Supplementary S2: Scripts

Paulino Pérez-Rodríguez & Gustavo de los Campos

Box S1: Multi-trait GBLUP with unstructured and structured covariances

```
#(Continued from Box 1)

#Lower triangular elements are stored in column major order in text
#files, each row corresponds to a realization of a covariance matrix

#Read file "UN_R.dat"
R<-read.table(file="UN_R.dat",header=FALSE)

#check number of rows/columns in matrix R, must be 4
rows <- (-1 + sqrt(1 + 8 * ncol(R)))/2
rows

#Trace and density plots, for some elements e.g, R[1,1], R[4,3]

RowColumnToLinear<-function(n,i,j){
   (n+1)*j-j*(j+1)/2-(n-i)
}

par(mfrow=c(2,2))

whichCol<-RowColumnToLinear(rows,1,1)
plot(R[,whichCol],type="b",main="Trace plot",
     ylab=expression(R[11]),xlab="Thinned iteration")
hist(R[501:1000,whichCol],freq=FALSE,main="Density",
     xlab=expression(R[11]))

whichCol<-RowColumnToLinear(rows,4,3)

plot(R[,whichCol],type="b",main="Trace plot",
     ylab=expression(R[43]),xlab="Thinned iteration")

hist(R[501:1000,whichCol],freq=FALSE,main="Density",
     xlab=expression(R[43]))

#Read file "UN_Omega_1.dat"
Omega<-read.table(file="UN_Omega_1.dat",header=FALSE)

#Trace and density plots for some elements, e.g, Omega[3,2], Omega[4,4]

par(mfrow=c(2,2))

whichCol<-RowColumnToLinear(rows,3,2)
```

```
plot(Omega[,whichCol],type="b",main="Trace plot",
     ylab=expression(Omega[32]),xlab="Thinned iteration")
hist(Omega[501:1000,whichCol],freq=FALSE,
     main="Density",xlab=expression(Omega[32]))

whichCol<-RowColumnToLinear(rows,4,4)

plot(Omega[,whichCol],type="b",main="Trace plot",
     ylab=expression(Omega[44]),xlab="Thinned iteration")

hist(Omega[501:1000,whichCol],freq=FALSE,main="Density",
     xlab=expression(Omega[44]))

#See file S3 for resulting figures
```

Box S2a: Multi-trait GBLUP with structured covariance matrices, recursive and diagonal

```
#(continued from Box 2)

#Read Psi
#each row contains the diagonal elememts from PSI
Psi<-read.table(file="REC_DIAG_PSI_1.dat",header=FALSE)

#Trace and density plots for some elements, e.g, Psi[1,1], Psi[3,3]

par(mfrow=c(2,2))

plot(Psi[,1],type="b",main="Trace plot",
     ylab=expression(Psi[11]),xlab="Thinned iteration")
hist(Psi[501:1000,1],freq=FALSE,main="Density",xlab=expression(Psi[11]))

plot(Psi[,3],type="b",main="Trace plot",
     ylab=expression(Psi[33]),xlab="Thinned iteration")
hist(Psi[501:1000,3],freq=FALSE,main="Density",xlab=expression(Psi[33]))

#W
#Only entries set to TRUE in M1 are saved in a row vector
#in consecutive order
W<-read.table(file="REC_DIAG_W_1.dat")

#Trace and density plots for some elements, e.g, W[3,2], W[4,2],
#W[4,3]

par(mfrow=c(3,2))

plot(W[,1],type="b",main="Trace plot",
     ylab=expression(W[32]),xlab="Thinned iteration")
hist(W[501:1000,1],freq=FALSE,main="Density",
     xlab=expression(W[32]))

plot(W[,2],type="b",main="Trace plot",
     ylab=expression(W[42]),xlab="Thinned iteration")
hist(W[501:1000,2],freq=FALSE,main="Density",
```

```
     xlab=expression(W[42]))

plot(W[,3],type="b",main="Trace plot",
     ylab=expression(W[43]),xlab="Thinned iteration")
hist(W[501:1000,3],freq=FALSE,main="Density",
     xlab=expression(W[43]))

#See file S3 for resulting figures
```

