The RepeatABEL package - a tutorial

Lars Rönnegård

December 12, 2025

Introduction

This vignette gives an introduction to the RepeatABEL package. The package performs GWAS for data where there are repeated observations on individuals that may be related. Various random effects can be fitted. Random polygenic effects are fitted by default, but also other random effects can be fitted including spatial effects. A model without the fixed SNP effect is fitted using the hglm package for estimating variance components (see *Ronnegard, Shen & Alam (2010) hglm: A package for fitting hierarchical generalized linear models. The R Journal 2:20-28*). These variance component estimates are subsequently used in the GWAS where each marker is fitted one at a time. The package consists of four main functions `rGLS`, `preFitModel`, `simulate_genData` and `simulate_PhenData`.

Theory for the `rGLS` function

The `rGLS` function is the main function of the package and by default fits a linear mixed model including permanent environmental effects $\boldsymbol{p}$ and polygenic effects $\boldsymbol{g}$ with correlation matrix given by the genomic relationship matrix (aka kinship matrix) in

$$\boldsymbol{y} = \boldsymbol{X\beta} + \boldsymbol{Zg} + \boldsymbol{Zp} + \boldsymbol{e} \tag{1}$$

where $\boldsymbol{y}$ is the studied trait, $\boldsymbol{\beta}$ are fixed effects, $\boldsymbol{e}$ are residuals with $\boldsymbol{e} \sim N(0, \sigma_e^2)$ having residual variance σ_e^2. Furthermore, $\boldsymbol{X}$ and $\boldsymbol{Z}$ are model matrices. The random effects are assumed multivariate normal such that $\boldsymbol{p} \sim N(0, \boldsymbol{I}\sigma_p^2)$ and $\boldsymbol{g} \sim N(0, \boldsymbol{G}\sigma_g^2)$ where $\boldsymbol{G}$ is the genomic relationship matrix. Thus the estimated (co)variance matrix for this model is

$$\hat{\boldsymbol{V}} = \boldsymbol{ZGZ}^T\hat{\sigma}_g^2 + \boldsymbol{ZZ}^T\hat{\sigma}_p^2 + \boldsymbol{I}\hat{\sigma}_e^2. \tag{2}$$

Subsequently, a linear model is fitted (using generalized least squares, GLS) for each marker where the covariate x_{SNP} is coded as 0,1,2:

$$\boldsymbol{y} = \boldsymbol{X\beta} + x_{SNP}\beta + \boldsymbol{\varepsilon} \tag{3}$$

with

$$\boldsymbol{\varepsilon} \sim N(0, \sigma_\epsilon^2\hat{\boldsymbol{V}}). \tag{4}$$

A Wald test is used to compute the P value for SNP effect β.

The computations are made fast by applying a eigen-decomposition of $\hat{V}$ and using the built-in `qr` function in R to fit the linear models (3).

Using the `rGLS` function

The following example illustrates the use of the function. There are two data objects to be included in the input: a GenABEL-like object (of class gwaa.data2) including the genotypic information and a data frame including the phenotypic information. The name of the ID variable in the phenotype data frame should be "id" (otherwise specify `id.name` equal to the ID variable name).

In this example there are 400 observations from 100 individuals, and there are 1000 SNP to be tested.

```
> library(RepeatABEL)
> set.seed(1234)
> #GenABEL object including IDs and marker genotypes
> Gen.Data <- simulate_gendata(n=100, p=1000)
> #Phenotype data with repeated observations
> Phen.Data <- simulate_PhenData(y ~ 1, genabel.data=Gen.Data,
                                 n.obs=rep(4, nids(Gen.Data)),
                                 SNP.eff=1,
                                 SNP.nr=500,
                                 VC=c(1,1,1)
                                 )
```

The data frame `Phen.Data` includes the trait value y and the ID of the individuals. The function input is

```
> GWAS1 <- rGLS(y ~ 1, genabel.data = Gen.Data,
    phenotype.data = Phen.Data)
GRM ready
Variance component estimation ready
[1] "Rotation matrix ready"
Rotate LMM started
Rotate LMM ready
```

The computations in this function consists of four parts: construction of the GRM, variance component estimation using the `hglm` function, rotating the GLS using eigen-decomposition transforming it to an ordinary least squares (OLS) problem, and finally fitting an OLS for each SNP. How far the computations have come is shown in the output.

