A Block Coordinate Descent Algorithm

We use a general purpose block coordinate descent algorithm (CGD) [58] to solve (16). At each iteration, the algorithm approximates the negative log-likelihood $f(\cdot)$ in $Q_\lambda(\cdot)$ by a strictly convex quadratic function and then applies block coordinate decent to generate a decent direction followed by an inexact line search along this direction [58]. For continuously differentiable $f(\cdot)$ and convex and block-separable $P(\cdot)$ (i.e. $P(\boldsymbol{\beta}) = \sum_i P_i(\beta_i)$), [58] show that the solution generated by the CGD method is a stationary point of $Q_\lambda(\cdot)$ if the coordinates are updated in a Gauss-Seidel manner i.e. $Q_\lambda(\cdot)$ is minimized with respect to one parameter while holding all others fixed. The CGD algorithm can thus be run in parallel and therefore suited for large p settings. It has been successfully applied in fixed effects models (e.g. [59], [20]) and [57] for mixed models with an ℓ_1 penalty. Following Tseng and Yun [58], the CGD algorithm is given by Algorithm 2.

The Armijo rule is defined as follows [58]:

Choose $\alpha_{init}^{(k)} > 0$ and let $\alpha^{(k)}$ be the largest element of $\left\{\alpha_{init}^{k}\delta^r\right\}_{r=0,1,2,\ldots}$ satisfying

$$Q_\lambda(\Theta_j^{(k)} + \alpha^{(k)} d^{(k)}) \leq Q_\lambda(\Theta_j^{(k)}) + \alpha^{(k)} \varrho \Delta^{(k)} \tag{45}$$

where $0 < \delta < 1$, $0 < \varrho < 1$, $0 \leq \gamma < 1$ and

$$\Delta^{(k)} := \nabla f(\Theta_j^{(k)}) d^{(k)} + \gamma (d^{(k)})^2 H_{jj}^{(k)} + \lambda P(\Theta_j^{(k)} + d^{(k)}) - \lambda P(\Theta^{(k)}) \tag{46}$$

Common choices for the constants are $\delta = 0.1$, $\varrho = 0.001$, $\gamma = 0$, $\alpha_{init}^{(k)} = 1$ for all k [57].

Below we detail the specifics of Algorithm 2 for the ℓ_1 penalty.

Algorithm 2: Coordinate Gradient Descent Algorithm to solve (16)

```
Set the iteration counter $k \leftarrow 0$ and choose initial values for the parameter vector $\boldsymbol{\Theta}^{(0)}$;
repeat
    Approximate the Hessian $\nabla^2 f(\boldsymbol{\Theta}^{(k)})$ by a symmetric matix $H^{(k)}$:
        $$H^{(k)} = \mathrm{diag}\left[\min\left\{\max\left\{\left[\nabla^2 f(\boldsymbol{\Theta}^{(k)})\right]_{jj}, c_{min}\right\} c_{max}\right\}\right]_{j=1,\dots,p} \quad (41)$$
    for $j = 1, \dots, p$ do
        Solve the descent direction $d^{(k)} := d_{H^{(k)}}(\Theta_j^{(k)})$ ;
        if $\Theta_j^{(k)} \in \{\beta_1, \dots, \beta_p\}$ then
            $$d_{H^{(k)}}(\Theta_j^{(k)}) \leftarrow \arg\min_d \left\{\nabla f(\Theta_j^{(k)})d + \frac{1}{2}d^2 H_{jj}^{(k)} + \lambda P(\Theta_j^{(k)} + d)\right\} \quad (42)$$
        end
    end
    Choose a stepsize;
        $$\alpha_j^{(k)} \leftarrow \text{line search given by the Armijo rule}$$
    Update;
        $$\widehat{\Theta}_j^{(k+1)} \leftarrow \widehat{\Theta}_j^{(k)} + \alpha_j^{(k)} d^{(k)}$$
    Update;
        $$\widehat{\eta}^{(k+1)} \leftarrow \arg\min_\eta \frac{1}{2}\sum_{i=1}^{N_T} \log(1 + \eta(\Lambda_i - 1)) + \frac{1}{2\sigma^{2\,(k)}} \sum_{i=1}^{N_T} \frac{\left(\widetilde{Y}_i - \sum_{j=0}^p \widetilde{X}_{ij+1}\beta_j^{(k+1)}\right)^2}{1 + \eta(\Lambda_i - 1)} \quad (43)$$
    Update;
        $$\widehat{\sigma^2}^{(k+1)} \leftarrow \frac{1}{N_T} \sum_{i=1}^{N_T} \frac{\left(\widetilde{Y}_i - \sum_{j=0}^p \widetilde{X}_{ij+1}\beta_j^{(k+1)}\right)^2}{1 + \eta^{(k+1)}(\Lambda_i - 1)} \quad (44)$$
    $k \leftarrow k + 1$
until *convergence criterion is satisfied*;
```

A.1 ℓ_1 penalty

The objective function is given by

$$Q_\lambda(\boldsymbol{\Theta}) = f(\boldsymbol{\Theta}) + \lambda|\boldsymbol{\beta}| \tag{47}$$

A.1.1 Descent Direction

For simplicity, we remove the iteration counter (k) from the derivation below.

For $\Theta_j^{(k)} \in \{\beta_1, \ldots, \beta_p\}$, let

$$d_H(\Theta_j) = \arg\min_d G(d) \tag{48}$$

where

$$G(d) = \nabla f(\Theta_j)d + \frac{1}{2}d^2 H_{jj} + \lambda|\Theta_j + d|$$

Since $G(d)$ is not differentiable at $-\Theta_j$, we calculate the subdifferential $\partial G(d)$ and search for d with $0 \in \partial G(d)$:

$$\partial G(d) = \nabla f(\Theta_j) + dH_{jj} + \lambda u \tag{49}$$

where

$$u = \begin{cases} 1 & \text{if} \quad d > -\Theta_j \\ -1 & \text{if} \quad d < -\Theta_j \\ [-1, 1] & \text{if} \quad d = \Theta_j \end{cases} \tag{50}$$

We consider each of the three cases in (49) below

1. $d > -\Theta_j$

$$\partial G(d) = \nabla f(\Theta_j) + dH_{jj} + \lambda = 0$$
$$d = \frac{-(\nabla f(\Theta_j) + \lambda)}{H_{jj}}$$

Since $\lambda > 0$ and $H_{jj} > 0$, we have

$$\frac{-(\nabla f(\Theta_j) - \lambda)}{H_{jj}} > \frac{-(\nabla f(\Theta_j) + \lambda)}{H_{jj}} = d \overset{\text{def}}{>} -\Theta_j$$

The solution can be written compactly as

$$d = \text{mid}\left\{\frac{-(\nabla f(\Theta_j) - \lambda)}{H_{jj}}, -\Theta_j, \frac{-(\nabla f(\Theta_j) + \lambda)}{H_{jj}}\right\}$$

804 where mid $\{a, b, c\}$ denotes the median (mid-point) of a, b, c [58].