Box S2b: Multi-trait GBLUP with structured covariance matrices, factor analytic and diagonal

```
#(continued from Box 2)

#Regression coefficients for FA
#Only entries set to TRUE in M1 are saved in a row vector
#in consecutive order
W<-read.table(file="FA_DIAG_W_1.dat")

#Trace and density plots for some elements, e.g, W[2,1], W[3,1],
#W[4,1]

par(mfrow=c(3,2))

plot(W[,1],type="b",main="Trace plot",
     ylab=expression(W[21]),xlab="Thinned iteration")
hist(W[501:1000,1],freq=FALSE,main="Density",
     xlab=expression(W[21]))

plot(W[,2],type="b",main="Trace plot",
     ylab=expression(W[31]),xlab="Thinned iteration")
hist(W[501:1000,2],freq=FALSE,main="Density",
     xlab=expression(W[31]))

plot(W[,3],type="b",main="Trace plot",
     ylab=expression(W[41]),xlab="Thinned iteration")
hist(W[501:1000,3],freq=FALSE,main="Density",
     xlab=expression(W[41]))

#See file S3 for resulting figures
```

Box S3a: Unstructured covariances for markers, pedigree and residual

```
library(BGLR)
data(wheat)

#Compute genomic relationship matrix
M<-scale(wheat.X,center=TRUE)
K1<-tcrossprod(M)/ncol(M)
#Relationship matrix derived from pedigree
K2<-wheat.A

#Define linear predictor
```

```
ETA1<-list(mar=list(K=K1,model="RKHS"),
           ped=list(K=K2,model="RKHS"))

#Fit model
set.seed(1)
fm1<-Multitrait(y=wheat.Y,ETA=ETA1,nIter=10000,burnIn=5000,
                saveAt= "m1_",verbose=FALSE)

#Estimated residual covariance matrix
fm1$resCov

#Estimated Omega_1, equivalent to fm1$ETA[[1]]$Cov
fm1$ETA$mar$Cov

#Estimated Omega_2, equivalent to fm1$ETA[[2]]$Cov
fm1$ETA$ped$Cov

#Predicted u_1
fm1$ETA$mar$u

#Predicted u_2
fm1$ETA$ped$u

#Estimated intercept
fm1$mu
```

Box S3b: FA for markers + Recursive for pedigree + unstructured residual

```
#Continued from Box S3a
#Define covariance structure for G_1
M1<-matrix(TRUE,nrow=4,ncol=1)
Cov1<-list(type="FA",M=M1)

#Define covariance structure for G_2
M2 <- matrix(nrow = 4, ncol = 4, FALSE)

#Adding recursion from trait 2 onto traits 3 and 4
M2[3, 2] <- TRUE
M2[4, 2] <- TRUE

#Adding recursion from trait 3 onto trait 4
M2[4, 3] <- TRUE
Cov2<-list(type="REC",M=M2)

ETA3<-list(mar=list(K=K1,model="RKHS",Cov=Cov1),
           ped=list(K=K2,model="RKHS",Cov=Cov2))

Res<-list(type="DIAG")

#Fit the model
set.seed(3)
fm3<-Multitrait(y=wheat.Y,ETA=ETA3,nIter=10000,burnIn=5000,
                resCov=Res,saveAt= "m3_",verbose=FALSE)
```

```
#Retrieving results
#Estimated Omega_2
fm3$ETA$ped$Cov

#Estimated W_2
fm3$ETA$ped$Cov$W

#Estimated PSI_2
fm3$ETA$ped$Cov$PSI
```

Box S4a: Simulating three traits using the mice data set

```
library(BGLR)

set.seed(195021)

data(mice)
nTraits<-3
nMrk<-1000
nQTL<-12
QTL<-floor(seq(from=10,to=nMrk-9,length=nQTL))

B<-matrix(nrow=nMrk,ncol=nTraits,0)
b<-runif(min=.5,max=1,n=nQTL)
B[QTL[1:6],1]<-b[1:6]
B[QTL[4:9],2]<-b[4:9]
B[QTL[7:12],3]<-b[7:12]

cols<-floor(seq(from=1,to=10000,length=nMrk))
X<-scale(mice.X[,cols],scale=F,center=T)
U<-X%*%B
G0<-cov(U) # realized genomic variance
R0<-diag(diag(G0)*c(9,19,9)) # scales to generate h2
R0[2,1]<-R0[1,2]<-0.5*sqrt(R0[1,1]*R0[2,2])
R0[3,1]<-R0[1,3]<-0.3*sqrt(R0[1,1]*R0[3,3])
R0[3,2]<-R0[2,3]<-0.1*sqrt(R0[2,2]*R0[3,3])

n<-nrow(X)
E<-matrix(nrow=n,ncol=3,rnorm(n*3))%*%chol(R0)

Y<-U+E
INT<-c(120,30,40)
for(i in 1:ncol(Y)){
   #adds intercetps
   Y[,i]<-Y[,i]+INT[i]
}
# Realized heritabilities
apply(FUN=var,X=U,MARGIN=2)/apply(FUN=var,X=Y,MARGIN=2)
```

