Supplementary Material I

R script to estimate the power to detect a gene (rs174575) by environment (breastfeeding/milk) effect in a dataset with related individuals with longitudinal measures. Due to the extensive missingness, the power is calculated by means of data simulation, i.e., we calculate the observed or empirical power based on the likelihood ratio test over many replications. To reduce computational burden, the background covariance matrix, i.e., conditional on the fixed effects, was fixed to its expected (true) value in optimizing the likelihood with and without the effects of interest (e.g., the interaction).

In the first part of the script several functions are created that are necessary to simulate and analyze the data, namely a loglikelihood function, a function to standardize data in a matrix and a multivariate normal density function.

Then, the number of datasets to simulate is specified as well as the data structure. The user can indicate the number of monozygotic (MZ) and dizygotic (DZ) twin families and the patterns of missingness that occur in the dataset (individuals from a family that have no data at all, and individuals that have no data at a particular time point).

Then, the MAF of the candidate gene, the frequency of breastfeeding and the correlation of breastfeeding across family members are specified as well as the effect sizes of the candidate gene, breast feeding and interaction effect expressed in terms of proportion of variance explained.

In addition, the alpha level and the background covariance matrices (covariance within person across time, covariance within MZ twin pairs, DZ twin pairs and twin-sibling pairs) can be specified.

Next, genotype and breastfeeding data are simulated. Then, phenotypic data are simulated given a particular genetic model (additive, dominant or recessive) and the effect sizes specified in the model. Subsequently, the simulated dataset is analyzed using the loglikelihood function and the statistical output is collected.

```
rm(list=ls(all=TRUE))
#
# semfitg is a function that calculates the derivatives of the loglikelihood
# function with respect to the parameters
# b0 (intercept), b1 (gene), b2 (milk), and b3 (gene x milk):
#
semfitg=function(par,alldatm,alldat,zyg,
nmz,ndz, eff, npar,smz,sdz) {
#
nfam=nmz+ndz
np=1 # parameter 4
gp=rep(0,1+sum(eff))
g0=g1=g2=g3=0
b0=par[np]
b1=b2=b3=0
if (eff[1]==1) {np=np+1; b1=par[np]}
if (eff[2]==1) {np=np+1; b2=par[np]}
if (eff[3]==1) {np=np+1; b3=par[np]}
#print(c(b1,b2,b3))
mus=matrix(0,nfam,22)
mus[,1:5]=(b1*alldat[,2]+b2*alldat[,5]+b3*alldat[,2]*alldat[,5])
mus[,6:10]=(b1*alldat[,3]+b2*alldat[,6]+b3*alldat[,3]*alldat[,6])
mus[,11:15]=(b1*alldat[,4]+b2*alldat[,7]+b3*alldat[,4]*alldat[,7])
mus=mus+b0
#
# derives ----------------------
for (i in 1:nfam) {
if (i==(ndz+1) | i==1) { test=rep(-1,15) }
cnew=(sum(test==alldatm[i,8:22]))==15
keep=which(alldatm[i,8:22]==1)
keep2=keep+7
n1=sum(alldatm[i,8:12]==1)
```

```
n2=sum(alldatm[i,13:17]==1)
n3=sum(alldatm[i,18:22]==1)
#
mum=as.vector(mus[i,keep])
if (zyg[i]==0 & !cnew) {
stmpi=solve(sdz[keep,keep]);
} #
if (zyg[i]==1 & !cnew) {
