#Simulate a dataset

#In R:###############################################################

Nsnps=1000

Nind=1000

H2=0.25

MAFlim=0.05

seed=110585

setwd("/Users/dancrouch/Documents/Work/Teaching/GMS")

library(bindata);library(Matrix);library(parallel)

set.seed(seed)

sig=matrix(rexp(Nind^2, rate = 20),ncol=Nind);sig=sig+t(sig)

sig[sig>1]=1

diag(sig)=1

f=rbeta(Nsnps,shape1=0.8,shape2=0.8)

f[f<=MAFlim]=MAFlim+f[f<=MAFlim]

f[(1-f)<=MAFlim]=f[(1-f)<=MAFlim]-MAFlim

g=matrix(nrow=Nind,ncol=Nsnps)

for(x in 1:Nind){

if(x==1){g[x,]=rbinom(Nsnps,p=f,size=1)}else{

g[x,]=(g[x-1,]==1)\*rbinom(Nsnps,p=sig[x-1,x]+f\*(1-sig[x-1,x]),size=1)+(g[x-1,]==0)\*rbinom(Nsnps,p=f\*(1-sig[x-1,x]),size=1)  
 }

}

geno1=g

g=matrix(nrow=Nind,ncol=Nsnps)

for(x in 1:Nind){

if(x==1){g[x,]=rbinom(Nsnps,p=f,size=1)}else{

g[x,]=(g[x-1,]==1)\*rbinom(Nsnps,p=sig[x-1,x]+f\*(1-sig[x-1,x]),size=1)+(g[x-1,]==0)\*rbinom(Nsnps,p=f\*(1-sig[x-1,x]),size=1)

}

}

geno2=g

genoM=geno1+geno2

betas=rnorm(Nsnps,mean=0,sd=sqrt(H2/Nsnps))

genoScaled=scale(genoM,center=T,scale=T)

pheno=genoScaled%\*%betas+rnorm(Nind,mean=0,sd=sqrt(1-H2))

pheno=scale(pheno,center=T,scale=T)

write.table(as.matrix(cbind(pheno,genoM)),file="/Users/dancrouch/Documents/Work/Teaching/GMS/GMS\_GenomeWideAnalysis\_dataset.txt",col.names=F,row.names=F,quote=F)

#End of code#########################################################

#James-Stein estimation

#In R:###############################################################

setwd("/Users/dancrouch/Documents/Work/Teaching/GMS")

data=read.table("/Users/dancrouch/Documents/Work/Teaching/GMS/GMS\_GenomeWideAnalysis\_dataset.txt",header=F)

pheno=data[,1];geno=as.matrix(data[,-1])

gwas<-function(x){

model=lm(y~x,data=data.frame(y=pheno,x=x))

return(summary(model)$coefficients[2,3])

}

zStats=apply(X=geno,FUN=gwas,MARGIN=2)

tiff("/Users/dancrouch/Desktop/Rplot.tiff")

hist(zStats)

dev.off()

write.table(as.numeric(zStats),file="GMS\_GenomeWideAnalysis\_zStats.txt",quote=F,row.names=F,col.names=F)

p=apply(genoM,FUN=mean,MARGIN=2)/2

p[p>0.5]=1-p[p>0.5]

write.table(as.numeric(p),file="GMS\_GenomeWideAnalysis\_MAF.txt",quote=F,row.names=F,col.names=F)

#End of code#########################################################

#For practical (effect size estimation)

#In R:###############################################################

setwd("/Users/dancrouch/Documents/Work/Teaching/GMS")

zStats=read.table("GMS\_GenomeWideAnalysis\_zStats.txt",header=F)[,1]

p=read.table("GMS\_GenomeWideAnalysis\_MAF.txt",header=F)[,1]