Box S4b: Extracting estimates, and producing posterior plots and summaries for Box 4

```
# (continued from box 4a)

par(mfrow=c(3,1))

for(i in 1:3){
    lab<-expression(paste("P[",b[j]!=0, "|data]"))
    main<-paste("Trait",i)
    col<-ifelse(fmSS$ETA[[1]]$d[,i]>=0.8,2, "skyblue")
    pch<-ifelse(fmSS$ETA[[1]]$d[,i]>=0.8,19,1)
    plot(fmSS$ETA[[1]]$d[,i],ylim=0:1,ylab=lab,xlab="SNP",
          col=col, main=main,pch=pch)
    abline(h=0.8,col=8,lty=2)
    abline(v=QTL[1:6+(i-1)*3],lty=2)
 }
```

Box S4c: Examining posterior probabilities of inclusion within a region

```
# Reading samples of effects saved in binary files
 B=readBinMatMultitrait('ETA_1_beta.bin')

# posterior probability of inclusion by SNPs (nearby QTL2)
 colMeans(B[,99:101,1]!=0)

# posterior probability that at least one of the 3 has effect !=0
 mean(apply(X=B[,99:101,1]!=0,MARGIN=1,FUN=any))

# trace plot by SNP
#(the plot shows that when one SNP is active, the other two are not)
 par(mfrow=c(3,1))
 plot(B[,99,1],cex=.5,col=4,type='o')
 plot(B[,100,1],cex=.5,col=4,type='o')
 plot(B[,101,1],cex=.5,col=4,type='o')
```

Box S4d: A more general approach to examine posterior probability of inclusion by region

```
library(BGData)

# Reading samples of effects saved in binary files
 B=readBinMatMultitrait('ETA_1_beta.bin')
 B=B[-(1:200),,1]# effects for trait 1 removing burn-in

 # ETA[[1]]$d report probabilities of inclusion by SNP and trait
  # checking for trait 1
  plot(colMeans(B!=0),fmSS$ETA[[1]]$d[,1])

 # Identifying segments with elevated probability of inclusion
 SEGMENTS=segments(chr=rep(1,ncol(B)),
              bp=1:ncol(B),
              statistic=1-fmSS$ETA[[1]]$d[,1],# local FDR
              threshold=0.7,gap=3)
 SEGMENTS
```

```
 # Plot
   plot(fmSS$ETA[[1]]$d[,1],cex=.5,col=4)
   points(x=QTL[1:6],y= fmSS$ETA[[1]]$d[QTL[1:6],1],col=2)
   abline(v=SEGMENTS[,'start'],col=8,lty=2)
   abline(v=SEGMENTS[,'end'],col=8,lty=2)

 # Computing joint probabilities of inclusion for each discovery
 SEGMENTS=cbind(SEGMENTS,segment_prob=NA)
 for(i in 1:nrow(SEGMENTS)){
   chunk=SEGMENTS$start[i]:SEGMENTS$end[i]
   SEGMENTS$segment_prob[i]=mean(apply(FUN=any,MARGIN=1,X=B[,chunk,drop=FALSE]!=0))
 }
```

Box S4e: univariate analysis using BayesC

```
set.seed(123)

par(mfrow=c(3,1))

for(i in 1:3)
{
  saveAt= paste("SSUni_",i,"_",sep="")
    fmSSUni<-BGLR(y=Y[,i],ETA=list(list(X=X,model="BayesC")),
                  nIter=12000,burnIn=2000,saveAt=saveAt,
                  verbose=FALSE)
    lab<-expression(paste("P[",b[j]!=0, "|data]"))
    main<-paste("Trait",i)
    col<-ifelse(fmSSUni$ETA[[1]]$d>=0.8,2, "skyblue")
    pch<-ifelse(fmSSUni$ETA[[1]]$d>=0.8,19,1)
    plot(fmSSUni$ETA[[1]]$d,ylim=0:1,ylab=lab,xlab="SNP",
         col=col, main=main,pch=pch)
    abline(h=0.8,col=8,lty=2)
    abline(v=QTL[1:6+(i-1)*3],lty=2)
}
```