stmpi=solve(smz[keep,keep]) #
}
dldm=stmpi%*%as.matrix((alldat[i,keep2]-mum))
g0=g0-sum(dldm)
#
if (eff[1]==1) {
tmp=c(rep(alldat[i,2],n1),rep(alldat[i,3],n2),rep(alldat[i,4],n3))
g1=g1-sum(dldm*tmp)
}
if (eff[2]==1) {
tmp=c(rep(alldat[i,5],n1),rep(alldat[i,6],n2),rep(alldat[i,7],n3))
g2=g2-sum(dldm*tmp)
}
if (eff[3]==1) {
tmp=c(rep(alldat[i,5]*alldat[i,2],n1),
rep(alldat[i,6]*alldat[i,3],n2),
rep(alldat[i,7]*alldat[i,4],n3))
g3=g3-sum(dldm*tmp)
}
test=alldatm[i,8:22]
}
np=1
gp[np]=g0
if (eff[1]==1) {np=np+1; gp[np]=g1}
if (eff[2]==1) {np=np+1; gp[np]=g2}
if (eff[3]==1) {np=np+1; gp[np]=g3}
# print(gp)
gp
}
#
# The semfit function calculates the loglikelihood function
# We seek values of b0, b1, b2, b3 to maximize this function
# Once maximized the values of b0 b1 b2 b3 are maximumlikelihood estimates
#
semfit=function(par,alldatm,alldat,zyg,
nmz,ndz, eff, npar,smz,sdz) {
#
nfam=nmz+ndz
np=1 # parameter 4
b0=par[np]
b1=b2=b3=0
if (eff[1]==1) {np=np+1; b1=par[np]}
if (eff[2]==1) {np=np+1; b2=par[np]}
if (eff[3]==1) {np=np+1; b3=par[np]}
#print(c(b1,b2,b3))
mus=matrix(0,nfam,22)
mus[,1:5]=(b1*alldat[,2]+b2*alldat[,5]+b3*alldat[,2]*alldat[,5])
mus[,6:10]=(b1*alldat[,3]+b2*alldat[,6]+b3*alldat[,3]*alldat[,6])
mus[,11:15]=(b1*alldat[,4]+b2*alldat[,7]+b3*alldat[,4]*alldat[,7])
mus=mus+b0
#
# log likelihood function ----------------------
logl=0
for (i in 1:nfam) {
if (i==(ndz+1) | i==1) { tmp=rep(-1,15) }
cnew=(sum(tmp==alldatm[i,8:22]))==15
keep=which(alldatm[i,8:22]==1)
keep2=keep+7
mum=as.vector(mus[i,keep])
if (zyg[i]==0 & !cnew) {
```

```
#
sdzkk=as.matrix(sdz[keep,keep])
#
stmpi=solve(sdzkk);
#
#logdet=log(det(sdz[keep,keep]))
logdet=log(det(sdzkk))  # does not work no scalar s[i,i]
#
} #
if (zyg[i]==1 & !cnew) {
#
smzkk=as.matrix(smz[keep,keep])
#
stmpi=solve(smz[keep,keep]) #
#logdet=log(det(sdz[keep,keep]))
logdet=log(det(sdzkk))
#print(logdet)
}
#
# dmv is the multivariate normal density function
#
#if (length(keep2)>1) {
d=dmv(x=alldat[i,keep2],mean=mum,sigma=stmpi,logdet)
#}
#if (length(keep2)==1) {
#d=dnorm(x=alldat[i,keep2],mean=mum,sigma=sqrt(1/stmpi),log=T)
#}
# cannot happen
if (is.nan(d)) {return(100000)}
#
logl=logl+d
#
tmp=alldatm[i,8:22]
}
# print(c(logl,b0,b1,b2,b3))
#
-logl
}
#
# standi is a function to standardize data in a matrix
#
standi=function(dat) {
nr=dim(dat)[1]
nc=dim(dat)[2]
sds=diag(1/apply(dat,2,sd))
ms=t(as.matrix(apply(dat,2,mean)))
dat=(dat-(matrix(1,nr,1)%x%ms))%*%sds
dat
}
#
# dmv is the multivariate normal density function
#
dmv=function (x, mean, sigma, logdet)
{
    if (is.vector(x)) {
        x <- matrix(x, ncol = length(x))
    }
    distval <- mahalanobis(x, center = mean, cov = sigma, inverted=TRUE)
    logretval <- -(ncol(x) * log(2 * pi) + logdet + distval)/2
        return(logretval)
}
# ------------------------------------------------
#
# Start main program
# Specify the number of simulations, the data structure and patterns of missingness
#
library(MASS)
library(mvtnorm)
```

```
#
T=5                       # the number of measurements over time
nmem=3                    # the number of individuals per family
maxny=ny=T*nmem