The class of the output object GWAS1 is `gwaa.scan2`, and there are generic functions `summary()` and `plot()` that are slimmed versions of the corresponding functions in the archived GenABEL package.

```
> summary(GWAS1)
Summary for top 10 results, sorted by P1df
          effB   se_effB chi2.1df         P1df Pc1df
500  1.2043035 0.2324286       NA 2.202370e-07    NA
501  0.9557313 0.2377023       NA 5.802385e-05    NA
499  0.9897830 0.2592018       NA 1.342216e-04    NA
504  0.9384720 0.2480936       NA 1.551159e-04    NA
503  0.9490128 0.2766188       NA 6.018926e-04    NA
502  0.8714747 0.2574326       NA 7.111262e-04    NA
226 -0.7793113 0.2447636       NA 1.452877e-03    NA
334  0.8324358 0.2753942       NA 2.505256e-03    NA
335  0.7834660 0.2630524       NA 2.897914e-03    NA
174  0.7349158 0.2485552       NA 3.108994e-03    NA
```

Extracting the genotypic variance, a 95% CI and the heritability

From the GWAS1 object, the estimated variance components for the prefitted model, not including SNP effects, can be extracted as follows.

```
> est.hglm <- GWAS1@call$hglm
> cat("Genotypic and permanent env. variance components:","\n",
        est.hglm$varRanef,", resp.","\n",
        "The residual variance is", est.hglm$varFix,".","\n")
Genotypic and permanent env. variance components:
 1.4784 1.254914 , resp.
 The residual variance is 0.9468596 .
```

Furthermore, the standard errors for the estimated variance components are computed on a log scale. Thereby, 95% confidence intervals can be computed for the variance components. Here the computations for the genotypic variance are shown.

```
> logVCE <- est.hglm$SummVC2[[1]]
> cat("Estimate and SE for the genotypic variance on a natural log scale:",
        "\n", logVCE[1],"and",logVCE[2],", resp.", "\n \n",
        "Confidence interval: [", exp( logVCE[1] - 1.96*logVCE[2] ) , "," ,
        exp( logVCE[1] + 1.96*logVCE[2] ) , "]" , "\n")
Estimate and SE for the genotypic variance on a natural log scale:
 0.3909603 and 0.2131857 , resp.

 Confidence interval: [ 0.9734744 , 2.245222 ]
```

The heritability for this example is computed as:

```
> cat("Heritability:",
        est.hglm$varRanef[1]/(est.hglm$varFix + sum(est.hglm$varRanef)),
        "\n")
Heritability: 0.4017202
```

Using the preFitModel function

The function preFitModel is used for variance component estimation and increases the flexibility of modeling in the rGLS function.

Fitting the same model as above

To start with we have a look at how the model in the previous section can be fitted in two steps using the preFitModel function and thereafter the rGLS function. Note that the results are exactly the same as in the previous section.

```
> #The same results can be computed using the preFitModel as follows
> fixed=y ~ 1
> Mod1 <- preFitModel(fixed, random=~1|id,
      genabel.data = Gen.Data,
      phenotype.data = Phen.Data,
      corStruc=list( id=list("GRM","Ind") ))
GRM ready
> GWAS1b <- rGLS(fixed, genabel.data = Gen.Data,
    phenotype.data = Phen.Data, V = Mod1$V)
[1] "Rotation matrix ready"
Rotate LMM started
Rotate LMM ready
> summary(GWAS1b)
Summary for top 10 results, sorted by P1df
          effB   se_effB chi2.1df         P1df Pc1df
500  1.2043035 0.2324286       NA 2.202370e-07    NA
501  0.9557313 0.2377023       NA 5.802385e-05    NA
499  0.9897830 0.2592018       NA 1.342216e-04    NA
504  0.9384720 0.2480936       NA 1.551159e-04    NA
503  0.9490128 0.2766188       NA 6.018926e-04    NA
502  0.8714747 0.2574326       NA 7.111262e-04    NA
226 -0.7793113 0.2447636       NA 1.452877e-03    NA
334  0.8324358 0.2753942       NA 2.505256e-03    NA
335  0.7834660 0.2630524       NA 2.897914e-03    NA
174  0.7349158 0.2485552       NA 3.108994e-03    NA
```

The only information transferred from the `preFitModel` function to `rGLS` is the estimated (co)variance matrix `Mod1$V`. The `corStruc` option specifies the correlation structure to be applied on each random effect. In the example we wish to fit polygenic effects and permanent environmental effects. The former requires a correlatioon structure given by the GRM whereas the permanent environmental effects are iid. Consequently, `corStruc=list( id=list("GRM","Ind") ))` is specfied.

