Supplementary information

Our focus in this section will be in *R* and compared to *SAS*. Some comments on Bayesian methods will be given at later part of this section.

R implementation

The **kinship** package was used for kinship calculation, linear mixed model as with mixed Cox models. The package was originally implemented in *S-PLUS* and ported to *R* as described in [61]. Some recent initatives have been made to improve the facilities for handling sparse matrices, various tools for family data including pedigree drawing as with kinship calculation, and mixed effects Cox model, so the original **kinship** package was partitioned into three separate packages called **bdsmatrix**, **kinship2** and **coxme**. The **pedigreemm** package [21] is appropriate for modeling polygenes within the GLMM framework.

Kinship calculation

A function *makefamid* from **kinship** will generate a "pedigree" type, which can be used by function *makekinship* to obtain kinship matrices from different families,

```
library(kinship)
pid <- with(fam, makefamid(ID, FA, MO))
kmat <- with(fam, makekinship(pid, ID, FA, MO))
```

Note that with GWAS this only needs to be done once and does not have a big overhead. Interestingly, the models for a collection of monozygotic (MZ) and dizygotic (DZ) twins can be treated as a special case. A model using an exchangeable correlation, say, will not be so desirable compared to those using the kinship information[11]. Consider a study of nMZ MZ and nDZ DZ twins, we can order that data such that MZ twins precede their DZ counterpart, then function *bdsmatrix* is called to generate the kinship matrix to be used by *glm* for a sporadic model or *lmekin* for a linear mixed model.

[11] For twin data, to account for the relationship between twin pairs one can pragmatically specify the correlation structure. In one study of physical activity, we order twin pairs by zygosity such that MZ precede DZ twins in the data, we can then subject the data for analysis with following code.

```
idn <- 1:(2*(nMZ+nDZ))
kmat <- bdsmatrix(rep(2,nMZ+nDZ),c(rep(0.5,3*nMZ),
                      rep(c(0.5,0.25,0.5),nDZ)),
                      dimnames=list(idn,idn))

glmfit <- glm(paee ~ walkability + age + weight + sex)
summary(glmfit)
kfit <- lmekin(paee ~ walkability + age + weight + sex,
               random = ~1|id, varlist=list(kmat))
kfit
```

There are a number of other packages available, e.g., **gap** and **identity**.

Linear mixed model

```
library(kinship)
kkin <- lmekin(Q1 ~ SEX + AGE + SMOKE,
        data=pg, random = ~1|ID, varlist=list(kmat))
kibd <- lmekin(Q1 ~ SEX + AGE + SMOKE,
        data=pg, random = ~1|ID,
        varlist=list(kmat,ibdmat))
```

Generalized linear mixed model

```
library(pedigreemm)
bt <- pedigreemm(AFFECTED ~ SEX + AGE + SMOKE + (1|ID),
      data=pg, family="binomial"(link="logit"),
      pedigree=list(ID=ped))
```

Multivariate model

The package **multic** [46] has facility for multivariate analysis, however, it was bound to particular environments. In principle, this is a computing problem that can be fixed.

Marginal models

A notable implementation is *R* **gee** package.

```
library(gee)
m1 <- gee(Q1 ~ SEX + AGE + SMOKE, id=pid,
          data=pg, corstr="exchangeable")
summary(m1)
```

To ensure maximum compatibility with the GLMM fit, the scale parameter is chosen to be fixed at the default value of 1. The structure "exchangeable" assumes equal correlations between relatives in a pedigree but in principle this could be modified to use kinship matrix as in *SAS* below.

Mixed Cox models

This is a Cox model with correlated frailty.

```
library(kinship)
kcox <- coxme(Surv(age, AFFECTED) ~ SEX + SMOKE,
        data=pg, random = ~1|ID, varlist=list(kmat))
```

More information is available from the package vignette.

kinship2 and coxme

As of 14 March 2012, the current version of **kinship** at CRAN will be archived. This has been due to a recent development which involves splitting the package into three separate ones, namely **bdsmatrix**, **kinship2**, and **coxme**. One only expects a slight change from a user's perspective, e.g., the way to specify random effects associated with **coxme**. Although this also involves *lmekin*, here only examples with respect to Cox model are given. For the Framingham data contained in **nf**, the diabetes status and age onset were available and the modeling syntax is as follows,

```
library(kinship2)
attach(nf)
f <- makefamid(shareid, fshare, mshare)
k <- makekinship(f, shareid, fshare, mshare)
detach(nf)
library(coxme)
print("Cox model with random intercept")
f1 <- coxme(Surv(agediab, diabetes) ~ sex + (1|pedno), nf)
```

```
f1
print("Cox model with random intercept and additive variance")
k2 <- 2*as.matrix(k)
f2 <- coxme(Surv(agediab, diabetes) ~ sex + (1|shareid), nf,
            varlist=coxmeMlist(k2, rescale=FALSE))
f2
```

The standard Cox model provides a baseline to compare. Note that **kinship2** depends on **Matrix** so k2 is created for *coxme*.