2. $d < -\Theta_j$

$$\partial G(d) = \nabla f(\Theta_j) + dH_{jj} - \lambda = 0$$
$$d = \frac{-(\nabla f(\Theta_j) - \lambda)}{H_{jj}}$$

Since $\lambda > 0$ and $H_{jj} > 0$, we have

$$\frac{-(\nabla f(\Theta_j) + \lambda)}{H_{jj}} < \frac{-(\nabla f(\Theta_j) - \lambda)}{H_{jj}} = d \overset{\text{def}}{<} -\Theta_j$$

Again, the solution can be written compactly as

$$d = \text{mid}\left\{\frac{-(\nabla f(\Theta_j) - \lambda)}{H_{jj}}, -\Theta_j, \frac{-(\nabla f(\Theta_j) + \lambda)}{H_{jj}}\right\}$$

3. $d_j = -\Theta_j$

There exists $u \in [-1, 1]$ such that

$$\partial G(d) = \nabla f(\Theta_j) + dH_{jj} + \lambda u = 0$$
$$d = \frac{-(\nabla f(\Theta_j) + \lambda u)}{H_{jj}}$$

For $-1 \leq u \leq 1$, $\lambda > 0$ and $H_{jj} > 0$ we have

$$\frac{-(\nabla f(\Theta_j) + \lambda)}{H_{jj}} \leq d \overset{\text{def}}{=} -\Theta_j \leq \frac{-(\nabla f(\Theta_j) - \lambda)}{H_{jj}}$$

The solution can again be written compactly as

$$d = \text{mid}\left\{\frac{-(\nabla f(\Theta_j) - \lambda)}{H_{jj}}, -\Theta_j, \frac{-(\nabla f(\Theta_j) + \lambda)}{H_{jj}}\right\}$$

We see all three cases lead to the same solution for (48). Therefore the descent direction for $\Theta_j^{(k)} \in \{\beta_1, \ldots, \beta_p\}$ for the ℓ_1 penalty is given by

$$d = \text{mid}\left\{\frac{-(\nabla f(\beta_j) - \lambda)}{H_{jj}}, -\beta_j, \frac{-(\nabla f(\beta_j) + \lambda)}{H_{jj}}\right\} \tag{51}$$

A.1.2 Solution for the β parameter

If the Hessian $\nabla^2 f(\mathbf{\Theta}^{(k)}) > 0$ then $H^{(k)}$ defined in (41) is equal to $\nabla^2 f(\mathbf{\Theta}^{(k)})$. Using $\alpha_{init} = 1$, the largest element of $\left\{\alpha_{init}^{(k)}\delta^r\right\}_{r=0,1,2,\ldots}$ satisfying the Armijo Rule inequality is reached for $\alpha^{(k)} = \alpha_{init}^{(k)}\delta^0 = 1$. The Armijo rule update for the $\boldsymbol{\beta}$ parameter is then given by

$$\beta_j^{(k+1)} \leftarrow \beta_j^{(k)} + d^{(k)}, \qquad j = 1, \ldots, p \tag{52}$$

Substituting the descent direction given by (51) into (52) we get

$$\beta_j^{(k+1)} = \text{mid}\left\{\beta_j^{(k)} + \frac{-(\nabla f(\beta_j^{(k)}) - \lambda)}{H_{jj}}, 0, \beta_j^{(k)} + \frac{-(\nabla f(\beta_j^{(k)}) + \lambda)}{H_{jj}}\right\} \tag{53}$$

We can further simplify this expression. Let

$$w_i := \frac{1}{\sigma^2\left(1 + \eta(\Lambda_i - 1)\right)} \tag{54}$$

.

Re-write the part depending on $\boldsymbol{\beta}$ of the negative log-likelihood in (14) as

$$g(\boldsymbol{\beta}^{(k)}) = \frac{1}{2}\sum_{i=1}^{N_T} w_i \left(\widetilde{Y}_i - \sum_{\ell\neq j}\widetilde{X}_{i\ell}\beta_\ell^{(k)} - \widetilde{X}_{ij}\beta_j^{(k)}\right)^2 \tag{55}$$

The gradient and Hessian are given by

$$\nabla f(\beta_j^{(k)}) \coloneqq \frac{\partial}{\partial\beta_j^{(k)}} g(\boldsymbol{\beta}^{(k)}) = -\sum_{i=1}^{N_T} w_i\widetilde{X}_{ij}\left(\widetilde{Y}_i - \sum_{\ell\neq j}\widetilde{X}_{i\ell}\beta_\ell^{(k)} - \widetilde{X}_{ij}\beta_j^{(k)}\right) \tag{56}$$

$$H_{jj} \coloneqq \frac{\partial^2}{\partial{\beta_j^{(k)}}^2} g(\boldsymbol{\beta}^{(k)}) = \sum_{i=1}^{N_T} w_i\widetilde{X}_{ij}^2 \tag{57}$$

Substituting (56) and (57) into $\beta_j^{(k)} + \frac{-(\nabla f(\beta_j^{(k)})-\lambda)}{H_{jj}}$

$$\begin{aligned}
&\beta_j^{(k)} + \frac{\sum_{i=1}^{N_T} w_i\widetilde{X}_{ij}\left(\widetilde{Y}_i - \sum_{\ell\neq j}\widetilde{X}_{i\ell}\beta_\ell^{(k)} - \widetilde{X}_{ij}\beta_j^{(k)}\right) + \lambda}{\sum_{i=1}^{N_T} w_i\widetilde{X}_{ij}^2} \\
&= \beta_j^{(k)} + \frac{\sum_{i=1}^{N_T} w_i\widetilde{X}_{ij}\left(\widetilde{Y}_i - \sum_{\ell\neq j}\widetilde{X}_{i\ell}\beta_\ell^{(k)}\right) + \lambda}{\sum_{i=1}^{N_T} w_i\widetilde{X}_{ij}^2} - \frac{\sum_{i=1}^{N_T} w_i\widetilde{X}_{ij}^2\beta_j^{(k)}}{\sum_{i=1}^{N_T} w_i\widetilde{X}_{ij}^2} \\
&= \frac{\sum_{i=1}^{N_T} w_i\widetilde{X}_{ij}\left(\widetilde{Y}_i - \sum_{\ell\neq j}\widetilde{X}_{i\ell}\beta_\ell^{(k)}\right) + \lambda}{\sum_{i=1}^{N_T} w_i\widetilde{X}_{ij}^2}
\end{aligned} \tag{58}$$