If you would like to fit a GWAS assuming independent observations (for instance if there is one observation per individual and the individuals are unrelated), you can do this as

```
> #Assuming independent observations
> V0 = diag(400)
> GWAS0 <- rGLS(y ~ 1, genabel.data = Gen.Data,
    phenotype.data = Phen.Data, V = V0)
[1] "Rotation matrix ready"
Rotate LMM started
Rotate LMM ready
> summary(GWAS0)
Summary for top 10 results, sorted by P1df
          effB   se_effB chi2.1df         P1df Pc1df
```

```
500  1.2162613 0.0000000       NA 0.000000e+00    NA
501  0.9894738 0.1212485       NA 3.330669e-16    NA
499  1.0908618 0.1366703       NA 1.443290e-15    NA
504  0.9299940 0.1329840       NA 2.685407e-12    NA
502  0.9259355 0.1356279       NA 8.669288e-12    NA
334  0.9146119 0.1431164       NA 1.651446e-10    NA
335  0.8705490 0.1377146       NA 2.592058e-10    NA
256 -0.8682749 0.1406033       NA 6.601542e-10    NA
503  0.8970006 0.1454301       NA 6.919589e-10    NA
226 -0.7720748 0.1261882       NA 9.449507e-10    NA
> #This example shows very large inflation
> print(estlambda(GWAS0@results$P1df, method="median"))
$estimate
[1] 3.862137

$se
[1] NA
```

Note that the data was simulated assuming dependence between observations and if we assume independence the inflation factor becomes much larger than 1 in this example. Hence, this assumption is not suitable.

Using the `simulate_PhenData` function

Suppose for instance we want to simulate 4 observations from each individual and the three variance components (polygenic, permanent env., residual) are 1.

```
> VC.poly <- VC.perm <- VC.res <- 1
> n.obs <- rep(4, nids(Gen.Data))
```

In this example, the object `Gen.Data` is used as input and an additive genetic effect of 2.0 is simulated at the location of SNP number 1000. The phenotype data is then simulated as

```
> Phen.Data <- simulate_PhenData(y ~ 1, genabel.data=Gen.Data,
                                 n.obs=n.obs, SNP.eff=2,
                                 SNP.nr=500,
                                 VC=c(VC.poly, VC.perm,VC.res)
                                 )
>
```

The simulated data can then be fitted as

```
> GWAS.sim1 <- rGLS(y ~ 1, genabel.data = Gen.Data,
    phenotype.data = Phen.Data)
```

Simulating year effects using the simulate_PhenData function

Year effects can be added to the simulated response after running the simulate_PhenData function.

```
> Phen.Sim <- simulate_PhenData(y ~ 1,
    genabel.data = Gen.Data, n.obs = n.obs, SNP.eff = 2,
    SNP.nr = 500, VC = c(VC.poly, VC.perm, VC.res), beta = c(0,1))
> ####### PRODUCE YEAR EFFECTS FOR EACH INDIVIDUAL
> sd.year = 1 #Standard Deviation of Year Effects
> beta <- rnorm(max(n.obs), 0, sd.year) #Simulated Year Effects
> year.effects <- years <- NULL
> for (i in 1:length(n.obs)) {
      yr.i <- sort(sample(1:max(n.obs), n.obs[i]))
      years <- c(years, yr.i)
            year.effects <- c(year.effects, beta[yr.i])
    }
> ########################
> #### A FUNCTION TO ADD A VARIABLE TO A LIST
> add.var <- function(x, add.new, new.name) {
      x[[length(x) + 1]] <- add.new
            names(x)[length(x)] <- new.name
            return(x)
    }
> ########################
> #ADDS A NEW PHENOTYPE WITH YEAR EFFECTS ADDED
> Phen.Sim <- add.var(Phen.Sim, Phen.Sim$y + year.effects, "y.yrs")
> #ADD YEAR AS FACTOR
> Phen.Sim <- add.var(Phen.Sim, as.factor(years), "Years")
> ########################
> #RUN THE ANALYSIS
> GWAS.sim1 <- rGLS(y.yrs ~ Years, genabel.data = Gen.Data,
    phenotype.data = Phen.Sim)
```

Simulating binary data using the simulate_PhenData function

Binary data can be simulated using a threshold model where a Gaussian response is first simulated and all values greater than a specified threshold τ are 1's on the observed scale. The binary data can be simulated and analyzed using the following code.

```
> Phen.Sim <- simulate_PhenData(y ~ 1,
    genabel.data = Gen.Data, n.obs = n.obs, SNP.eff = 2,
    SNP.nr = 500, VC = c(VC.poly, VC.perm, VC.res), beta = c(0,1))
> tau = 0
> Phen.Sim$y_observed <- as.numeric(Phen.Sim$y > tau)
```

```
> GWAS.sim1 <- rGLS(y_observed ~ 1, genabel.data = Gen.Data,
    phenotype.data = Phen.Sim)
```