Suppose we intend to read output **k.dat** from **PLINK**, we can use the following code,

```
k <- read.table("k.dat",header=TRUE)
library(bdsmatrix)
attach(k)
ID <- unique(c(IID1,IID2))
t1 <- cbind(IID1,IID2,PI_HAT)
t2 <- cbind(IID1=ID,IID2=ID,PI_HAT=0.5)
trio <- rbind(t1,t2)
k2 <- bdsmatrix.ibd(trio)
detach(k)
save(k,k2,file="k.RData")
```

Note that we add the diagonal elements in the kinship matrix, which can be loaded with *load*(“k.RData”).

regress and MASS

After submission of the paper, we learned about the work in a similar but alternative context [62]. It turned out that the associated package **regress** yielded comparable results to *lmekin* from **kinship** (data not shown).

We have also become aware of the possibility to use *glmmPQL* available from **MASS** [63] and it appears straightforward to use the *corSymm* function in **nlme** to construct a correlation for data on twins and affected sib pairs as input for the *correlation* option, but for data containing general pedigrees this is more involved and we had limited experience of success.

SAS implementation

SAS has used G and R to indicate the variance-covariance matrices associated with random and fixed effects. The procedures of interest in *SAS* are MIXED, GLIMMIX, NLMIXED. GLIMMIX is an extension to both PROC GENMOD and PROC MIXED.

When individuals in a pedigree is ordered appropriately, the specification should be as follows,

```
proc inbreed data=families covar outcov=kmat;
     var id fa mo;
run;
```

The following block is for the polygenic model.

```
title kinship only;
proc mixed data=pheno covtest asycov noclprint;
     class id;
     model q1=sex age smoke SNP / noint solution covb;
     random id / type=lin(1) ldata=kmat solution;
run;
proc glimmix data=pheno asycov method=mmpl;
     class id;
     model affected(event='1')=sex age smoke SNP
           / dist=binary link=logit solution covb;
     random id / type=lin(1) ldata=kmat solution;
     random _residual_;
run;
```

By default, PROC MIXED employs REML method. For PROC GLIMMIX, maximum or restricted maximum likelihood approach was applied to a pseudo-likelihood (PL) in the sense that a linearization is applied, leading to abbreviations such as M_PL and R_PL, where _ can be subject-specific (S) expansion where linearization is carried out about the current estimate or β and U, or a marginal (M) expansion where the linearization is about a current estimate of β and $E(U) = 0$. If method=QUAD is specified, an adaptive Gauss-Hermite quadrature is used.

The following block is for both oligogenic and polygenic effects

```
title kinship and ibd;
proc mixed data=pheno covtest asycov noprofile;
     class id;
     model q1=sex age smoke / noint solution covb;
     random id / type=lin(2) ldata=kibd solution;
run;
proc glimmix data=pheno asycov method=mmpl;
     class id;
     model affected(event='1')=sex age smoke
           / dist=binary link=logit solution covb;
     random id / type=lin(2) ldata=kibd solution;
     random _residual_;
run;
```

Therefore the method of estimation here is maximum marginal pseudo-likelihood. As can be seen, the second part has extended those available from *R* and the statement "random _residual_" also allows for overdispersion.

For Cox model, one can take advantage of the PHREG procedure with RANDOM statement to specify a shared frailty model which can be compared with a model using ID statement to identify clusters.

Finally, one can specify the relationship as *R* part of the variance-covariance matrix as follows,

```
proc mixed data=pheno sandwich;
     class id;
     model q1=sex age smoke / noint;
     repeated / type=lin(1) ldata=kmat sub=pid;
run;
```

Where the PROC statement specifies the data set to be analyzed using a sandwich estimator, MODEL the statistical model, REPEATED the *R* matrix incorporating kinship information.