Box S5: Estimating Genetic (co)variances in Spike Slab Models (run Box S4a first)

```
#Auxiliary functions

#Genetic covariance matrix
#See Cheng et al., 2018, page 95
#svar sum of variance of columns of the predictors
#if the predictors are centered and standardized
#by columns is equal to number of columns

covBeta<-function(d,Omega,traits,svar){
   Q<-matrix(NA,nrow=traits,ncol=traits)
   for(i in 1:traits){
     Q[i,i]<-Omega[i,i]*sum(d[,i]==1)/nrow(d)
   }

   for(i in 1:traits){
```

```
     for(j in 1:traits){
       if(j<i){
        Q[i,j]<-Q[j,i]<-Omega[i,j]*sum(d[,i]==1 & d[,j]==1)/nrow(d)
       }
      }
   }
   Q<-svar*Q
   return(Q)
}

getG0i<-function(Z,Bi){
    U<-Z%*%Bi
    G0i<-cov(U)
    return(G0i[row(G0i)>=col(G0i)])
}

getG0<-function(X,B){
        q<-dim(B)[3]
        G<-t(apply(FUN=getG0i,X=B,Z=X,MARGIN=1))
        return(G)
}

#Fitting model
Z<-scale(X,center=TRUE,scale=TRUE)/sqrt(ncol(X))

nIter<-35000; burnIn=5000; thin=10
fm<-Multitrait(y=Y,ETA=list(list(X=Z,model='SpikeSlab',
               saveEffects=TRUE,saveIndicators=TRUE)),
               nIter=nIter,burnIn=burnIn,thin=thin,
               verbose=FALSE)
#Omega
fm$ETA[[1]]$Cov$Omega

#Cov between entries of beta
fm$ETA[[1]]$Cov$Sigma

#Method 2, Lehermeier et al., 2017.
#Genomic Variance Estimates: With or without Disequilibrium #Covariances?

B<-readBinMatMultitrait('ETA_1_beta.bin')
B<-B[-(1:(round(burnIn/thin))),,]

xpnd(colMeans(getG0(Z,B)))

#Method 3, Cheng et al., 2018.
#Sum of variance by columns
svar<-sum(apply(X=Z,MARGIN=2,FUN=var))

#Read Omega matrix
Omega<-read.table(file="Omega_1.dat",header=FALSE)
Omega<-as.matrix(Omega)
Omega<-Omega[-(1:(round(burnIn/thin))),]
d<-readBinMatMultitrait("ETA_1_d.bin.gz",storageMode="single")
```

```
d<-d[-(1:(round(burnIn/thin))),,]

out<-matrix(NA,nrow=nrow(Omega),ncol=ncol(Omega))
for(m in 1:nrow(Omega)){
  O<-xpnd(Omega[m,,drop=TRUE])
  indicator<-d[m,,]
  out[m,]<-vech(covBeta(d[m,,],O,3,svar))
}
xpnd(colMeans(out))
```

Box S6: Missing value patterns and plotting

```
#Run Box S4a first

#Missing values patterns
patterns<-matrix(NA,nrow=7,ncol=ncol(Y))
patterns[1,]<-c(F,F,T)
patterns[2,]<-c(F,T,F)
patterns[3,]<-c(F,T,T)
patterns[4,]<-c(T,F,F)
patterns[5,]<-c(T,F,T)
patterns[6,]<-c(T,T,F)
patterns[7,]<-c(T,T,T)

set.seed(123)
s<-sample(1:7,size=180,replace=TRUE)
index<-sample(1:nrow(Y),size=180,replace=FALSE)

YNa<-Y

for(i in 1:length(s)){
   YNa[index[i],patterns[s[i],]]<-NA
}

#After fitting the model, the objects $missing_records and $patterns
#can be used to extract the prediction and plotting.
#The following code shows how to plot.

#Missing values for trait 3
whichNa3<-fmG$missing_records[fmG$patterns[,3]]
Y[whichNa3,3]          #Observed values
fmG$ETAHat[whichNa3,3] #Predicted values

plot(Y[,3],fmG$ETAHat[,3],
     xlab="Observed value",ylab="Predicted value")
points(Y[whichNa3,3],fmG$ETAHat[whichNa3,3],col="red",pch=19)
legend("bottomright",legend=c("Observed","Missing"),pch=c(1,19),
       col=c("black","red"),bty="n")
```