Similarly, substituting (56) and (57) in $\beta_j^{(k)} + \frac{-(\nabla f(\beta_j^{(k)})+\lambda)}{H_{jj}}$ we get

$$\frac{\sum_{i=1}^{N_T} w_i\widetilde{X}_{ij}\left(\widetilde{Y}_i - \sum_{\ell\neq j}\widetilde{X}_{i\ell}\beta_\ell^{(k)}\right) - \lambda}{\sum_{i=1}^{N_T} w_i\widetilde{X}_{ij}^2} \tag{59}$$

Finally, substituting (58) and (59) into (53) we get

$$\begin{aligned}\beta_j^{(k+1)} &= \text{mid}\left\{\frac{\sum_{i=1}^{N_T} w_i\widetilde{X}_{ij}\left(\widetilde{Y}_i - \sum_{\ell\neq j}\widetilde{X}_{i\ell}\beta_\ell^{(k)}\right)-\lambda}{\sum_{i=1}^{N_T} w_i\widetilde{X}_{ij}^2}, 0, \frac{\sum_{i=1}^{N_T} w_i\widetilde{X}_{ij}\left(\widetilde{Y}_i - \sum_{\ell\neq j}\widetilde{X}_{i\ell}\beta_\ell^{(k)}\right)+\lambda}{\sum_{i=1}^{N_T} w_i\widetilde{X}_{ij}^2}\right\} \\ &= \frac{\mathcal{S}_\lambda\left(\sum_{i=1}^{N_T} w_i\widetilde{X}_{ij}\left(\widetilde{Y}_i - \sum_{\ell\neq j}\widetilde{X}_{i\ell}\beta_\ell^{(k)}\right)\right)}{\sum_{i=1}^{N_T} w_i\widetilde{X}_{ij}^2} \end{aligned} \tag{60}$$

Where $\mathcal{S}_\lambda(x)$ is the soft-thresholding operator

$$\mathcal{S}_\lambda(x) = \text{sign}(x)(|x|-\lambda)_+$$

sign(x) is the signum function

$$\text{sign}(x) = \begin{cases} -1 & x < 0 \\ 0 & x = 0 \\ 1 & x > 0 \end{cases}$$

and $(x)_+ = \max(x, 0)$.

815 # B Additional Real Data Analysis Results

816 ## B.1 Distribution of SNPs used in UK Biobank analysis

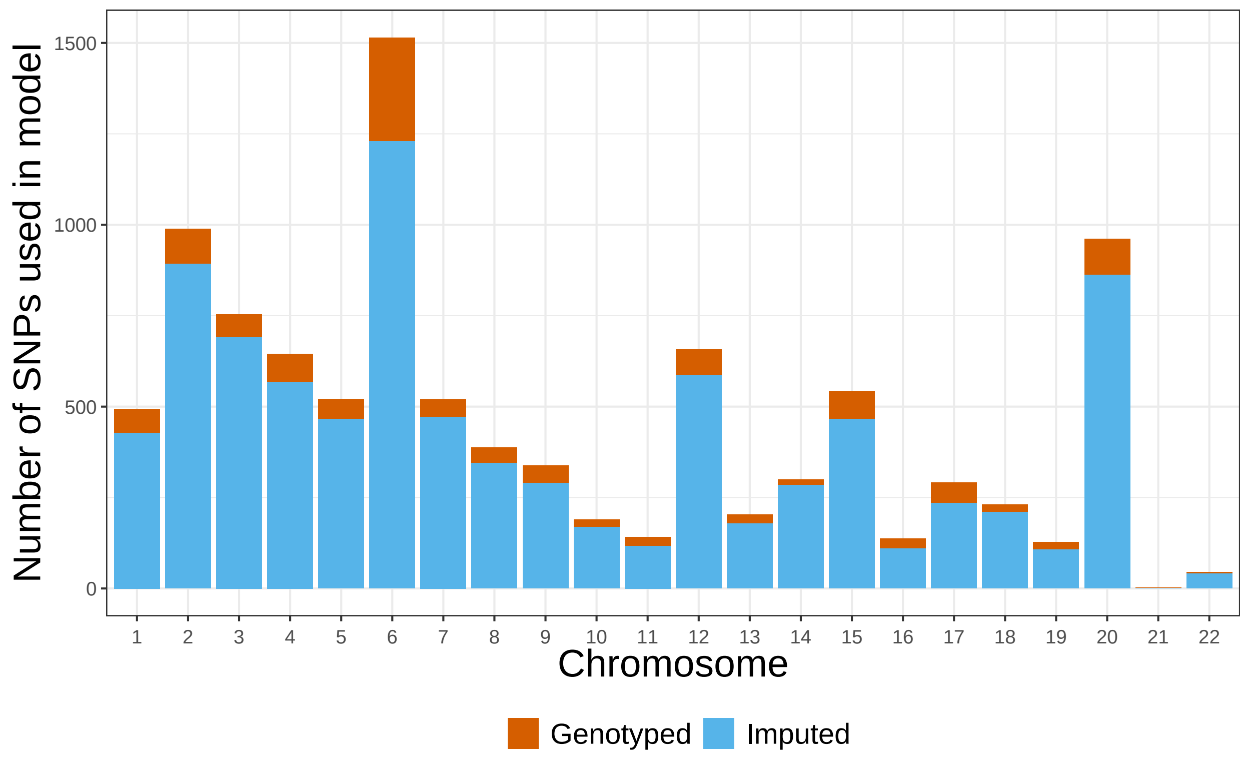


Figure B.1: Distribution of SNPs used in UK Biobank analysis by chromosome and whether or not the SNP was imputed.

B.2 LD structure among the markers in the GAW20 and the mouse dataset

We illustrate the LD structure among the markers in the GAW20 dataset and the mouse dataset separately in Figures B.2 and B.3, respectively. In Figure B.2, we show the pairwise r^2 for 655 SNPs within a 1Mb-window around the causal SNP rs9661059 (indicated) that we focused on. The dotplot above the heatmap denotes r^2 between each SNP and the causal SNP. It is clear that although strong correlation does exist between some SNPs, none of these nearby SNPs is correlated with the causal SNP. The only dot denoting an $r^2 = 1$ represents the causal SNP itself.

