Supplementary Methods for "Fast Effect Size Shrinkage Software for Beta-Binomial Models of Allelic Imbalance"

Model Setup

For the g-th gene ($1 \leq g \leq G$), a beta-binomial GLM was fit to model ASRe counts as follows. Let Y_{ig} be the read counts of the first of the two alleles (which allele is designated as the first allele is arbitrary) for the i-th subject, $1 \leq i \leq I$. Investigators may designate the first and second alleles of a gene as the paternal and maternal alleles or as the alternate and reference alleles. It is assumed that $Y_{ig} \sim \text{BetaBin}(n_{ig}, p_{ig}, \phi_g)$, where n_{ig} is the equal to the total counts (of both alleles) for the i-th subject, p_{ig} is the probability of counts belonging to the first allele of the i-th subject, and ϕ_g is the overdispersion parameter. In this case, $\phi \to \infty$ implies no overdispersion beyond what would be seen in a binomial distribution and $\phi \to 0$ implies infinite dispersion. $n_{1g}, ..., n_{Ig}$ are assumed to be fixed and known. As the beta-binomial probability density function has multiple forms and parameterizations, we specify our parametrization below:

$$f(y_{ig}; n_{ig}, p_{ig}, \phi_g) = \frac{\binom{n_{ig}}{y_{ig}} B(y_{ig} + \phi_g p, n_{ig} - y_{ig} + \phi_g(1 - p_{ig}))}{B(\phi_g p_{ig}, \phi_g(1 - p_{ig}))} \tag{1}$$

where B specifies the beta function. Furthermore, let $\mathbf{x}_i^T$ be the i-th row of our design matrix $\mathbf{X}$ (matrix where columns are vectors of covariates of interest). Potential predictors include disease status for association studies, parent of origin for imprinting studies, and the presence of an SNP for eQTL linkage studies. We also assume that $p_{ig} = [1 + \exp(-\mathbf{x}_i^T \boldsymbol{\beta}_g)]^{-1}$, or equivalently $\text{logit}(p_{ig}) = \mathbf{x}_i^T \boldsymbol{\beta}_g$, where $\boldsymbol{\beta}_g = (\beta_{1g}, ..., \beta_{Kg})^T$ is a vector of coefficients representing the effect sizes for the predictors in our design matrix.

Apeglm additionally assumes a zero-centered Cauchy prior distribution for the effects of one of the predictors (Zhu, Ibrahim and Love 2018). Apeglm only shrinks the effect of one chosen predictor at a time, across all genes. We will denote the predictor whose effect investigators wish to shrink as the j-th predictor, $1 \leq j \leq K$.

To recap, we have:

$$Y_{ig} | \boldsymbol{\beta}_g \sim \text{BetaBin}(n_{ig}, p_{ig}, \phi_g) \tag{2}$$

$$p_{ig} = \frac{1}{1 + \exp(-\mathbf{x}_i^T \boldsymbol{\beta}_g)} \tag{3}$$

$$\beta_{jg} \sim \text{Cauchy}(0, \gamma_j) \tag{4}$$

ML estimation only assumes (2) and (3), and estimation of $\boldsymbol{\beta}_g$ by maximum likelihood gives us our ML estimates. Apeglm additionally assumes (4), and the posterior of $\boldsymbol{\beta}_g$ is the product of the above Cauchy prior and beta-binomial likleihood. That is, $\boldsymbol{\beta}_g | Y_{1g}, ..., Y_{Ig} \sim \text{Cauchy}(0, \gamma_j) \times \prod_{i=1}^{I} \text{BetaBin}(n_{ig}, p_{ig}, \phi_g)$. Genes with lower expression, smaller numbers of heterozygous subjects and higher dispersion in allelic proportions will have flatter likelihoods, which will lead to the prior having more influence and shrinkage being greater. Furthermore, if the ML estimates are tightly clustered about zero, the estimated scale parameter of our Cauchy prior will be smaller. This will lead to more peakedness in the prior and also cause shrinkage to be greater. The mode of the posterior gives us our apeglm estimates of $\boldsymbol{\beta}_g$.

