top X\widehat{\beta}\right)^\top \left(D + \delta I\right)^{-1}\left(U^\top y - U^\top X\widehat{\beta}\right) \tag{9}$$

and by using the fact that $|U| = 1$, Equation 7 changes to

$$\begin{aligned}\ell\ell\left(\delta\right) &= -\frac{1}{2}\left(n\log\left(2\pi\widehat{\sigma}^2_{g\,ML}\right) + \log\left|U\left(D + \delta I\right)U^\top\right| + n\right) \\ &= -\frac{1}{2}\left(n\log\left(2\pi\widehat{\sigma}^2_{g\,ML}\right) + \sum_{i=1}^{n}\log(\lambda_i + \delta) + n\right).\end{aligned} \tag{10}$$

Restricted maximum likelihood

We can extend Equation 4 to get the restricted maximum likelihood (REML):

$$\begin{aligned}\ell\ell_{REML}\left(\beta, \sigma^2_g, \delta\right) &= \ell\ell\left(\beta, \sigma^2_g, \delta\right) + \frac{1}{2}\left(c\log(2\pi) + \log\left|X^\top X\right| - \log\left|X^\top V^{-1} X\right|\right) \\ &= \ell\ell\left(\beta, \sigma^2_g, \delta\right) + \frac{1}{2}\left(c\log(2\pi\sigma^2_g) + \log\left|X^\top X\right| - \log\left|X^\top H^{-1} X\right|\right).\end{aligned} \tag{11}$$

Then the formula for $\widehat{\beta}$ remains unchanged and the variance component estimate is given by

$$\widehat{\sigma}^2_{g\,REML} = \frac{1}{n - c}\left(y - X\widehat{\beta}\right)^\top H^{-1}\left(y - X\widehat{\beta}\right). \tag{12}$$

Similarly to equation 7 we can put $\widehat{\beta}$ and $\widehat{\sigma}^2_{g\,REML}$ into Equation 11 and get

$$\begin{aligned}&\ell\ell_{REML}\left(\widehat{\beta}, \widehat{\sigma}^2_{g\,REML}, \delta\right) = \ell\ell_{REML}\left(\delta\right) \\ &= -\frac{1}{2}\left((n - c)\log\left(2\pi\widehat{\sigma}^2_{g\,REML}\right) + \log|H| + (n - c) - \log\left|X^\top X\right| + \log\left|X^\top H^{-1} X\right|\right).\end{aligned} \tag{13}$$

Using again the spectral decomposition of K, this can be efficiently reformulated to

$$-\frac{1}{2}\left((n - c)\log\left(2\pi\widehat{\sigma}^2_{g\,REML}\right) + \sum_{i=1}^{n}\log(\lambda_i + \delta) + (n - c) - \log\left|X^\top X\right| + \log\left|\left(U^\top X\right)^\top \left(D + \delta I\right)^{-1} U^\top X\right|\right), \tag{14}$$

with

$$\widehat{\sigma}^2_{g\,REML} = \frac{1}{n - c}\left(U^\top y - U^\top X\widehat{\beta}\right)^\top \left(D + \delta I\right)^{-1}\left(U^\top y - U^\top X\widehat{\beta}\right). \tag{15}$$

Hence, maximizing the log likelihood function 11 is equivalent to finding $\delta \in \mathbb{R}$ that maximizes Equation 14. In order to find the globally optimal δ we use an approach similar to Kang *et al.* (2008) and Lippert *et al.* (2011): First, we compute the function

values with respect to Equation 14 of 100 equidistant values between 10^{-5} and 10^{5} on a logarithmic scale. Then, for each triplet of neighboring points where the function value in the middle is greater than at the boundaries, we apply Brent's method to find the locally optimal δ. Finally, we take the optimal δ among all evaluated points. To speed up the computation, we estimate δ and σ_g^2 only once for a null model without any genetic markers, and reuse the estimates for the alternative models including the markers of interest as described in Kang *et al.* (2010). Once we have estimated the variance components, we use an F-test to test the null hypothesis that the marker of interest has no effect on a given phenotype. If the resulting p-value is below a predefined significance threshold, we reject the null hypothesis.
Because we test thousands to millions of markers at once in a typical GWAS, we need to correct for these multiple hypotheses to avoid thousands of false-positive associations. One way to do this is to empirically estimate the family-wise error rate, i.e., the probability of making at least one false positive, by computing a permutation-based significance threshold (John *et al.*, 2022).