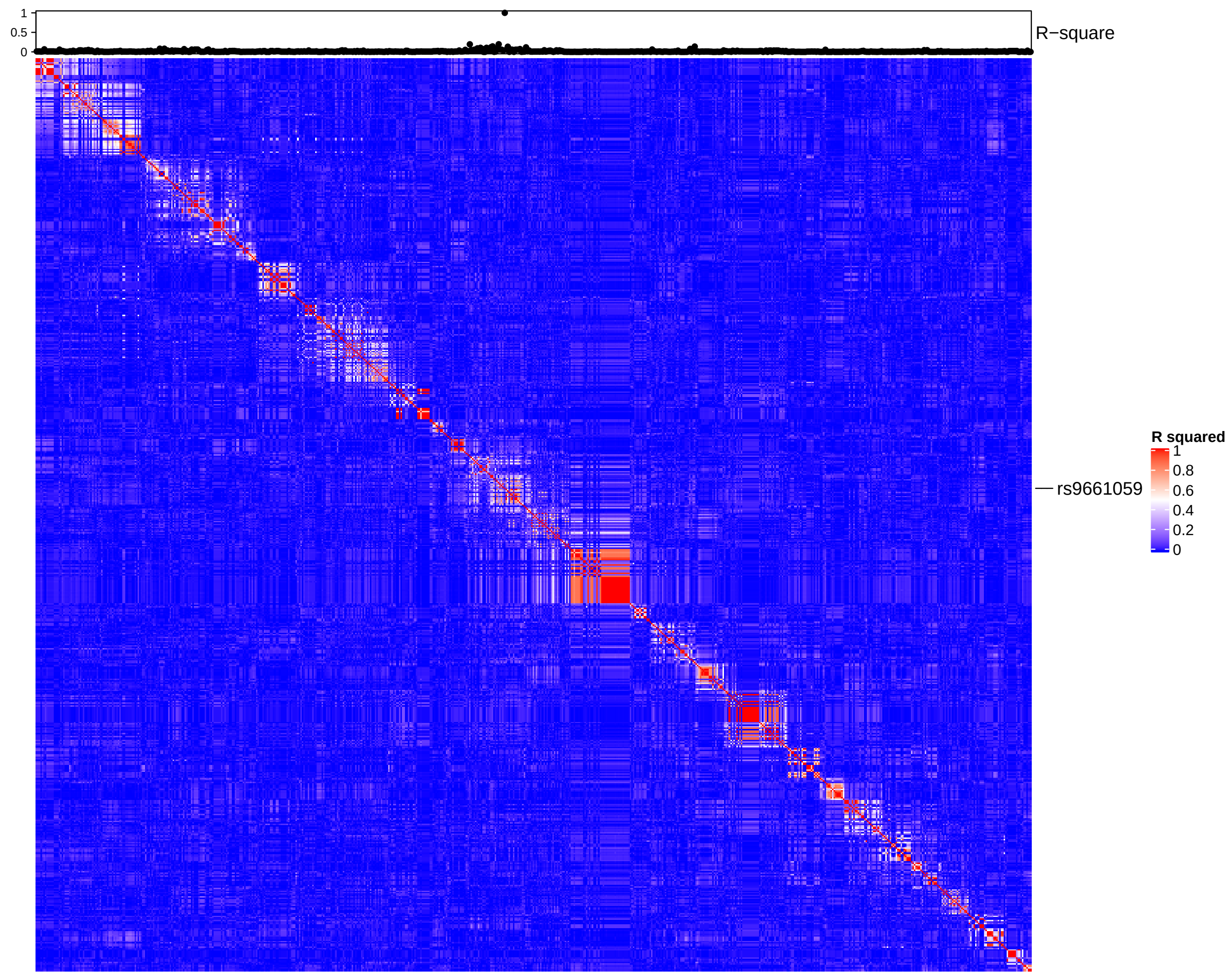


Figure B.2: LD structure among the markers in the GAW20 dataset

In Figure B.3, we show the pairwise r^2 for all microsatellite markers in the mouse dataset. It is clear that many markers are considerably strongly correlated with each other, as we expected.

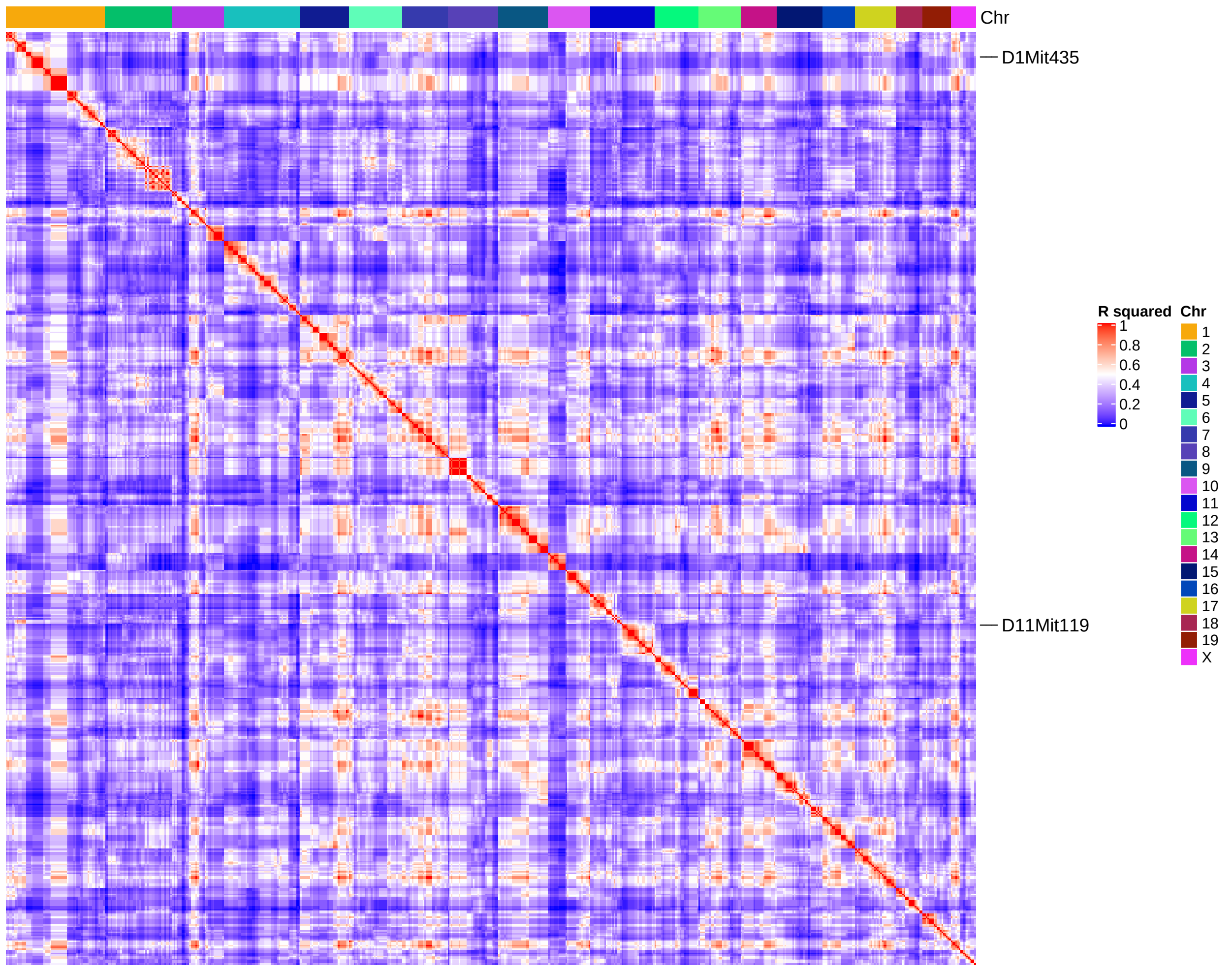


Figure B.3: LD structure among the markers in the mouse dataset

C ggmix Package Showcase

In this section we briefly introduce the freely available and open source ggmix package in R. More comprehensive documentation is available at https://sahirbhatnagar.com/ggmix. Note that this entire section is reproducible; the code and text are combined in an .Rnw[1] file and compiled using knitr [66].