#
# Specify the number of MZ and DZ twin families, the number of simulations and the
# patterns of missingness
#
selfseed=F
if (selfseed) set.seed(10208)
nrep=10000                        # specify the number of simulations
nmulti=1
nmz0=round(300*nmulti)            # specify the number of monozygotic twin families
ndz0=round(344*nmulti)            # specify the number of dizygotic twin families
#
nmismz=14    # specify the number of missingness patterns for monozygotic twin families
nmisdz=14    # specify the number of missingness patterns for dizygotic twin families
#
# now the patterns of missingness will be specified by use of a missing values
# indicator, 1=observed 0=missing. Each row specifies a particular pattern of
# missingness per family with the following columns: p geno milk t12345 t12345 t12345
#
#     p = proportion of the total dataset with this pattern of missingness
#     geno and milk indicate which individuals in the family have data available
#     geno = geno_twin1 geno_twin2 geno_sib
#     milk = milk_twin1 milk_twin2 milk_sib
#     t12345 indicates at which time points individuals have data available
#     t12345 t12345 t12345, IQ measurements 1-5 twin1, twin2, sib
#
missmz=matrix(c(
# p   geno   milk   t12345     t12345     t12345
.033, 1,0,0, 1,0,0, 1,1,1,1,1, 0,0,0,0,0, 0,0,0,0,0, # sample I –1 twin
.110, 1,1,0, 1,1,0, 1,1,1,1,1, 1,1,1,1,1, 0,0,0,0,0, # sample I -2 twins
.053, 1,1,1, 1,1,1, 1,1,1,1,1, 1,1,1,1,1, 0,0,0,0,1, # sample I –basis
.130, 1,1,0, 1,1,0, 0,0,1,1,0, 0,0,1,1,0, 0,0,0,0,0, # sample II -2 twins
.060, 1,0,0, 1,0,0, 1,0,0,1,0, 0,0,0,0,0, 0,0,0,0,0, # sample III –1 twin
.113, 1,1,0, 1,1,0, 1,0,0,1,1, 1,0,0,1,1, 0,0,0,0,0, # sample III –2 twins - a
.090, 1,1,0, 1,1,0, 1,0,0,1,0, 1,0,0,1,0, 0,0,0,0,0, # sample III –2 twins - b
.023, 1,1,1, 1,1,1, 1,0,0,1,1, 1,0,0,1,1, 0,0,0,1,0, # sample III –basis - a
.030, 1,1,1, 1,1,1, 1,0,0,1,0, 1,0,0,1,0, 0,0,0,1,0, # sample III –basis - b
.010, 1,0,0, 1,0,0, 0,0,0,0,1, 0,0,0,0,0, 0,0,0,0,0, # sample IV –1 twin
.157, 1,1,0, 1,1,0, 0,0,0,0,1, 0,0,0,0,1, 0,0,0,0,0, # sample IV -2 twins
.010, 1,0,0, 1,0,0, 0,0,0,0,1, 0,0,0,0,0, 0,0,0,0,0, # sample V -1 twin
.157, 1,1,0, 1,1,0, 0,0,0,0,1, 0,0,0,0,1, 0,0,0,0,0, # sample V -2 twins
.023, 1,1,1, 1,1,1, 0,0,0,0,1, 0,0,0,0,1, 0,0,0,0,1  # sample V -basis
),nmismz,22,byrow=T)
missdz=matrix(c(
# p   geno   milk   t12345     t12345     t12345
.012, 1,0,0, 1,0,0, 1,1,1,1,1, 0,0,0,0,0, 0,0,0,0,0, # sample I –1 twin
.160, 1,1,0, 1,1,0, 1,1,1,1,1, 1,1,1,1,1, 0,0,0,0,0, # sample I -2 twins
.073, 1,1,1, 1,1,1, 1,1,1,1,1, 1,1,1,1,1, 0,0,0,0,1, # sample I –basis
.148, 1,1,0, 1,1,0, 0,0,1,1,0, 0,0,1,1,0, 0,0,0,0,0, # sample II -2 twins
.015, 1,0,0, 1,0,0, 1,0,0,1,0, 0,0,0,0,0, 0,0,0,0,0, # sample III –1 twin
.076, 1,1,0, 1,1,0, 1,0,0,1,1, 1,0,0,1,1, 0,0,0,0,0, # sample III –2 twins - a
.105, 1,1,0, 1,1,0, 1,0,0,1,0, 1,0,0,1,0, 0,0,0,0,0, # sample III –2 twins - b
.061, 1,1,0, 1,1,0, 1,0,0,0,0, 1,0,0,0,0, 0,0,0,0,0, # sample III –2 twins - c
.023, 1,1,1, 1,1,1, 1,0,0,1,1, 1,0,0,1,1, 0,0,0,1,0, # sample III -basis
.049, 1,0,0, 1,0,0, 0,0,0,0,1, 0,0,0,0,0, 0,0,0,0,0, # sample IV –1 twin