2.2 Permutation-based significance thresholds

To compute a permutation-based significance threshold for a given significance level α, we use the *maxT* method of Westfall and Young (1993). For this, we first permute the phenotype y q-times and compute the test statistics ${}^k t_i$ for each permutation $k \in \{1, \dots, q\}$ and marker $i \in \{1, \dots, m\}$. Then for each permutation, we take the maximal test statistic over all markers ${}^k t_{\max} := \max_{i \in \{1,\dots,m\}} {}^k t_i$ which corresponds to the minimal p-value ${}^k \mathfrak{p}_{\min}$. Finally, the adjusted threshold is given as the α-th percentile of the minimal p-values. This permutation-based threshold is able to control the family-wise error rate, as shown in John *et al.* (2022).
However, the permutation strategy presented above does not take into account the underlying population structure of the given phenotype. In fact, permuting the phenotype breaks not only the correlation between the phenotypic and genotypic values, but also the relatedness between the individuals. To preserve this relatedness, one can alternatively compute the covariance matrix anew for each permutation or, equivalently, permute the rows and columns of the covariance matrix using the same permutation as for the phenotype vector. By Lemma 1 this is equivalent to permuting the fixed effects matrix containing the covariates and the SNP of interest.

Lemma 1. *Consider the LMM from the Equation 1, i.e., $y = X\beta + u + \epsilon$ with covariance matrix $V = \sigma_g^2 K + \sigma_e^2 I$. Let $\tau \colon \{1, \dots, n\} \to \{1, \dots, n\}$ be a permutation. Then permuting the entries of the phenotype vector y and the rows and columns of V with respect to τ is equivalent to permuting the rows of X with respect to the inverse permutation τ^{-1}.*

Proof. Let $P_\tau \in \mathbb{R}^{n \times n}$ denote the permutation matrix obtained by permuting the rows of the identity matrix $I \in \mathbb{R}^{n \times n}$ according to τ. Note that permutation matrices are orthogonal, i.e., $P_\tau P_\tau^\top = I$, and that $P_\tau^\top = P_\tau^{-1} = P_{\tau^{-1}}$ is the permutation matrix of the inverse permutation τ^{-1}. Then multiplying a matrix A with P_τ from left permutes the

rows of A and multiplying it from right with $P_\tau^\top$ permutes the columns of A according to τ.
Now consider the LMM $P_\tau y = X\beta + u + \epsilon$ with covariance matrix $P_\tau V P_\tau^\top$. Plugging it in Equation 3, the log likelihood function changes to

$$\begin{aligned}
\ell\ell\left(\beta, \sigma_g^2, \sigma_e^2\right) &= \log \mathcal{N}\left(P_\tau y \mid X\beta, P_\tau V P_\tau^\top\right) \\
&= -\frac{1}{2}\left(n \log\left(2\pi\right) + \log\left|P_\tau V P_\tau^\top\right| + \left(P_\tau y - X\beta\right)^\top \left(P_\tau V P_\tau^\top\right)^{-1} \left(P_\tau y - X\beta\right)\right) \\
&= -\frac{1}{2}\left(n \log\left(2\pi\right) + \log|V| + \left(P_\tau y - X\beta\right)^\top P_\tau V^{-1} P_\tau^\top \left(P_\tau y - X\beta\right)\right) \\
&= -\frac{1}{2}\left(n \log\left(2\pi\right) + \log|V| + \left(y - P_\tau^\top X\beta\right)^\top V^{-1} \left(y - P_\tau^\top X\beta\right)\right) \\
&= \log \mathcal{N}\left(y \mid P_\tau^\top X\beta, V\right),
\end{aligned}$$

which is the log likelihood function of the LMM $y = P_\tau^\top X\beta + u + \epsilon$ with covariance matrix V. Similarly, the restricted log likelihood changes to

$$\begin{aligned}
\ell\ell_{REML}\left(\beta, \sigma_g^2, \sigma_e^2\right) &= \ell\ell\left(\beta, \sigma_g^2, \sigma_e^2\right) + \frac{1}{2}\left(c \log\left(2\pi\right) + \log\left|X^\top X\right| - \log\left|X^\top \left(P_\tau V P_\tau^\top\right)^{-1} X\right|\right) \\
&= \ell\ell\left(\beta, \sigma_g^2, \sigma_e^2\right) + \frac{1}{2}\left(c \log\left(2\pi\right) + \log\left|X^\top P_\tau P_\tau^\top X\right| - \log\left|X^\top P_\tau V^{-1} P_\tau^\top X\right|\right) \\
&= \ell\ell\left(\beta, \sigma_g^2, \sigma_e^2\right) + \frac{1}{2}\left(c \log\left(2\pi\right) + \log\left|\left(P_\tau^\top X\right)^\top P_\tau^\top X\right| - \log\left|\left(P_\tau^\top X\right)^\top V^{-1} P_\tau^\top X\right|\right).
\end{aligned}$$