C.1 Installation

The package can be installed from GitHub via

```
install.packages("pacman")
pacman::p_load_gh('sahirbhatnagar/ggmix')
```

To showcase the main functions in ggmix, we will use the simulated data which ships with the package and can be loaded via:

```
library(ggmix)
data("admixed")
names(admixed)

##  [1] "y"              "x"              "causal"
##  [4] "beta"           "kin"            "Xkinship"
##  [7] "not_causal"     "causal_positive" "causal_negative"
## [10] "x_lasso"
```

For details on how this data was simulated, see help(admixed).

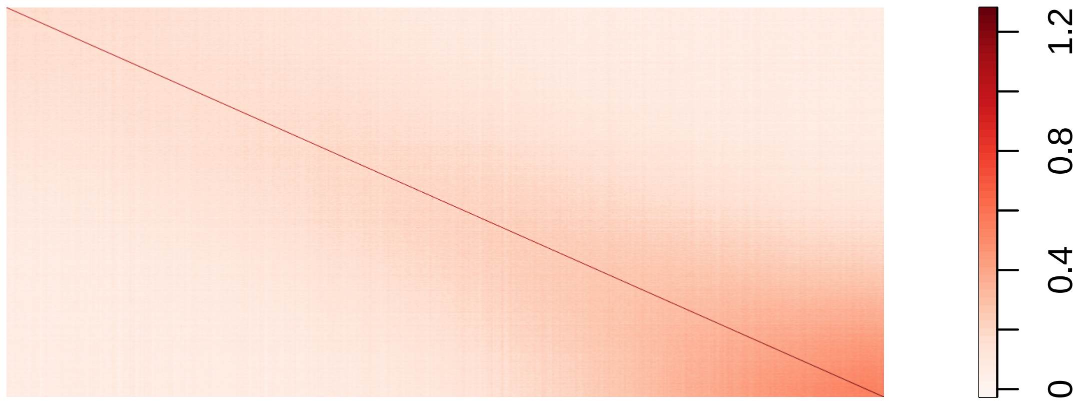
There are three basic inputs that ggmix needs:

1. Y: a continuous response variable
2. X: a matrix of covariates of dimension $N \times p$ where N is the sample size and p is the number of covariates
3. $\mathbf{\Phi}$: a kinship matrix

[1]scripts available at https://github.com/sahirbhatnagar/ggmix/tree/pgen/manuscript

We can visualize the kinship matrix in the `admixed` data using the `popkin` package:

```r
# need to install the package if you don't have it
# pacman::p_load_gh('StoreyLab/popkin')
popkin::plotPopkin(admixed$kin)
```



C.2 Fit the linear mixed model with Lasso Penalty

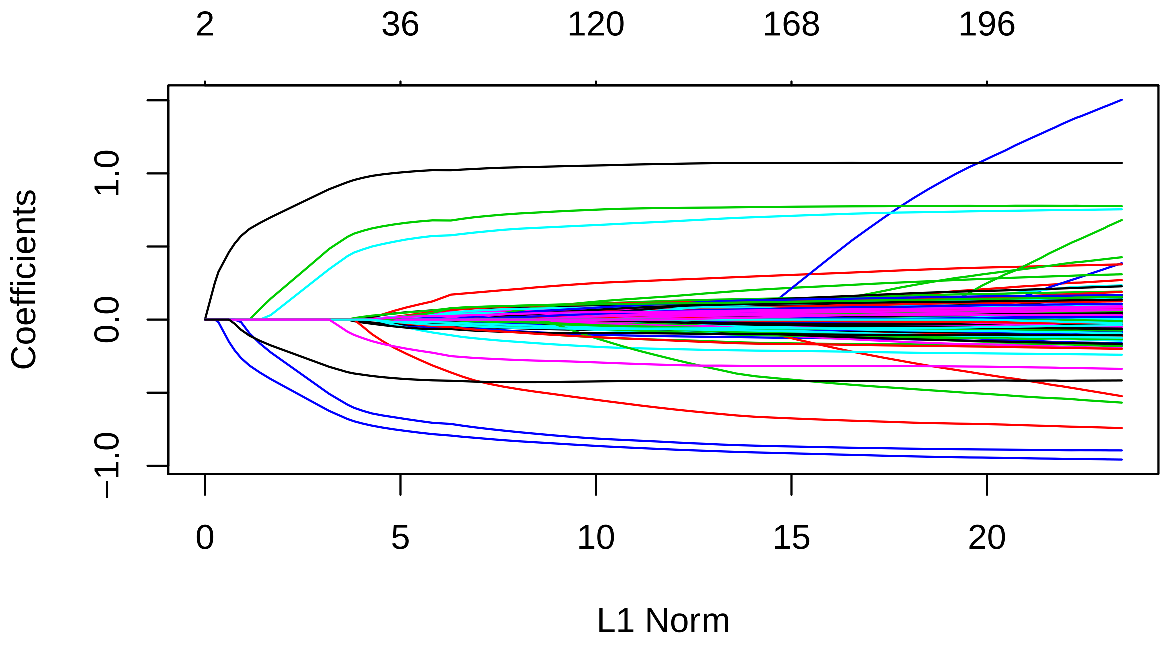
We will use the most basic call to the main function of this package, which is called `ggmix`. This function will by default fit a L_1 penalized linear mixed model (LMM) for 100 distinct values of the tuning parameter λ. It will choose its own sequence:

```r
fit <- ggmix(x = admixed$x, y = admixed$y, kinship = admixed$kin)
```

```
names(fit)

##  [1] "result"       "ggmix_object" "n_design"     "p_design"
##  [5] "lambda"       "coef"         "b0"           "beta"
##  [9] "df"           "eta"          "sigma2"       "nlambda"
## [13] "cov_names"    "call"

class(fit)

## [1] "lassofullrank" "ggmix_fit"
```

We can see the solution path for each variable by calling the `plot` method for objects of class `ggmix_fit`:

```
plot(fit)
```



We can also get the coefficients for given value(s) of lambda using the `coef` method for objects of class `ggmix_fit`:

```
# only the first 5 coefficients printed here for brevity
```

```
coef(fit, s = c(0.1,0.02))[1:5, ]

## 5 x 2 Matrix of class "dgeMatrix"
##                       1            2
## (Intercept) -0.3824525 -0.030224599
## X62          0.0000000  0.000000000
## X185         0.0000000  0.001444518
## X371         0.0000000  0.009513475
## X420         0.0000000  0.000000000
```

Here, `s` specifies the value(s) of λ at which the extraction is made. The function uses linear interpolation to make predictions for values of `s` that do not coincide with the lambda sequence used in the fitting algorithm.