.090, 1,1,0, 1,1,0, 0,0,0,0,1, 0,0,0,0,1, 0,0,0,0,0, # sample IV -2 twins
.020, 1,0,0, 1,0,0, 0,0,0,0,1, 0,0,0,0,0, 0,0,0,0,0, # sample V -1 twin
.137, 1,1,0, 1,1,0, 0,0,0,0,1, 0,0,0,0,1, 0,0,0,0,0, # sample V -2 twins
.032, 1,1,1, 1,1,1, 0,0,0,0,1, 0,0,0,0,1, 0,0,0,0,1  # sample V -basis
),nmisdz,22,byrow=T)
#
nmz=sum(round(missmz[,1]*nmz0)) # just in case ndz .ne. ndz0
ndz=sum(round(missdz[,1]*ndz0)) # just in case ndz .ne. ndz0
#
p=maf=.258                        # specify MAF of rs174575
```

```
pmilk=pm=.409                    # specify probability of being breastfed
#
# correlation of breastfeeding within family members.
# matrix: twin1 - twin2 - sib
#
dzmilk=mzmilk=matrix(c(
1,.997,.71,
.997,1,.71,
.71,.71,1),3,3,byrow=T)
#
# specify the effect sizes of the main effects and the interaction effect
# expressed as proportion of variance explained
#
mainmilk=0.04764                 # specify the effect size of breastfeeding
maincan=0.00002415               # specify the effect size of rs174575
milkxcan=0.00265                 # specify the effect size of the interaction effect
#
alpha=.05                        #specify alpha
#
pu=matrix(0,nrep,5)
#
rtot=1-(mainmilk+maincan+milkxcan) # residual variance
#
b0=0 # intercept.
b1=sqrt(mainmilk)
b2=sqrt(maincan)
b3=sqrt(milkxcan)
# ---------------------------------------------------
#
# specify the background covariance structure
#
# background correlations within the individual across time as observed in the
# current dataset.
#
Sph=matrix(c(
1.0000,0.5984,0.5782,0.5174,0.4599,
0.5984,1.0000,0.6915,0.6086,0.6446,
0.5782,0.6915,1.0000,0.7360,0.7216,
0.5174,0.6086,0.7360,1.0000,0.7181,
0.4599,0.6446,0.7216,0.7181,1.0000)*rtot,5,5)
#
# background correlations across dizygotic twins across time as observed in the
# current dataset.
#
Sdz2=matrix(c(
0.5487,0.3444,0.3204,0.2909,0.2338,
0.3444,0.4112,0.3282,0.3089,0.2440,
0.3204,0.3282,0.4336,0.4234,0.3297,
0.2909,0.3089,0.4234,0.4772,0.3738,
0.2338,0.2440,0.3297,0.3738,0.3632)*rtot,5,5)
#
# background correlations across monozygotic twins across time as observed in the
# current dataset.
#
Smz2=matrix(c(
0.6782,0.5711,0.5445,0.4899,0.4121,
0.5711,0.5541,0.6243,0.5557,0.4900,
0.5445,0.6243,0.7677,0.7167,0.6786,
0.4899,0.5557,0.7167,0.7998,0.6573,
0.4121,0.4900,0.6786,0.6573,0.7148)*rtot,5,5)
#
# background correlations across sibling-twin pairs across time as observed in the
# current dataset.
#
Ssib2=matrix(c(
0.2,0.2,0.2,0.2921,0.1424,
0.2,0.2,0.2,0.2,0.3453,
0.2,0.2,0.2,0.2,0.2440,
```

```
0.2,0.2,0.2,0.1569,0.1445,
0.2,0.2,0.2,0.1445,0.3488)*rtot,5,5)
#
# total 15 x 15 variance/covariance matrix in MZ families
#
Smz=rbind(
cbind(Sph,t(Smz2),t(Ssib2)),
cbind(Smz2,Sph,t(Ssib2)),
cbind(Ssib2,Ssib2,Sph))
#
# total 15 x 15 variance/covariance matrix in DZ families
#
Sdz=rbind(
cbind(Sph,t(Sdz2),t(Ssib2)),
cbind(Sdz2,Sph,t(Ssib2)),
cbind(Ssib2,Ssib2,Sph))
#
# now simulate genotypes for family members, start from parental genotypes
#
nmember=nmem2=nmem+2 # plus parents.
q=1-p
gf=c(p^2,2*p*q,q^2) # genotype freqs
pf=outer(gf,gf) # parent uncorrelated candidate.