Multivariate implementations

Simulation and estimation for a tri-variate normal data

We simulated 500 samples under a tri-variate normal $N(\mu, \Sigma)$ with

$$\mu = \begin{pmatrix} 1 \\ 2 \\ 3 \end{pmatrix} \quad \text{and} \quad \Sigma = \begin{pmatrix} 10 & 1 & 2 \\ 1 & 20 & 3 \\ 2 & 3 & 50 \end{pmatrix}$$

The simulation and estimation are furnished as follows,

```
library(regress)
library("MASS")
set.seed(12345)
n <- 500
m <- c(1,2,3)
S <- matrix(c(10,1,2, 1,20,3, 2,3,50),3,3)
Y <- mvrnorm(n,m,S)
y <- as.vector(t(Y))
c <- kronecker(rep(1,n),diag(1,3))
V1 <- matrix(c(1,0,0, 0,0,0, 0,0,0),3,3,byrow=TRUE)
V2 <- matrix(c(0,1,0, 1,0,0, 0,0,0),3,3,byrow=TRUE)
V3 <- matrix(c(0,0,0, 0,1,0, 0,0,0),3,3,byrow=TRUE)
V4 <- matrix(c(0,0,1, 0,0,0, 1,0,0),3,3,byrow=TRUE)
V5 <- matrix(c(0,0,0, 0,0,1, 0,1,0),3,3,byrow=TRUE)
V6 <- matrix(c(0,0,0, 0,0,0, 0,0,1),3,3,byrow=TRUE)
id <- as.vector(t(cbind(1:n,1:n,1:n)))
s1 <- kronecker(diag(1,n),V1)
s2 <- kronecker(diag(1,n),V2)
s3 <- kronecker(diag(1,n),V3)
s4 <- kronecker(diag(1,n),V4)
s5 <- kronecker(diag(1,n),V5)
s6 <- kronecker(diag(1,n),V6)
results <- regress(y˜c-1,˜s1+s2+s3+s4+s5+s6,pos=c(1,0,1,0,0,1),
           identity=FALSE,start=c(10,1,20,1,1,30))
apply(Y,2,mean)
cov(Y)
```

which produces results as follows,

```
Likelihood kernel: K = c1+c2+c3

Maximized log likelihood with kernel K is  -3041.732

Linear Coefficients:
    Estimate Std. Error
 c1    0.891      0.144
 c2    2.026      0.201
 c3    3.592      0.313

Variance Coefficients:
    Estimate Std. Error
 s1   10.313      0.653
 s2    1.313      0.649
 s3   20.241      1.281
 s4    3.476      1.017
 s5    2.881      1.414
 s6   48.863      3.093

> apply(Y,2,mean)
[1] 0.8908874 2.0262134 3.5922123
> cov(Y)
          [,1]      [,2]      [,3]
[1,] 10.312879  1.313217  3.475876
[2,]  1.313217 20.240644  2.881214
[3,]  3.475876  2.881214 48.863036
```

Through package **regress** we obtained

$$\hat{X} = \begin{pmatrix} 0.89 \\ 2.03 \\ 3.59 \end{pmatrix} \quad \text{and} \quad S = \begin{pmatrix} 10.31 & 1.31 & 3.48 \\ 1.31 & 20.24 & 2.88 \\ 3.48 & 2.88 & 48.86 \end{pmatrix}$$

agreeing with the simulated data. We now turn to the GAW17 data using a multivariate model for Q1, Q2, Q4, with a bit of simplication over covariance specification,

```
library(foreign)
pheno <- read.dta("pheno2.dta")
```

```
kid <- read.csv("kmat.csv")
k <- as.matrix(kid[,-1])

library(regress)
library("MASS")

n <- 697
v1 <- v2 <- v3 <- v4 <- v5 <- v6 <- matrix(0,3,3)
v1[1,1] <- 1
v2[1,2] <- v2[2,1] <- 1
v3[2,2] <- 1
v4[1,3] <- v4[3,1] <- 1
v5[2,3] <- v5[3,2] <- 1
v6[3,3] <- 1
s1 <- kronecker(v1,k)
s2 <- kronecker(v2,k)
s3 <- kronecker(v3,k)
s4 <- kronecker(v4,k)
s5 <- kronecker(v5,k)
s6 <- kronecker(v6,k)
c <- kronecker(rep(1,n),diag(1,3))
id <- as.vector(t(cbind(1:n,1:n,1:n)))
results <- regress(q~-1+c+sex+age+smoke,~s1+s2+s3+s4+s5+s6,
                   identity=FALSE,pos=c(1,0,1,0,0,1),
                   start=c(5.546, 2.999, 3.940, -1.260, -0.780, 0.680),
                   data=pheno)
results
```

The results are given as follows,

```
Likelihood kernel: K = c1+c2+c3+sex+age+smoke

Maximized log likelihood with kernel K is  -1393.867

Linear Coefficients:
       Estimate Std. Error
 c1       0.565      0.108
 c2       0.531      0.109
 c3       0.526      0.109
 sex     -0.005      0.043
 age     -0.013      0.001
 smoke   -0.019      0.051

Variance Coefficients:
       Estimate Std. Error
    s1    4.219      0.227
    s2   -0.103      0.166
    s3    4.542      0.244
    s4    0.601      0.178
    s5   -0.108      0.183
    s6    5.115      0.275
```