We can also get predictions ($X\widehat{\boldsymbol{\beta}}$) using the `predict` method for objects of class `ggmix_fit`:

```
# need to provide x to the predict function
# predict for the first 5 subjects
predict(fit, s = c(0.1,0.02), newx = admixed$x[1:5,])

##               1          2
## id1 -1.19165061 -1.3123392
## id2 -0.02913052  0.3885923
## id3 -2.00084875 -2.6460043
## id4 -0.37255277 -0.9542463
## id5 -1.03967831 -2.1377268
```

C.3 Find the Optimal Value of the Tuning Parameter

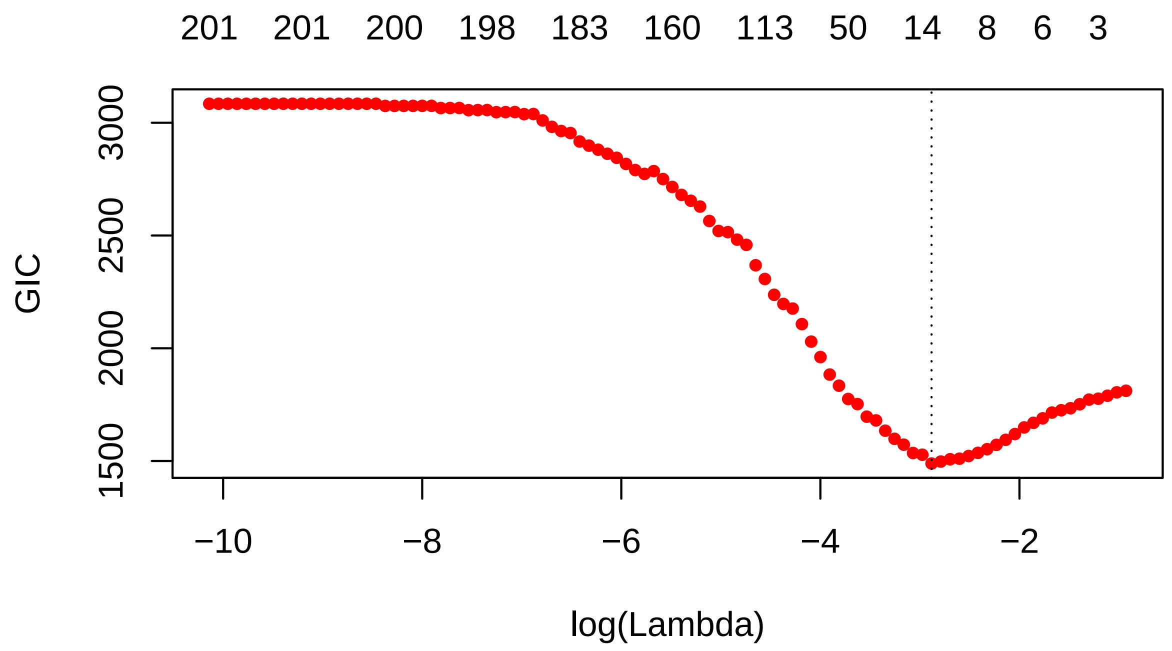
We use the Generalized Information Criterion (GIC) to select the optimal value for λ. The default is $a_n = log(log(n)) * log(p)$ which corresponds to a high-dimensional BIC (HD-BIC):

```
# pass the fitted object from ggmix to the gic function:
```

```
hdbic <- gic(fit)
class(hdbic)

## [1] "ggmix_gic"     "lassofullrank" "ggmix_fit"

# we can also fit the BIC by specifying the an argument
bicfit <- gic(fit, an = log(length(admixed$y)))
```

We can plot the HDBIC values against $\log(\lambda)$ using the `plot` method for objects of class `ggmix_gic`:

```
plot(hdbic)
```

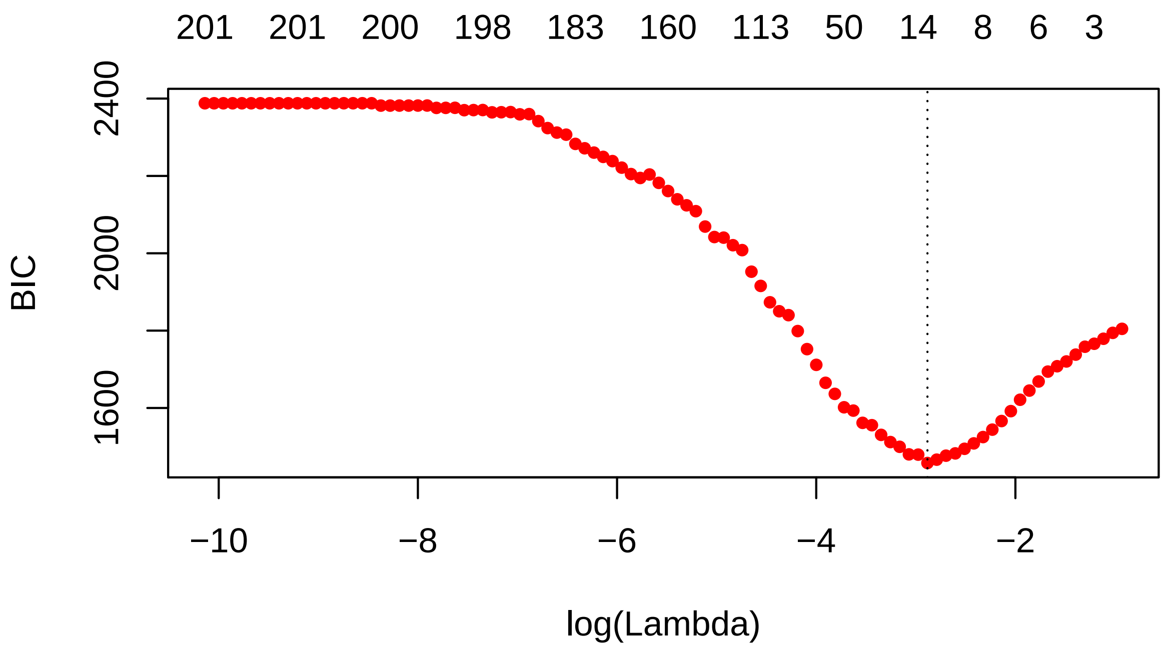


The optimal value for λ according to the HDBIC, i.e., the λ that leads to the minium HDBIC is:

```
hdbic[["lambda.min"]]

## [1] 0.05596623
```