□

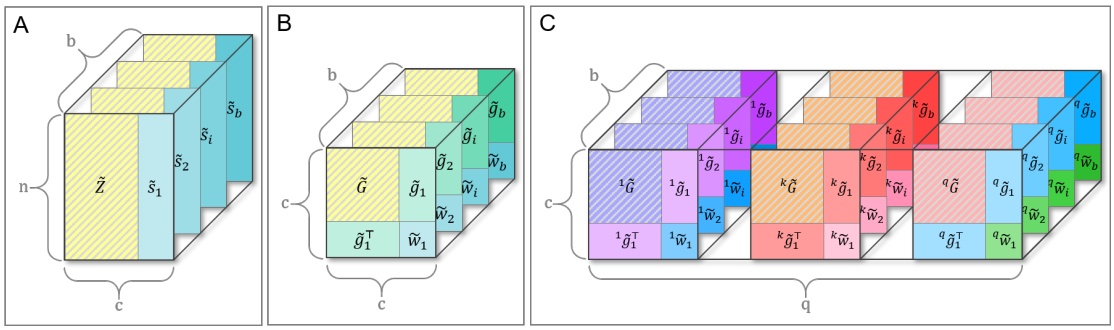

Figure 1: Schematic visualization of tensors and block matrices of the permGWAS2 architecture. Note that shaded areas are the same in each layer within a 3D tensor. (A) 3D tensor $\widetilde{X}_{(1:b)} \in \mathbb{R}^{b \times n \times c}$ containing the fixed effects matrices $\widetilde{X}_i = \left[\widetilde{Z}, \widetilde{s}_i\right] \in \mathbb{R}^{n \times c}$ with SNPs s_i for $i \in \{1, \ldots, b\}$. (B) 3D representation of block matrix structure with $\widetilde{G} := \widetilde{Z}^\top E^{-1} \widetilde{Z}$, $\widetilde{g}_i := \widetilde{Z}^\top E^{-1} \widetilde{s}_i$ and $\widetilde{w}_i := \widetilde{s}_i^\top E^{-1} \widetilde{s}_i$ for $i \in \{1, \ldots, b\}$. (C) 4D representation of permutation-based block matrix structure with ${}^k\widetilde{G} := \widetilde{Z}^\top {}^k E^{-1} \widetilde{Z}$, ${}^k\widetilde{g}_i := \widetilde{Z}^\top {}^k E^{-1} \widetilde{s}_i$ and ${}^k\widetilde{w}_i := \widetilde{s}_i^\top {}^k E^{-1} \widetilde{s}_i$ for $i \in \{1, \ldots, b\}$ an $k \in \{1, \ldots, q\}$.

2.3 permGWAS2 architecture

Both of the above permutation strategies are computationally expensive and thus mostly infeasible in practice. To accelerate permutation-based approaches while efficiently computing univariate statistical tests, John *et al.* (2022) introduced batch-wise LMMs. Although the originally introduced permGWAS outperforms commonly used LMMs such as EMMAX (Kang *et al.*, 2010) and FaST-LMM (Lippert *et al.*, 2011) in terms of computational and statistical power, it still performs many redundant computations. In addition, batch-wise computations lead to an increased memory footprint compared to computing single univariate tests one at a time.
In the following, we first review the mathematical framework for batch-wise LMMs based on John *et al.* (2022), and then show how an elegant block matrix decomposition can be used to compute LMMs more efficiently.

Batch-wise linear mixed models

Let b be the number of genetic markers to test in parallel. Let $Z \in \mathbb{R}^{n\times(c-1)}$ be the matrix containing a column of ones for the intercept and all covariates, and let $s_i \in \mathbb{R}^n$ be the i-th SNP for $i \in \{1,\dots,b\}$. Let $X_i := [Z, s_i] \in \mathbb{R}^{n\times c}$ be the matrix of fixed effects consisting of Z and the SNP s_i.
Let $E := D + \delta I$. Given the spectral decomposition $H = U\,(D+\delta I)\,U^\top = UEU^\top$, let $\widetilde{X}_i := U^\top X_i$ and $\widetilde{y} := U^\top y$. Denote by $\widetilde{X}_{(1:b)}$ the 3-dimensional tensor in $\mathbb{R}^{b\times n\times c}$ containing the matrices $\widetilde{X}_1$ to $\widetilde{X}_b$ (see Figure 1 (A)). Further, for each matrix $A \in \mathbb{R}^{n\times d}$ denote by $A_{(b)}$ the 3-dimensional tensor in $\mathbb{R}^{b\times n\times d}$ obtained by stacking b copies of A. Then the effect sizes and the residual sums of squares for all SNPs can be computed simultaneously via

$$\beta_{(1:b)} = \left(\widetilde{X}_{(1:b)}^\top E_{(b)}^{-1} \widetilde{X}_{(1:b)}\right)^{-1} \widetilde{X}_{(1:b)}^\top E_{(b)}^{-1} \widetilde{y}_{(b)} \tag{16}$$

$$\mathrm{RSS}_{(1:b)} = \left(\widetilde{y}_{(b)} - \widetilde{X}_{(1:b)}\beta_{(1:b)}\right)^\top E_{(b)}^{-1} \left(\widetilde{y}_{(b)} - \widetilde{X}_{(1:b)}\beta_{(1:b)}\right) \tag{17}$$