#
nfamdz=ndz
nfammz=nmz
nfam=ndz+nmz #
nsib=nmem=3
nf=round(pf*nfam)
zyg=c(rep(0,nfamdz),rep(1,nfammz))
psib=array(0,c(3,3,3))
# A is decreasing its freq is MAF = p
#
# conditional offspring genotype frequencies
#
psib[1,1,1:3]=c(1,.0,.0) # parents AA=3 & AA=3, offspring AA
psib[1,2,1:3]=c(.5,.5,.0) # parents AA and Aa, offspring AA (.5) and Aa (.5)
psib[1,3,1:3]=c(.0,1,0) # parents AA and aa, offspring Aa
psib[2,1,1:3]=c(.5,.5,.0) # parents Aa and AA, offspring AA (.5) Aa (.5)
psib[2,2,1:3]=c(.25,.5,.25) # etc.
psib[2,3,1:3]=c(.0,.5,.5)
psib[3,1,1:3]=c(.0,1,.0)
psib[3,2,1:3]=c(.0,.5,.5)
psib[3,3,1:3]=c(.0,.0,1) # parents aa & aa, offspring aa
dose=c(3,2,1)
dose0=dose-1
dosep=c(1,2,3)
#
#specify the genetic model 'dom' or 'rec' or 'add'
#
gact='rec'
#
for (irep in 1:nrep) {
print(irep)
genos=matrix(0,nfam,nmem)
genosp=matrix(0,nfam,2)
#
# parental genotypes
#
genosp[,1]=sample(dosep,nfam,replace=T,prob=gf) # father
genosp[,2]=sample(dosep,nfam,replace=T,prob=gf) # mother
#
# offspring genotypes
#
for (i in 1:nfam) {
ig1=genosp[i,1] #*-1+4
ig2=genosp[i,2] #*-1+4
#
```

```
# conditional probabilities given parental genotypes
#
ptmp=psib[ig1,ig2,1:3]
#
# offspring genotypes
#
genos[i,1:nmem]=sample(dose,nsib,replace=T,prob=ptmp)
}
#
# mz twins are identical
#
for (i in 1:nfam) {
if (zyg[i]==1) {genos[i,2]=genos[i,1]}
}
#
# simulate the variable milk
#
milkdat=matrix(1,nfam,nmem)
tmp=mvrnorm(nfam,mu=rep(0,nmem),Sigma=dzmilk)
thresh=qnorm(pmilk)
milkdat[tmp>thresh]=0     # no milk
#
# collect the data in a matrix
#
alldat=matrix(0,nfam,22) # zyg m1-m3 g1-g3 t1-t5 t1-t5 t1-t5
#
alldat[,1]=zyg
alldat[,2:4]=milkdat
if (gact=='dom') genos[genos==3]=2  # 1,2=3
if (gact=='rec') {  # 1=2, 3 recoded 1,2
genos[genos==2]=1;
genos[genos==3]=2;
}
alldat[,5:7]=genos[,1:3]-1 # coded 0,1
#
#  background covariance structure
#
alldat[zyg==1,8:22]=mvrnorm(nmz,mu=rep(0,maxny),Sigma=Smz)
alldat[zyg==0,8:22]=mvrnorm(ndz,mu=rep(0,maxny),Sigma=Sdz)
#
# effect sizes main effects and milk X candidate gene
#
# ... b0+alldat[,2]*b1+alldat[,5]*b2+alldat[,2]*alldat[,5]*b3
#
alldat[,2:7]=standi(alldat[,2:7])
# add effects
alldat[,8:12]=alldat[,8:12]+
b0+alldat[,2]*b1+alldat[,5]*b2+alldat[,2]*alldat[,5]*b3
alldat[,13:17]=alldat[,13:17]+
b0+alldat[,3]*b1+alldat[,6]*b2+alldat[,3]*alldat[,6]*b3
alldat[,18:22]=alldat[,18:22]+
b0+alldat[,4]*b1+alldat[,7]*b2+alldat[,4]*alldat[,7]*b3
#
# now get missing indicators of each case
#
misnmz1=diffinv(round(missmz[,1]*nmz0))