We can also plot the BIC results:

```
plot(bicfit, ylab = "BIC")
```



```
bicfit[["lambda.min"]]

## [1] 0.05596623
```

C.4 Get Coefficients Corresponding to Optimal Model

We can use the object outputted by the `gic` function to extract the coefficients corresponding to the selected model using the `coef` method for objects of class `ggmix_gic`:

```
coef(hdbic)[1:5, , drop = FALSE]

## 5 x 1 sparse Matrix of class "dgCMatrix"
##                      1
## (Intercept) -0.2668419
## X62          .
## X185         .
## X371         .
## X420         .
```

We can also extract just the nonzero coefficients which also provide the estimated variance components η and σ^2:

```
coef(hdbic, type = "nonzero")

##                       1
## (Intercept) -0.26684191
## X336        -0.67986393
## X7638        0.43403365
## X1536        0.93994982
## X1943        0.56600730
## X2849       -0.58157979
## X56         -0.08244685
## X4106       -0.35939830
## eta          0.26746240
## sigma2       0.98694300
```

We can also make predictions from the `hdbic` object, which by default will use the model corresponding to the optimal tuning parameter:

```
predict(hdbic, newx = admixed$x[1:5,])

##              1
## id1 -1.3061041
## id2  0.2991654
## id3 -2.3453664
## id4 -0.4486012
## id5 -1.3895793
```

C.5 Extracting Random Effects

The user can compute the random effects using the provided `ranef` method for objects of class `ggmix_gic`. This command will compute the estimated random effects for each subject using the parameters of the selected model:

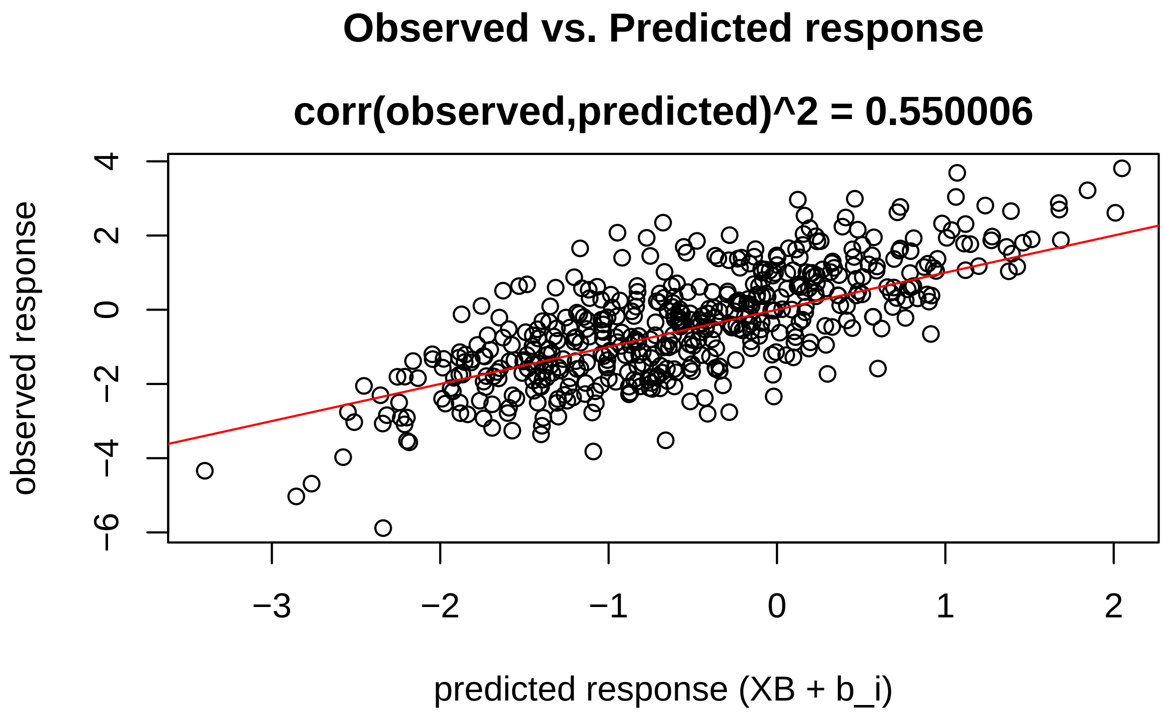
```
ranef(hdbic)[1:5]

## [1] -0.85505649 -0.88390831  0.09124002 -0.55031545 -0.20364743
```

C.6 Diagnostic Plots

We can also plot some standard diagnotic plots such as the observed vs. predicted response, QQ-plots of the residuals and random effects and the Tukey-Anscombe plot. These can be plotted using the `plot` method on a `ggmix_gic` object as shown below.

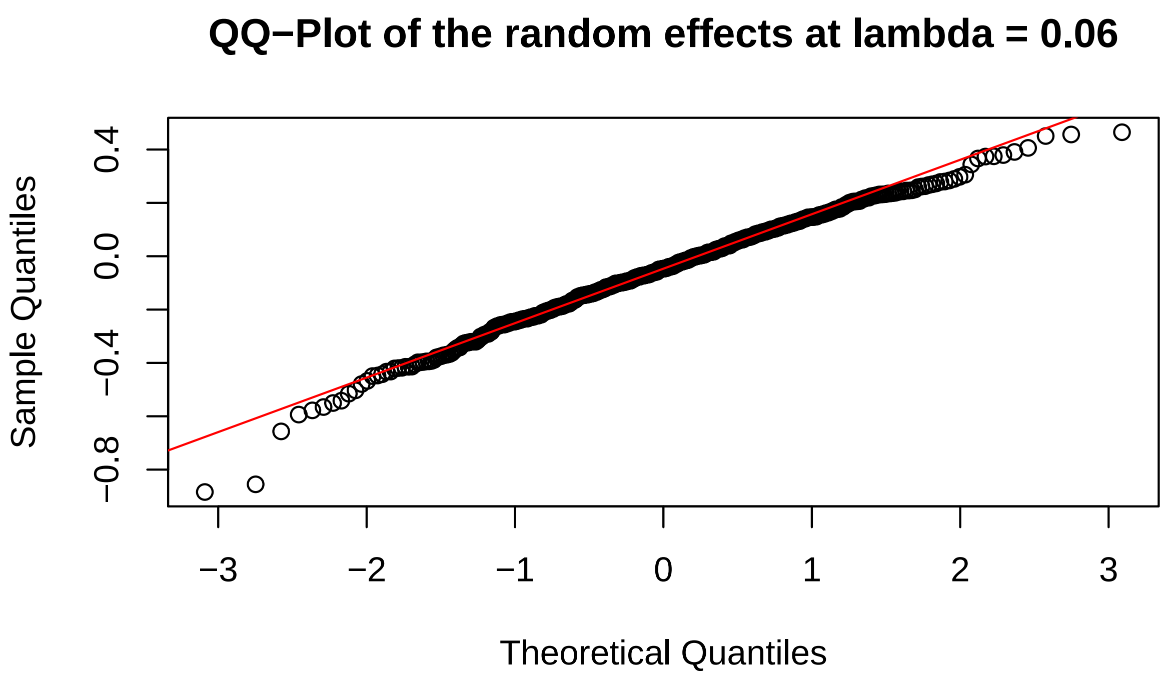
C.6.1 Observed vs. Predicted Response

```
plot(hdbic, type = "predicted", newx = admixed$x, newy = admixed$y)
```