Now let q be the number of permutations. First, consider the original permGWAS permutation strategy (John *et al.*, 2022), where we only permute the phenotype. Here, for each permutation ${}^k y$ of y with $k \in \{1,\dots,q\}$ let ${}^k\widetilde{y} := U^\top {}^k y$ and let ${}^k\widetilde{y}_{(b)}$ denote the 3-dimensional tensor with b copies of ${}^k\widetilde{y}$. By stacking ${}^k\widetilde{y}_{(b)}$ for all $k \in \{1,\dots,q\}$, we get a 4-dimensional tensor ${}^{(1:q)}\widetilde{y}_{(b)} \in \mathbb{R}^{q\times b\times n\times 1}$. Similarly denote by ${}^{(1:q)}E_{(b)}$ the 4-dimensional tensor in $\mathbb{R}^{q\times b\times n\times n}$ containing the 3-dimensional tensors ${}^k E_{(b)} = \left(D + {}^k\delta I\right)_{(b)}$ for all $k \in \{1,\dots,q\}$. Further, for each 3-dimensional tensor $A \in \mathbb{R}^{b\times n\times d}$ denote by ${}^{(q)}A$ the 4-dimensional tensor in $\mathbb{R}^{q\times b\times n\times d}$ obtained by stacking q copies of A. Then the effect sizes and the residual sums of squares for b SNPs and q permutations can be computed

simultaneously via

$$ {}^{(1:q)}\beta_{(1:b)} = \left({}^{(q)}\widetilde{X}_{(1:b)}^{\top} \, {}^{(1:q)}E_{(b)}^{-1} \, {}^{(q)}\widetilde{X}_{(1:b)} \right)^{-1} {}^{(q)}\widetilde{X}_{(1:b)}^{\top} \, {}^{(1:q)}E_{(b)}^{-1} \, {}^{(1:q)}\widetilde{y}_{(b)} \tag{18} $$

$$ {}^{(1:q)}\mathrm{RSS}_{(1:b)} = \left({}^{(1:q)}\widetilde{y}_{(b)} - {}^{(q)}\widetilde{X}_{(1:b)} \, {}^{(1:q)}\beta_{(1:b)} \right)^{\top} {}^{(1:q)}E_{(b)}^{-1} \left({}^{(1:q)}\widetilde{y}_{(b)} - {}^{(q)}\widetilde{X}_{(1:b)} \, {}^{(1:q)}\beta_{(1:b)} \right) \tag{19} $$

For the second permutation strategy, where both the phenotype vector and the covariance matrix are permuted, ${}^{(1:q)}\beta_{(1:b)}$ and ${}^{(1:q)}\mathrm{RSS}_{(1:b)}$ can be computed similarly to the equations 18 and 19, except that instead of ${}^{k}y$ we compute permutations ${}^{k}X_i$ of X_i:

$$ {}^{(1:q)}\beta_{(1:b)} = \left({}^{(1:q)}\widetilde{X}_{(1:b)}^{\top} \, {}^{(1:q)}E_{(b)}^{-1} \, {}^{(1:q)}\widetilde{X}_{(1:b)} \right)^{-1} {}^{(1:q)}\widetilde{X}_{(1:b)}^{\top} \, {}^{(1:q)}E_{(b)}^{-1} \, {}^{(q)}\widetilde{y}_{(b)} \tag{20} $$

$$ {}^{(1:q)}\mathrm{RSS}_{(1:b)} = \left({}^{(q)}\widetilde{y}_{(b)} - {}^{(1:q)}\widetilde{X}_{(1:b)} \, {}^{(1:q)}\beta_{(1:b)} \right)^{\top} {}^{(1:q)}E_{(b)}^{-1} \left({}^{(q)}\widetilde{y}_{(b)} - {}^{(1:q)}\widetilde{X}_{(1:b)} \, {}^{(1:q)}\beta_{(1:b)} \right) \tag{21} $$

Efficient block matrix decomposition

In the following, we introduce a more efficient way to compute batch-wise LMMs and permutations, using an elegant block matrix decomposition that is applicable to both permutation strategies:

Lemma 2. *Let $Z \in \mathbb{R}^{n\times(c-1)}$, $s \in \mathbb{R}^n$, and let $A \in \mathbb{R}^{n\times n}$ be a symmetric matrix. Denote by $X := [Z, s] \in \mathbb{R}^{n\times c}$ the matrix obtained by concatenating the vector s to the matrix Z. Then the matrix $X^\top AX$ has the following block structure*

$$ X^\top EX = \begin{pmatrix} Z^\top AZ & Z^\top As \\ \left(Z^\top As\right)^\top & s^\top As \end{pmatrix}. \tag{22} $$

Proof. Let $Z = (z_{ij})_{\substack{i=1,\dots,n \\ j=1,\dots,c-1}}$, $A = (a_{it})_{\substack{i=1,\dots,n \\ t=1,\dots,n}}$, $s = (s_i)_{i=1,\dots,n}$ and $X = (x_{ik})_{\substack{i=1,\dots,n \\ k=1,\dots,c}}$. Then

$$ x_{ik} = \begin{cases} z_{ij} & \text{if } k = j \in \{1, \dots, c-1\} \\ s_i & \text{if } k = c \end{cases} $$