A=var(zStats)-1

#Note - can also use (better) estimator in Efron and Hastie

M=mean(zStats)

est=M+(zStats-M)\*A/(1+A)

tiff("/Users/dancrouch/Desktop/Rplot1.tiff")

hist(est)

dev.off()

tiff("/Users/dancrouch/Desktop/Rplot2.tiff")

plot(est,zStats)

dev.off()

s=1/sqrt(1000\*2\*p\*(1-p))

est\_mafCorrected=est\*s

tiff("/Users/dancrouch/Desktop/Rplot3.tiff")

hist(p)

dev.off()

tiff("/Users/dancrouch/Desktop/Rplot4.tiff")

plot(y=est\_mafCorrected,x=zStats)

dev.off()

#Can the students work out what the pattern here reflects?

#If there's time: predict phenotype using JS estimator and ML estimator - see what differences are.

#End of code#########################################################

#For practical (Mixed Model)

#In R:###############################################################

library(mvtnorm);library(numDeriv);library(MASS)

setwd("/Users/dancrouch/Documents/Work/Teaching/GMS")

data=read.table("/Users/dancrouch/Documents/Work/Teaching/GMS/GMS\_GenomeWideAnalysis\_dataset.txt",header=F)

geno=data[,-1]

pheno=data[,1]

tiff("Rplot1.tiff")

hist(pheno)

dev.off()

p=colMeans(geno)/2

genoScaled=as.matrix(scale(geno,center=2\*p,scale=sqrt(2\*p\*(1-p))))

#Or genoScaled=as.matrix(scale(geno,center=T,scale=T))

n=dim(geno)[1];N=dim(geno)[2]

GRM=genoScaled%\*%t(genoScaled)/N

#Diagonal will not be exactly 1 due to individuals covarying with themselves slightly (inbreeding)

covariates=rep(0,N)

#constrOptim minimises by default, so set to negative to 'maximise'

NegLogLikelihood<-function(x){

-sum(dmvnorm(pheno,mean=x[3]+x[4]\*covariates,sigma=x[1]\*GRM+x[2]\*diag(n),log=TRUE))

}

ui=matrix(0,nrow=4,ncol=4)

ui[1,1]=1;ui[2,1]=-1

ui[3,2]=1;ui[4,2]=-1

ci=c(0,-1,0,-1)

modelFit=constrOptim(theta=c(0.1,0.9,0,0),f=NegLogLikelihood,grad=NULL,ui=ui,ci=ci)

#Confidence intervals

FisherInfo=hessian(func=NegLogLikelihood,x=modelFit$par)

SE=sqrt(diag(ginv(FisherInfo)))

CI\_95perc=cbind(modelFit$par-1.96\*SE,modelFit$par,modelFit$par+1.96\*SE)

#End of code#########################################################

#Mixed-model GWAS

#In R:###############################################################

library(mvtnorm);library(numDeriv);library(MASS)

setwd("/Users/dancrouch/Documents/Work/Teaching/GMS")

data=read.table("/Users/dancrouch/Documents/Work/Teaching/GMS/GMS\_GenomeWideAnalysis\_dataset.txt",header=F)

geno=data[,-1]

pheno=data[,1]

tiff("Rplot1.tiff")

hist(pheno)

dev.off()

p=colMeans(geno)/2

genoScaled=as.matrix(scale(geno,center=2\*p,scale=sqrt(2\*p\*(1-p))))

#Or genoScaled=as.matrix(scale(geno,center=T,scale=T))

n=dim(geno)[1];N=dim(geno)[2]

GRM=genoScaled%\*%t(genoScaled)/N

#Diagonal will not be exactly 1 due to individuals covarying with themselves slightly (inbreeding)

covariates=rep(0,N)

#constrOptim minimises by default, so set to negative to 'maximise'

NegLogLikelihood<-function(x){

-sum(dmvnorm(pheno,mean=x[3]+x[4]\*covariates,sigma=x[1]\*GRM+x[2]\*diag(n),log=TRUE))