Estimation Procedure

For each gene g, we wish to fit the generalized linear model $\text{logit}(p_{ig}) = \mathbf{x}_i^T \boldsymbol{\beta}_g$ where $Y_{ig}|\boldsymbol{\beta}_g \sim \text{BetaBin}(n_{ig}, p_{ig}, \phi_g)$ and $n_{1g}, ..., n_{Ig}$ are treated as fixed and known. For apeglm estimation, we additionally introduce a penalization/shrinkage term for the j-th predictor by assuming $\beta_{jg} \sim \text{Cauchy}(0, \gamma_j)$. The scale parameter of the prior, γ_j, is estimated a priori by sharing information across genes and is assumed fixed and known afterwords (see the original apeglm paper for details). Thus, there are two unknown parameters in our model: $\boldsymbol{\beta}_g$ and the nucense parameter ϕ_g. We iterate between these parameters in the following manner:

1. Calculate the MLE $\boldsymbol{\beta}_g$ as $\widehat{\boldsymbol{\beta}}_g^{(1)}$ assuming $\phi_g = \phi_g^{(0)}$, where $\phi_g^{(0)}$ is some initial guess (default is 100).
2. Caculate the MLE for ϕ_g as $\hat{\phi}_g^{(1)}$ assuming $\widehat{\boldsymbol{\beta}} = \widehat{\boldsymbol{\beta}}_g^{(1)}$
3. Calculate the MLE for $\boldsymbol{\beta}_g$ as $\widehat{\boldsymbol{\beta}}_g^{(2)}$ via maximum likelihood assuming $\phi_g = \hat{\phi}_g^{(1)}$
4. Calculate the MLE for ϕ_g as $\hat{\phi}_g$ via maximum likelihood assuming $\boldsymbol{\beta}_g = \widehat{\boldsymbol{\beta}}_g^{(2)}$
5. Calculate the apeglm estimate or MLE for $\boldsymbol{\beta}_g$ as $\widehat{\boldsymbol{\beta}}_g$ assuming $\phi_g = \hat{\phi}_g$, and this is taken as our final ML or apeglm estimate. For assumptions of the generalized linear model to be satisfied, we assume ϕ_g is both known and fixed at $\hat{\phi}_g$, and only $\boldsymbol{\beta}_g$ is the unknown parameter vector to estimate.

Estimating Coefficients while Overdispersion is Fixed

The steps below relate to iteration steps 1, 3 or 5 above and assume that overdispersion is fixed and known.

We first discuss the methodology for apeglm estimation. In addition to assuming (2), (3) and (4), we also assume that $\beta_{j'g} \sim N(0, \sigma^2)$ for all $j' \neq j$ when calculating estimates, where σ^2 is some large value (the default is 15^2). That is, we assume vague uninformative normal priors for effect sizes of the predictors that apeglm does not shrink, following the behavior of the software packages `bayesglm` (Gelman et. al. 2008) and `DESeq2` (Love, Huber and Anders 2014). These priors do not make a difference except when there is near-separability of the data or when the MLE is highly unstable due to very weak information. In these rare cases, the wide normal priors allow for more numerically stable estimates and oftentimes give more reliable solutions.

For the g-th gene, let $\mathbf{Y}_g = (Y_{1g}, ..., Y_{Ig})^T$, $\mathbf{n}_g = (n_{1g}, ..., n_{Ig})^T$, $\boldsymbol{\phi}_{g;I\times 1} = (\phi_g, ..., \phi_g)^T$, $\boldsymbol{\beta}_g = (\beta_{1g}, ..., \beta_{Kg})^T$, and $\boldsymbol{X}$ be the design matrix as defined previously. Furthermore, let $\ell(\boldsymbol{\beta}_g)$ denote the log-prior of $\boldsymbol{\beta}_g$, $\ell(\mathbf{Y}_g|\boldsymbol{\beta}_g)$ denote the log-likelihood of $\mathbf{Y}_g$, and $\ell(\boldsymbol{\beta}_g|\mathbf{Y}_g)$ denote the log-posterior of $\boldsymbol{\beta_g}$. For the below notation, we abandon the 'g' subscript for simplicity. Then it can be shown that:

$$
\begin{aligned}
\ell(\boldsymbol{\beta}) &\propto \log\left(\beta_j^2/\gamma_j^2 + 1\right) + \frac{1}{2\sigma^2}\sum_{j'\neq j}\beta_{j'}^2 \\
\ell(\mathbf{Y}|\boldsymbol{\beta}) &\propto \text{sum}\left\{\log\left[\Gamma\left(\mathbf{n} - \mathbf{Y} + \boldsymbol{\phi} - \frac{\boldsymbol{\phi}}{1+\exp(-\mathbf{X}\boldsymbol{\beta})}\right)\right]\right. \\
&\quad + \log\left[\Gamma\left(\mathbf{Y} + \frac{\boldsymbol{\phi}}{1+\exp(-\mathbf{X}\boldsymbol{\beta})}\right)\right] \\
&\quad - \log\left[\Gamma\left(\boldsymbol{\phi} - \frac{\boldsymbol{\phi}}{1+\exp(-\mathbf{X}\boldsymbol{\beta})}\right)\right] \\
&\quad \left. - \log\left[\Gamma\left(\frac{\boldsymbol{\phi}}{1+\exp(-\mathbf{X}\boldsymbol{\beta})}\right)\right]\right\} \\
\ell(\boldsymbol{\beta}|\mathbf{Y}) &\propto \ell(\boldsymbol{\beta}) + \ell(\mathbf{Y}|\boldsymbol{\beta})
\end{aligned}
$$

The logarithmic, gamma, division and summation operations are element-wise operations. For instance, if $\mathbf{X} = [x_{ij}]$, then $\log \mathbf{X} = [\log x_{ij}]$, $\Gamma(\mathbf{X}) = [\Gamma(x_{ij})]$, $\mathbf{1}/\mathbf{X} = [1/x_{ij}]$ and $\text{sum}(\mathbf{X}) = \sum_i \sum_j x_{ij}$. The term on the left side of $\ell(\boldsymbol{\beta})$ is the log-prior assumed by apeglm for the coefficient we wish to shrink and the summation term on the right side is the log-prior of the non-shrunk coefficients introduced for numerical stability. Furthermore, one can show that:

$$
\begin{aligned}
\ell'(\boldsymbol{\beta}) &= \left(\frac{\beta_1}{\sigma^2}, ..., \frac{\beta_{j-1}}{\sigma^2}, \frac{2\beta_j}{\gamma^2 + \beta_j^2}, \frac{\beta_{j+1}}{\sigma^2}, ..., \frac{\beta_K}{\sigma^2} \right)^T \\
\ell'(\mathbf{Y}|\boldsymbol{\beta}) &= \mathbf{X}^T \left[\frac{\phi \exp(-\mathbf{X}\boldsymbol{\beta})}{(1+\exp(-X\boldsymbol{\beta}))^{\circ 2}} \circ \left\{ \psi\left(\mathbf{n} - \mathbf{Y} + \boldsymbol{\phi} - \frac{\boldsymbol{\phi}}{1+\exp(-\mathbf{X}\boldsymbol{\beta})} \right) \right.\right. \\
&\qquad -\psi\left(\mathbf{Y} + \frac{\boldsymbol{\phi}}{1+\exp(-\mathbf{X}\boldsymbol{\beta})} \right) \\
&\qquad -\psi\left(\boldsymbol{\phi} - \frac{\boldsymbol{\phi}}{1+\exp(-\mathbf{X}\boldsymbol{\beta})} \right) \\
&\qquad \left.\left. +\psi\left(\frac{\boldsymbol{\phi}}{1+\exp(-\mathbf{X}\boldsymbol{\beta})} \right) \right\} \right] \\
\ell'(\boldsymbol{\beta}|\mathbf{Y}) &= \ell'(\boldsymbol{\beta}) + \ell'(\mathbf{Y}|\boldsymbol{\beta})
\end{aligned}
$$