Denote the transposed matrices by $X^\top = (x'_{ki})_{\substack{k=1,\dots,c \\ i=1,\dots,n}}$ and $Z^\top = \left(z'_{ji}\right)_{\substack{j=1,\dots,c-1 \\ i=1,\dots,n}}$, where $x'_{ki} = x_{ik}$ and $z'_{ji} = z_{ij}$. It follows that

$$ X^\top AX = \left(x'_{li}\right)_{\substack{l=1,\dots,c \\ i=1,\dots,n}} \cdot (a_{it})_{\substack{i=1,\dots,n \\ t=1,\dots,n}} \cdot (x_{tk})_{\substack{t=1,\dots,n \\ k=1,\dots,c}} = \left(\sum_{t=1}^{n} \left(\sum_{i=1}^{n} x'_{li} a_{it} \right) x_{tk} \right)_{\substack{l=1,\dots,c \\ k=1,\dots,c}} \in \mathbb{R}^{c\times c} $$

with

$$\sum_{t=1}^{n}\left(\sum_{i=1}^{n} x'_{li}a_{it}\right)x_{tk} = \begin{cases} \sum_{t=1}^{n}\left(\sum_{i=1}^{n} z'_{li}a_{it}\right)z_{tk} & \text{if } l,k \in \{1,\dots,c-1\} \\ \sum_{t=1}^{n}\left(\sum_{i=1}^{n} z'_{li}a_{it}\right)s_{t} & \text{if } l \in \{1,\dots,c-1\} \text{ and } k = c \\ \sum_{t=1}^{n}\left(\sum_{i=1}^{n} s_{i}a_{it}\right)z_{tk} & \text{if } k \in \{1,\dots,c-1\} \text{ and } l = c \\ \sum_{t=1}^{n}\left(\sum_{i=1}^{n} s_{i}a_{it}\right)s_{t} & \text{if } l = k = c \end{cases}$$

Since $Z^\top AZ = \left(\sum_{t=1}^{n}\left(\sum_{i=1}^{n} z'_{hi}a_{it}\right)z_{tj}\right)_{\substack{h=1,\dots,c-1\\ j=1,\dots,c-1}} \in \mathbb{R}^{(c-1)\times(c-1)}$, $s^\top As = \sum_{t=1}^{n}\left(\sum_{i=1}^{n} s_i a_{it}\right)s_t \in \mathbb{R}$, $Z^\top As = \left(\sum_{t=1}^{n}\left(\sum_{i=1}^{n} z'_{ji}a_{it}\right)s_t\right)_{j=1,\dots,c-1} \in \mathbb{R}^{(c-1)\times 1}$, and $s^\top AZ = \left(Z^\top As\right)^\top \in \mathbb{R}^{1\times(c-1)}$, the claim follows.

□

Now let $\widetilde{Z} := U^\top Z$ and $\widetilde{s}_i := U^\top s_i$ for all $i \in \{1,\dots,b\}$. Note that $U^\top X_i = \left[U^\top Z, U^\top s_i\right] = \left[\widetilde{Z}, \widetilde{s}_i\right]$. Denote by $\widetilde{S}_{(1:b)} \in \mathbb{R}^{b\times n\times 1}$ the 3-dimensional tensor containing the batch of b SNPs. Then by Lemma 2, we can decompose $\widetilde{X}_{(1:b)}^\top E_{(b)}^{-1}\widetilde{X}_{(1:b)} \in \mathbb{R}^{b\times c\times c}$ into the following block structure

$$\widetilde{X}_{(1:b)}^\top E_{(b)}^{-1}\widetilde{X}_{(1:b)} = \begin{pmatrix} \left(\widetilde{Z}^\top E^{-1}\widetilde{Z}\right)_{(b)} & \left(\widetilde{Z}^\top E^{-1}\right)_{(b)}\widetilde{S}_{(1:b)} \\ \left(\left(\widetilde{Z}^\top E^{-1}\right)_{(b)}\widetilde{S}_{(1:b)}\right)^\top & \widetilde{S}_{(1:b)}^\top E_{(b)}^{-1}\widetilde{S}_{(1:b)} \end{pmatrix}. \tag{23}$$

Since $\widetilde{Z}^\top E^{-1}\widetilde{Z}$ is independent of the SNPs, we don't need to recompute it for each SNP. Hence, we only need to compute $\left(\widetilde{Z}^\top E^{-1}\right)_{(b)}\widetilde{S}_{(1:b)}$ and $\widetilde{S}_{(1:b)}^\top E_{(b)}^{-1}\widetilde{S}_{(1:b)}$ in batches and can assemble the block matrix afterward (see Figure 1 (B)). Similarly, one can show that

$$\widetilde{X}_{(1:b)}^\top E_{(b)}^{-1}\widetilde{y}_{(b)} = \begin{pmatrix} \left(\widetilde{Z}^\top E^{-1}\widetilde{y}\right)_{(b)} \\ \widetilde{S}_{(1:b)}^\top E_{(b)}^{-1}\widetilde{y}_{(b)} \end{pmatrix}, \tag{24}$$

where $\widetilde{Z}^\top E^{-1}\widetilde{y}$ is independent of the SNPs. Again, only $\widetilde{S}_{(1:b)}^\top E_{(b)}^{-1}\widetilde{y}_{(b)}$ needs to be computed in batches.