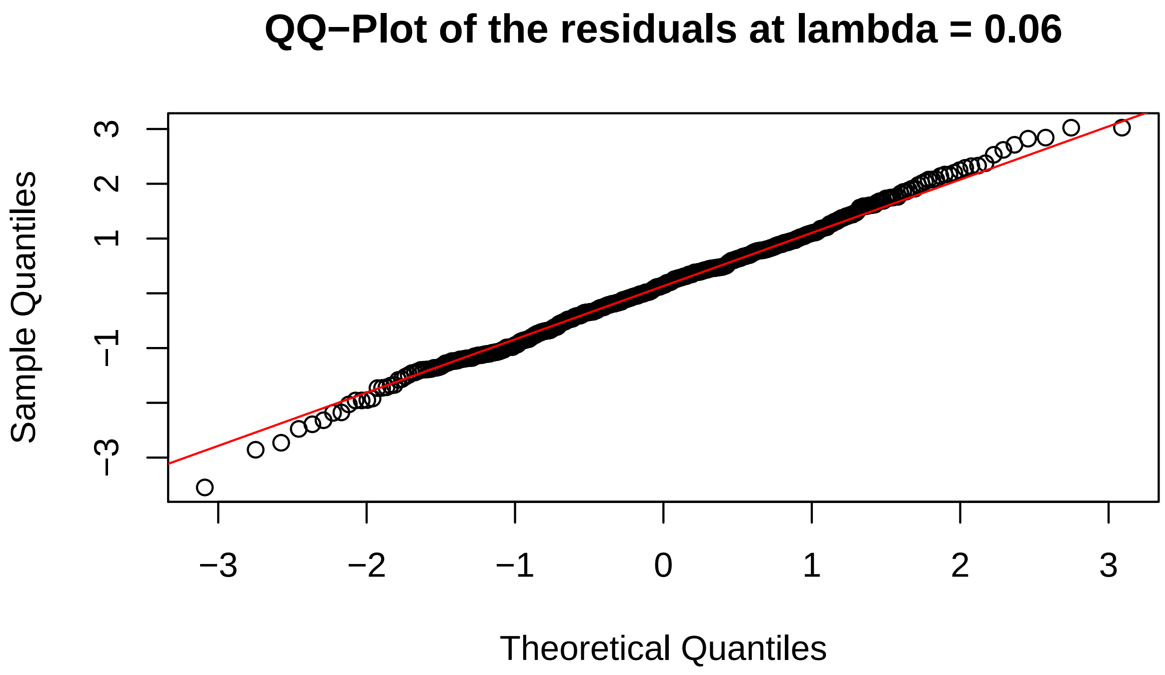
C.6.2 QQ-plots for Residuals and Random Effects

```r
plot(hdbic, type = "QQranef", newx = admixed$x, newy = admixed$y)
```



888

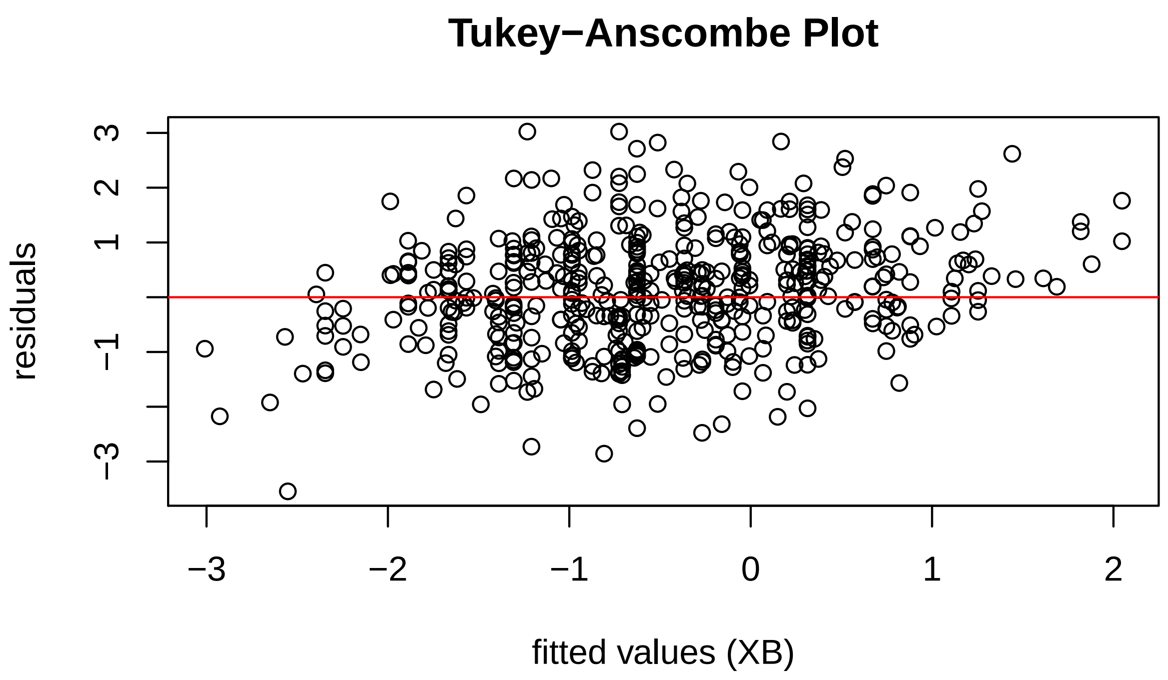
```r
plot(hdbic, type = "QQresid", newx = admixed$x, newy = admixed$y)
```



889

890 ### C.6.3 Tukey-Anscombe Plot

```
plot(hdbic, type = "Tukey", newx = admixed$x, newy = admixed$y)
```



891