misndz1=diffinv(round(missdz[,1]*ndz0))  + nmz
#
# the matrix for the missing indicator
#
alldatm=matrix(0,nfam,22) # zyg m1-m3 g1-g3 t1-t5 t1-t5 t1-t5
#
# introduce missingness
#
for (i in 1:nmismz) {
blockmz=round(missmz[i,1]*nmz0) # cvd 1 nov
tmp=matrix( missmz[i,2:22] , blockmz, 21, byrow=T)
alldatm[ (misnmz1[i]+1):(misnmz1[i+1]),2:22]=tmp
```

```
}
#
# missingness info is now in alldatm
#
for (i in 1:nmisdz) {
blockdz=round(missdz[i,1]*ndz0)  # cvd 1 nov + correction maria
tmp=matrix( missdz[i,2:22] , blockdz, 21, byrow=T)
alldatm[(misndz1[i]+1):(misndz1[i+1]),2:22]=tmp
}
# now fit the model using maximum likelihood (ML) estimation and a fixed background
# covariance structure
#
par=c()
npar=1
par[npar]=b0 # bo
eff=c(1,1,1)
if (eff[1]==1) {npar=npar+1; par[npar]=b1} # b1 gene
if (eff[2]==1) {npar=npar+1; par[npar]=b2} # b2 milk
if (eff[3]==1) {npar=npar+1; par[npar]=b3} # b3 gene X milk
#
# ---------- ML estimation 4 parameters
#
fit1=T
fit2=T
pchi=-1
if (fit1) {
res1=optim(par,semfit,semfitg,method="L-BFGS-B",lower = -Inf, upper =
Inf,alldatm=alldatm,alldat=alldat,zyg=zyg,
nmz=nmz,ndz=ndz,eff=eff,npar=npar,smz=Smz,sdz=Sdz)
} # fit1
#
par=c()
npar=1
par[npar]=b0 # bo
eff=c(1,1,0)
if (eff[1]==1) {npar=npar+1; par[npar]=b1} # b1
if (eff[2]==1) {npar=npar+1; par[npar]=b2} # b2
if (eff[3]==1) {npar=npar+1; par[npar]=b3} # b3
#
# ---------- ML estimation 3 parameters drop b3
#
if (fit2) {
res2=optim(par,semfit,semfitg,method="L-BFGS-B",lower = -Inf, upper =
Inf,alldatm=alldatm,alldat=alldat,zyg=zyg,
nmz=nmz,ndz=ndz,eff=eff,npar=npar,smz=Smz,sdz=Sdz)
}
#
# likelihood ratio test of dropped parameter
#
if (fit1 & fit2) {
chi2=-2*(res1$value-res2$value)
pchi=pchisq(chi2,1,lower=F)
}
ures11=lm(alldat[,8]~alldat[,2]*alldat[,5],data=as.data.frame(alldat))
ures12=lm(alldat[,8]~alldat[,2]+alldat[,5],data=as.data.frame(alldat))
ures11vs2=anova(ures12,ures11)
ures21=lm(alldat[,13]~alldat[,3]*alldat[,6],data=as.data.frame(alldat))
ures22=lm(alldat[,13]~alldat[,3]+alldat[,6],data=as.data.frame(alldat))
ures21vs2=anova(ures22,ures21)
ures31=lm(alldat[,18]~alldat[,4]*alldat[,7],data=as.data.frame(alldat))
ures32=lm(alldat[,18]~alldat[,4]+alldat[,7],data=as.data.frame(alldat))
ures31vs2=anova(ures32,ures31)
#
# collect p values
#
pu[irep,1]=ures11vs2[6][2,1]
pu[irep,2]=ures21vs2[6][2,1]
pu[irep,3]=ures31vs2[6][2,1]
```

```
pu[irep,4]=pchi
pu[irep,5]=chi2
#
} # next replication
# show empirical power given alpha
#
epower=apply(pu[,1:4]<alpha,2,mean)
epower
```