Now when performing GWAS with permutations, we can decompose ${}^k\widetilde{X}_{(1:b)}^\top {}^kE_{(b)}^{-1}\widetilde{X}_{(1:b)}$ and ${}^k\widetilde{X}_{(1:b)}^\top {}^kE_{(b)}^{-1}\,{}^k\widetilde{y}_{(b)}$ as in Equations 23 and 24 for each k. Since ${}^k\widetilde{Z}^\top {}^kE^{-1}\widetilde{Z}$ and ${}^k\widetilde{Z}^\top {}^kE^{-1}\,{}^k\widetilde{y}$ only depend on the permutation and not on the SNPs, we only need to compute them once per permutation (see Figure 1 (C)).

We call the new `permGWAS` version using this block matrix decomposition `permGWAS2`. Per default, `permGWAS2` uses the permutation strategy, where we permute both the phenotype and the covariance matrix. In the following, we refer to `permGWAS2` with the old permutation strategy, where we permute only the phenotype vector, as `permGWAS2(y)`.

2.4 Implementation & Availability

permGWAS2 is implemented in Python3 as a standalone command-line tool. It uses PyTorch in addition to common scientific computing libraries such as numpy, pandas, and scipy to support multi-core and GPU usage as well as efficient tensor arithmetic. We support several common genotype and phenotype file formats, including PLINK, CSV, and HDF5. For the genetic similarity matrix, the user can either provide a pre-computed matrix or use the implemented realized relationship kernel, which is computed by permGWAS2 by default. In addition, the user can specify covariates to account for certain fixed effects. To estimate the variance components, permGWAS2 includes a custom optimization function using Brent's method. For permutations, the user can choose between two strategies, the default strategy of permuting the phenotype vector and the covariance matrix (called permGWAS2), and the old strategy of only permuting the phenotype y (called permGWAS2(y)). To simplify the workflow, permGWAS2 supports the use of YAML configuration files. Our framework also includes functions to visualize p-values as Manhattan or QQ-plots. It is also possible to include models other than LMMs within the permGWAS2 framework. Our code is open source and publicly available on GitHub: https://github.com/grimmlab/permGWAS.

3 Results & Discussion

To compare the performance of permGWAS2 with block matrix decomposition with the original permGWAS version (John *et al.*, 2022), we performed several runtime experiments. Additionally, we compared the runtime of both versions with the two commonly used LMMs EMMAX (Kang *et al.*, 2010) and FaST-LMM. (Lippert *et al.*, 2011), for which we used the binary C/C++ implementations. The runtime comparisons were performed on a machine running Ubuntu 22.04.2 LTS with a total of 52 CPUs, 756 GB of memory, and 4 NVIDIA GeForce RTX 3090 GPUs with 24 GB of memory each. For our experiments, we used dedicated Docker containers where we limited the number of CPUs to one core and used a single GPU. We analyzed the runtime in terms of the number of markers, the number of samples, and the number of permutations. To do this, we generated synthetic data with varying numbers of samples and markers by up- and down-sampling a flowering time related phenotype from *Arabidopsis thaliana* and the corresponding genotype matrix (The 1001 Genomes Consortium, 2016). For each experiment, we took the average of three runs.

To evaluate the effect of increasing the number of markers, we first fixed the number of samples at 1000 and varied the number of SNPs between 10^4 and 5×10^6. As shown in Figure 2 (A), both permGWAS and permGWAS2 show similar performance with about 14 min for permGWAS2 and 17 min for permGWAS for 5 million SNPs, visibly outperforming EMMAX and FaST-LMM. In the second experiment, we fixed the number of markers at 10^6 and varied the number of individuals between 100 and 10^4 to analyze how the runtime depends on the number of samples. Again, permGWAS2 slightly outperforms permGWAS with about 55 min compared to 67 min for 10k samples, both at least an order of magnitude faster than the comparison partners as summarized in Figure 2 (B).