}

ui=matrix(0,nrow=4,ncol=4)

ui[1,1]=1;ui[2,1]=-1

ui[3,2]=1;ui[4,2]=-1

ci=c(0,-1,0,-1)

modelFit=constrOptim(theta=c(0.1,0.9,0,0),f=NegLogLikelihood,grad=NULL,ui=ui,ci=ci)

sig2U=modelFit$par[1];sig2E=modelFit$par[2]

NegLogLikelihoodAlt<-function(x){

sig2E\_b=sig2E-x[2]^2\*var(covariates)

-sum(dmvnorm(pheno,mean=x[1]+x[2]\*covariates,sigma=sig2U\*GRM+sig2E\_b\*diag(n),log=TRUE))

}

NegLogLikelihoodNull=-sum(dmvnorm(pheno,mean=rep(mean(pheno),n),sigma=sig2U\*GRM+sig2E\*diag(n),log=TRUE))

test<-function(i){

print(i)

covariates<<-geno[,i]

#constrOptim minimises by default, so set to negative to 'maximise'

modelFitAlt=optim(par=c(0,0),fn=NegLogLikelihoodAlt,gr=NULL)

Chisq=2\*(NegLogLikelihoodNull-modelFitAlt$value)

return(c(Chisq,modelFitAlt$par[2]))

}

results=lapply(X=1:20,FUN=test)

results=matrix(unlist(results),ncol=2,byrow=TRUE)

#compare with GWAS results

zStats=read.table("GMS\_GenomeWideAnalysis\_zStats.txt",header=F)[,1]

p=read.table("GMS\_GenomeWideAnalysis\_MAF.txt",header=F)[,1]

beta=zStats\*sqrt(2\*p\*(1-p))/sqrt(n)

tiff("/Users/dancrouch/Desktop/Rplot.tiff")

plot(beta[1:20],results[,2])

dev.off()

#End of code#########################################################

#predict phenotype

#In R:###############################################################

testSet=1:100

library(mvtnorm);library(numDeriv);library(MASS)

setwd("/Users/dancrouch/Documents/Work/Teaching/GMS")

data=read.table("/Users/dancrouch/Documents/Work/Teaching/GMS/GMS\_GenomeWideAnalysis\_dataset.txt",header=F)

geno=data[,-1]

pheno=data[,1]

tiff("Rplot1.tiff")

hist(pheno)

dev.off()

p=colMeans(geno)/2

genoScaled=as.matrix(scale(geno,center=2\*p,scale=sqrt(2\*p\*(1-p))))

#Or genoScaled=as.matrix(scale(geno,center=T,scale=T))

n=dim(geno)[1];N=dim(geno)[2]

GRM=genoScaled%\*%t(genoScaled)/N

#Diagonal will not be exactly 1 due to individuals covarying with themselves slightly (inbreeding)

covariates=rep(0,N)

#constrOptim minimises by default, so set to negative to 'maximise'

NegLogLikelihood<-function(x){

-sum(dmvnorm(pheno[-testSet],mean=x[3]+x[4]\*covariates[-testSet],sigma=x[1]\*GRM[-testSet,-testSet]+x[2]\*diag(n-length(testSet)),log=TRUE))

}

ui=matrix(0,nrow=4,ncol=4)

ui[1,1]=1;ui[2,1]=-1

ui[3,2]=1;ui[4,2]=-1

ci=c(0,-1,0,-1)

modelFit=constrOptim(theta=c(0.1,0.9,0,0),f=NegLogLikelihood,grad=NULL,ui=ui,ci=ci)

sig2U=modelFit$par[1];sig2E=modelFit$par[2]

predBreedingValues=sig2U\*GRM[testSet,-testSet]%\*%solve(GRM[-testSet,-testSet]\*sig2U+diag(n-length(testSet))\*sig2E)%\*%pheno[-testSet]

#Plot against actual phenotypes...

#What's correlation between breeding values and phenotype? Should be roughly equal to heritabilty if things are working well

#End of code#########################################################