where $\ell'(\boldsymbol{\beta}), \ell'(\mathbf{Y}|\boldsymbol{\beta}), \ell'(\boldsymbol{\beta}|\mathbf{Y})$ denote the gradient vectors of the log-prior, log-likelihood and log-posterior, respectively, $\mathbf{X} \circ \mathbf{Y}$ denotes element-wise multiplication of matrices X and Y, $\mathbf{X}^{\circ n}$ denotes the nth element-wise power of matrix X, division is element-wise as before, and ψ denotes element-wise operation of the digamma function. Using the above log-posterior and gradient vector, we estimate the mode of the log-posterior in C++ with the L-BFGS algorithm (Nocedal 1980) implemeneted within the `LBFGS++` and `RcppNumerical` libraries (Qiu et. al. 2018). The covariance matrix of our estimates is based on a Laplace approximation of the posterior. Specifically, the covariance matrix is estimated as the inverted Hessian, which is based on the coefficient estimates in C++ as well as the closed-form log posterior and gradient vector above, and is computed using the `optimHess` function in R.

For ML estimation, we just assume the same wide normal prior for all predictors, and the resulting changes to the log posterior and gradient of the log posterior are straightforward.

Estimating Overdispersion while Coefficients are Fixed

The steps below are executed in iteration steps 2 and 4 above and assume that coefficients are fixed and known. All computations are performed in R, and thus all referneced packaged and functions are R packages and functions.

The likelihood for the overdispersion parameter of the gth gene, ϕ_g, can be written as:

$$
\ell(\phi_g) = \sum_i \frac{\binom{n_{ig}}{Y_{ig}} B(Y_{ig} + \phi_g p_{ig}, n_{ig} - Y_{ig} + \phi_g(1-p_{ig}))}{B(\phi_g p_{ig}, \phi_g(1-p_{ig}))}
$$

n_{ig}, Y_{ig} and $p_{ig} = [1+\exp(-\mathbf{x}_i^T \boldsymbol{\beta}_g)]^{-1}$ were all defined in the "Model Setup" section. This is the summation of the beta-binomial density function as parametrized in (1), evaluated at each sample. The density for the beta-binomial is calculated using the `emdbook` library (Bolker 2019). Constrained optimization of the log-likelihood is performed using the `optimize` function (Brent 1973) with a minimum overdispersion parameter of 1 and a maximum of 500. These constrains are used so that genes with no overdispersion do not have infinite estimated values of ϕ.

Additional Technical Steps

Elements of the vector $\exp(-\mathbf{X}\boldsymbol{\beta})$ are capped between 0.001 and 999 when calculating $\ell(\mathbf{Y}|\boldsymbol{\beta})$ and $\ell'(\mathbf{Y}|\boldsymbol{\beta})$ to prevent calculations from becoming larger than what C++ double precision can handle. For the same reason, numerical variables are recommended to be standardized by the user to have mean 0 and variance 1. A linear transformation of the log-posterior is applied to prevent it from being too large or approaching too closely to zero. This improves computational performance and helps prevent convergence problems for our given posterior. When estimating overdispersion, the `optimize` function is technically used to estimate the MLE of $\log(\theta_g)$, which is then exponentiated to estimate the MLE of θ_g. The log-likelihood is more symmetrical with respect to $\log(\theta_g)$, which makes R's `optimize` function perform better. When running the `optim_lbfgs` function in C++ from the `RcppNumerical` library, we set `eps_f` to 1e-12 and `eps_g` to 1e-10 to control for the small relative movement in our posterior during convergence.