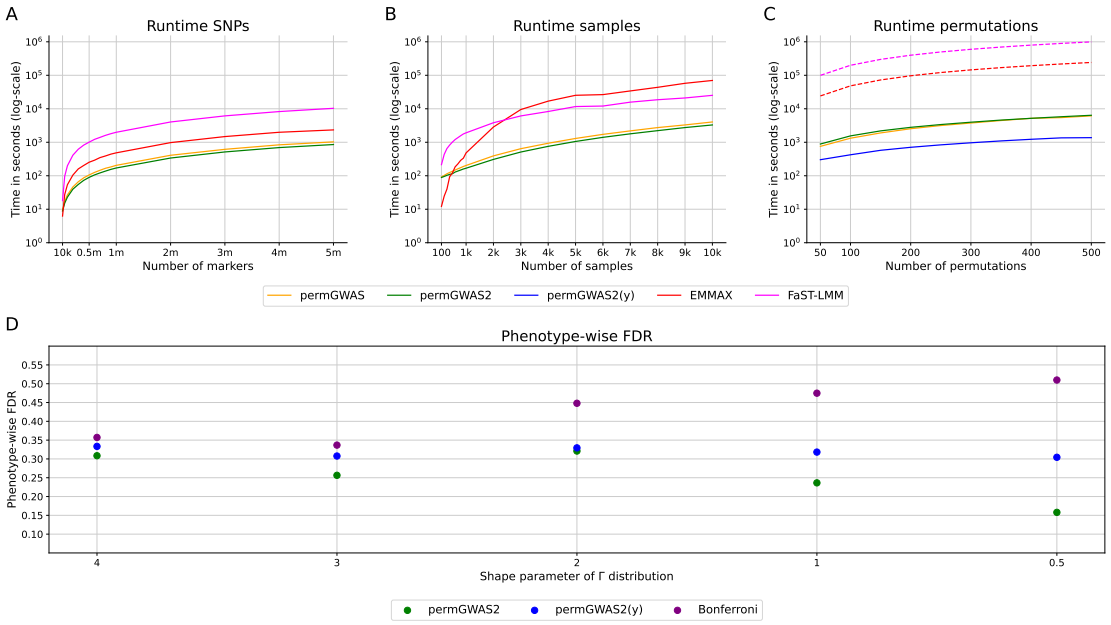

Figure 2: Performance analysis of `permGWAS2`. (A) - (C) Runtime comparisons of `permGWAS2` vs. `permGWAS`, `EMMAX` and `FaST-LMM`. Note all axes are log-scaled. (A) Computational time as function of number of SNPs with fixed number of 1 000 samples. (B) Computational time as function of number of samples with 10^6 marker each. (C) Computational time as function of number of permutations with 1 000 samples and 10^6 marker each. Dashed lines for `EMMAX` and `FaST-LMM` are estimated based on the computational time for 1 000 samples and 10^6 markers times the number of permutations.`permGWAS` and `permGWAS2` were run on a single GPU. `permGWAS2(y)` refers to `permGWAS2` with the same permutation strategy as in the previous `permGWAS` version where only the phenotype vector is permuted. (D) Phenotype-wise FDR for Bonferroni and permutation-based significance threshold with `permGWAS2` and `permGWAS2(y)`.

Finally, we again set the number of samples to 1000 and the number of markers to 10^6, and ran between 10 and 500 permutations. We compared not only `permGWAS` with `permGWAS2`, but also with the old permutation strategy `permGWAS2(y)`, which only permutes the phenotype vector. Since `EMMAX` and `FaST-LMM` do not support permutations, we took the runtime from the previous experiments and approximated the permutation-based time by multiplying it by the number of permutations. However, this is only an estimate of the runtime without taking into account the time needed for additional pre- and post-processing steps to prepare the permutations. For 500 permutations the performance of `permGWAS2` is similar to that of `permGWAS` with 107 min, and 102 min, respectively (see Figure 2 (C)). This is remarkable considering that `permGWAS2` permutes the fixed effects matrix for all markers instead of only permuting the phenotype vector. If we use this simpler permutation strategy with `permGWAS2(y)`, 500 permutations take around 23 min, so less than a quarter than the original `permGWAS`. In comparison, the

approximated runtimes of `EMMAX` and `FaST-LMM` are close to 3 days and more than 11 days, respectively.

To compare the two permutation methods `permGWAS2` and `permGWAS2(y)`, we simulated artificial phenotypes for 200 random *A. thaliana* individuals based on fully imputed genotype data with approximately 2.8 million SNPs (Arouisse *et al.*, 2020). We first computed the realized relationship kernel as a kinship matrix and its Cholesky decomposition $K = CC^\top$. To simulate the polygenic background we chose a random vector $u \in \mathbb{R}^n$ where each element was drawn from a Gaussian distribution with zero mean and a variance of 1 and multiplied it with C such that $\mathrm{Var}(Cu) = K$. Next, we added random noise drawn from a gamma distribution such that the noise contributed 70% to the total phenotypic variance. Finally, we added a causal SNP with a minor allele frequency greater than 5% and an effect size that explained about 20% of the phenotypic variance. To simulate differently skewed phenotypes, we used shape parameters of 0.5, 1, 2, 3 and 4 for the gamma distribution resulting in 5 different simulation settings. For each setting, we generated 100 simulations and ran `permGWAS2` and `permGWAS2(y)` with 500 permutations each. Since linkage disequilibrium decays on average within 10 kbp in *A. thaliana* (Kim *et al.*, 2007), we defined a phenotype as a true positive (TP) if any marker within a 10 kbp window around the causal SNP was deemed significant. If a marker outside this window was significant, we classified the phenotype as a false positive (FP). Thus, a phenotype could be both TP and FP at the same time. We then calculated for each of the five simulation settings the phenotype-wise false discovery rate $\mathrm{FDR} := \frac{\mathrm{FP}}{\mathrm{TP}+\mathrm{FP}}$ for both permutation-based and Bonferroni threshold.