SAS implementation

We first revisit the simulated data generated above. Assuming the Y and indicator c are stored in dataset *mv* while the coefficient matrices are stored in *mv_ldata*, then the appropriate syntax in *SAS* is as follows,

```
proc mixed data=mv covtest asycov noclprint;
     class id c;
     model q=c / noint solution;
     random c*id / type=lin(6) ldata=mv_ldata;
run;
```

Although *SAS* complains about *Convergence criteria met but final hessian is not positive definite*, it turns out that the estimats are fairely close.

```
            Covariance Parameter Estimates

                         Standard         Z
Cov Parm     Estimate       Error     Value          Pr Z

LIN(1)         9.3328      0.6529     14.30        <.0001
LIN(2)         1.3133      0.6494       2.02       0.0432
LIN(3)        19.2612      1.2814     15.03        <.0001
LIN(4)         3.4760      1.0169       3.42       0.0006
LIN(5)         2.8813      1.4138       2.04       0.0415
LIN(6)        47.8845      3.0936     15.48      <.0001
Residual       0.9797           0         .           .


            Asymptotic Covariance Matrix of Estimates

Cov Parm    CovP1    CovP2    CovP3    CovP4    CovP5    CovP6

LIN(1)     0.4262  0.05428 0.006913   0.1437  0.01830  0.04843
LIN(2)    0.05428   0.4218   0.1065  0.06870   0.1486  0.04015
LIN(3)   0.006913   0.1065   1.6421  0.01517   0.2338  0.03328
LIN(4)     0.1437  0.06870  0.01517   1.0341   0.1487   0.6808
LIN(5)    0.01830   0.1486   0.2338   0.1487   1.9988   0.5643
LIN(6)    0.04843  0.04015  0.03328   0.6808   0.5643   9.5704

-2 Res Log Likelihood          8853.4
AIC (smaller is better)        8867.4

                     Solution for Fixed Effects

                           Standard
Effect   c    Estimate        Error      DF    t Value    Pr > |t|

c        1      0.8909       0.1436    1497       6.20      <.0001
c        2      2.0262       0.2012    1497      10.07      <.0001
c        3      3.5922       0.3126    1497      11.49      <.0001


       Type 3 Tests of Fixed Effects

         Num     Den
Effect    DF      DF    F Value    Pr > F

c          3    1497      76.12    <.0001
```

We now return to the GAW17 data. With the same spcification of Q1, Q2, and Q4 in a single outcome, along with a variable c corresponding to particular traits, the GLMMIX counterpart is as follows,

```
title kinship and multivariate;
proc mixed data=pheno2 covtest asycov noclprint;
     class id c;
     model q=c sex age smoke / noint solution covb;
     random c*id / type=lin(6) ldata=ldata solution;
run;
```

Note that in addition to a comparable estimate to the *R* implementation, the *REPEATED /group=c* statement also adds trait-specific residual variances. Furthermore, *ldata* contains the coefficient matrix generated from kinship matrix *kmat* via the following code,

```
proc iml;
     use kmat;
     read all var _num_ into kmat;
     k=0;
     do i=1 to 3;
        do j=1 to i;
           j3=j(3,3,0);
           j3[i,j]=1;
           j3[j,i]=1;
           v=j3@kmat;
           k=k+1;
           vp=v||j(nrow(v),1,k);
           if k=1 then vps=vp;
           else vps=vps//vp;
        end;
     end;
     create vps from vps;
     append from vps;
     close vps;
quit;
libname x '.';
data x.ldata;
     set vps (rename=(col2092=parm));
     by parm;
     if first.parm then row=1;
     else row+1;
run;
```

By default variance components can have lower boundary constraint of 0, in cases this is not so one can use the PARMS statement, e.g., for the multivariate example as

```
parms / lowerb=1e-4,.,1e-4,.,.,1e-4,1e-4,1e-4,1e-4;
```

which informs the procedure to use default values (.) as lower boundaries for the the covariances while 0.0001 for the variances.

BUGS[12]

The BUGS (Bayesian inference Using Gibbs Sampling) project is concerned with flexible software for the Bayesian analysis of complex statistical models using Markov chain Monte Carlo (MCMC) methods. Initiatives have been made to make it available to Windows and other platforms and link with the *R* project.

Analysis of family data has been described but we could not access the source code associated with [26]. According to [25], the models we have described can be straightforwardly implemented in software such as *WinBUGS* but the implementation still requires founders precede their offsprings though it is not necessary to do so with *R* in general and *SAS* for the examples used here. It remains to explore the possibility to combine ideas in those implementations. However, analysis via WINBUGS is expected to be slower.

[12] See `http://www.mrc-bsu.cam.ac.uk/bugs/welcome.shtml`