Our experiments show that the phenotype-wise FDR for `permGWAS2(y)` is more or less stable over all simulation settings (see Figure 2 (D)). In contrast, the FDR for `permGWAS2` decreases for smaller shape parameters, i.e., when the phenotypes become more skewed. Thus, by taking the population structure into account when permuting, we are able to better control the phenotype-wise FDR. Remarkably, the FDR with Bonferroni threshold increases up to 50% for the most skewed phenotypes, meaning that we find as many TP as FP associations.

4 Conclusions

We introduced `permGWAS2`, an improved and accelerated modification of `permGWAS` (John *et al.*, 2022). By employing a block matrix decomposition, `permGWAS2` optimizes computations and significantly reduces redundancy. In particular, our framework incorporates an improved permutation strategy that accounts for population structure during permutation. Our results demonstrate that `permGWAS2` provides a more streamlined approach with a lower false discovery rate compared to its predecessor and the traditional Bonferroni correction. This advancement represents a significant refinement in computational efficiency and statistical robustness for GWAS.

Funding

The project is supported by funds of the Federal Ministry of Education and Research (BMBF), Germany [number 01|S21038B, D.G.G.].

Conflict of Interest: none declared.

References

The 1001 Genomes Consortium (2016). 1,135 genomes reveal the global pattern of polymorphism in arabidopsis thaliana. *Cell*, **166**(2), 481–491.

Arouisse, B., Korte, A., van Eeuwijk, F., and Kruijer, W. (2020). Imputation of 3 million snps in the arabidopsis regional mapping population. *The Plant Journal*, **102**(4), 872–882.

Che, R., Jack, J. R., Motsinger-Reif, A. A., and Brown, C. C. (2014). An adaptive permutation approach for genome-wide association study: evaluation and recommendations for use. *BioData mining*, **7**(1), 9.

Gumpinger, A. C., Roqueiro, D., Grimm, D. G., and Borgwardt, K. M. (2018). Methods and tools in genome-wide association studies. *Computational Cell Biology: Methods and Protocols*, pages 93–136.

John, M., Ankenbrand, M. J., Artmann, C., Freudenthal, J. A., Korte, A., and Grimm, D. G. (2022). Efficient permutation-based genome-wide association studies for normal and skewed phenotypic distributions. *Bioinformatics*, **38**(Supplement_2), ii5–ii12.

Kang, H. M., Zaitlen, N. A., Wade, C. M., Kirby, A., Heckerman, D., Daly, M. J., and Eskin, E. (2008). Efficient control of population structure in model organism association mapping. *Genetics*, **178**(3), 1709–1723.

Kang, H. M., Sul, J. H., Service, S. K., Zaitlen, N. A., Kong, S.-y., Freimer, N. B., Sabatti, C., and Eskin, E. (2010). Variance component model to account for sample structure in genome-wide association studies. *Nature genetics*, **42**(4), 348–354.

Kim, S., Plagnol, V., Hu, T. T., Toomajian, C., Clark, R. M., Ossowski, S., Ecker, J. R., Weigel, D., and Nordborg, M. (2007). Recombination and linkage disequilibrium in arabidopsis thaliana. *Nature genetics*, **39**(9), 1151–1155.

Lippert, C., Listgarten, J., Liu, Y., Kadie, C. M., Davidson, R. I., and Heckerman, D. (2011). Fast linear mixed models for genome-wide association studies. *Nature methods*, **8**(10), 833–835.

Loh, P.-R., Tucker, G., Bulik-Sullivan, B. K., Vilhjalmsson, B. J., *et al.* (2015). Efficient bayesian mixed-model analysis increases association power in large cohorts. *Nature genetics*, **47**(3), 284–290.

Nicod, J., Davies, R. W., Cai, N., Hassett, C., Goodstadt, L., Cosgrove, C., Yee, B. K., Lionikaite, V., McIntyre, R. E., Remme, C. A., Lodder, E. M., Gregory, J. S., Hough, T., Joynson, R., Phelps, H., Nell, B., Rowe, C., Wood, J., Walling, A., Bopp, N., Bhomra, A., Hernandez-Pliego, P., Callebert, J., Aspden, R. M., Talbot, N. P., Robbins, P. A., Harrison, M., Fray, M., Launay, J.-M., Pinto, Y. M., Blizard, D. A., Bezzina, C. R., Adams, D. J., Franken, P., Weaver, T., Wells, S., Brown, S. D. M., Potter, P. K., Klenerman, P., Lionikas, A., Mott, R., and Flint, J. (2016). Genome-wide association of multiple complex traits in outbred mice by ultra-low-coverage sequencing. *Nature genetics*, **48**(8), 912–918.

Westfall, P. H. and Young, S. S. (1993). *Resampling-based multiple testing: Examples and methods for p-value adjustment*, volume 279. John Wiley & Sons.