Box S7: Benchmark for Bayesian Ridge Regression (Gaussian prior) in BGLR

In order to run the benchmark it is necessary to create two files: a) R script to fit the models (e.g. `BRR.R`) and a submission script to submit jobs to the queue (e.g. `BRR_R.sub`). We assume that data are stored in

matrices $\boldsymbol{y}$ with 50000 rows and 4 columns and matrix $\boldsymbol{X}$ with 50000 rows and 50000 columns which are stored in the R file `sampleData.RData`. At the end of the running process the file `times_BRR_R.txt` will contain the running times in seconds for each scenario and replicate. The code in the file `BRR.R` is shown below:

```
 args=(commandArgs(TRUE))
 for(i in seq_along(args)){
      eval(parse(text=args[[i]]))
  }

# reads job ID
 jobID <- as.integer(Sys.getenv("SLURM_ARRAY_TASK_ID", "1"))

#Get sample data, y matrix with 50,000 rows and 4 columns
#X matrix with 50,0000 rows and 50,000 columns
 load('sampleData.RData')

#Load routines for analysis, using version 1.1.0 from github
 library(BGLR)

 p<-c(5000,10000,20000,50000)
 n<-c(5000,10000,20000,50000)
 traits<-c(2,3,4)
 rep<-1:15

 grid<-expand.grid(rep=rep,n=n,p=p,traits=traits)

 p<-as.integer(grid$p[jobID])
 n<-as.integer(grid$n[jobID])
 nTraits<-as.integer(grid$traits[jobID])
 replicate<-as.integer(grid$rep[jobID])

 W<-X[1:n,1:p]
 rm(X)
 gc()
 y<-y[1:n,1:nTraits]

 ETA<-list(list(X=W,model="BRR"))

 bglrDir <- paste0(tempfile(pattern = "BGLR-"), "/") # do not save output
 dir.create(bglrDir, recursive = TRUE, showWarnings = FALSE)

 tmp<-system.time(Multitrait(y=y,ETA=ETA,nIter=1000,burnIn=500,
                             saveAt=bglrDir,verbose=FALSE))[3]
 tmp<-as.numeric(tmp)

 unlink(bglrDir,recursive=TRUE)

 outFile<-paste0("times_BRR_R.txt")
 results<-paste(c(n,p,nTraits,replicate,tmp),collapse=' ')
```

The code in the file `BRR_R.sub` is given below:

```
#!/usr/bin/bash --login
#SBATCH --job-name=R_RR_4T
```

```
#SBATCH --time=4:00:00
#SBATCH --cpus-per-task=4
#SBATCH --mem=70gb
#SBATCH --constraint=intel18
#SBATCH --array=1-720

n=$SLURM_ARRAY_TASK_ID

PATH=/mnt/research/quantgen/projects/BGLR/multitrait/opt/R-4.2.0/bin:$PATH
export PATH
export OPENBLAS_NUM_THREADS=4

# run command
R CMD BATCH --no-save --no-restore '--args jobID='$n BRR.R brr_$n
```

Box S8: Benchmark for Bayesian Ridge Regression (Gaussian prior) using Julia package JWAS

In order to run the benchmark it is necessary to create two files: a) Julia script to fit the models (e.g. `BRR.jl`) and a submission script to submit jobs to the queue (e.g. `BRR_julia.sub`). We assume that data are stored in csv files `y.csv` with 50000 rows and 5 columns (first column with the ids for individuals, file with headers) and `X.csv` with 50000 rows and 50001 columns (first column with the ids for individuals, file with headers). At the end of the running process the file `times_BRR_JWAS.txt` will contain the running times in seconds for each scenario and replicate. The code in the file `BRR.jl` is shown below:

```
#File: BRR.jl
#Benchmark JWAS software
#Requiere: tmpRR folder
#          file times_BRR_JWAS.csv to append results
#          file y.csv
#          file X.csv

#Loading packages
using JWAS
using DataFrames
using CSV
using ArgParse
using LinearAlgebra

#### Command line arguments
s = ArgParseSettings()
@add_arg_table s begin
    "--case"
        help = "case from 1 to 720"
        arg_type=Int
        required=true
end;

parsed_args = parse_args(ARGS, s);

case=parsed_args["case"];
print("case=",case,"\n")

### End of command line arguments

### Setting the number of CPU threads
```

```
BLAS.get_num_threads()

BLAS.set_num_threads(4)

BLAS.get_num_threads()

### End setting the number of CPU threads

#Start reading the data and selecting number of markers and traits based on cases

ps=[5000,10000,20000,50000];
ns=[5000,10000,20000,50000];
ts=[2,3,4];
rs=collect(1:15);
vector = vec(collect(Base.product(rs,ns,ps,ts)));
grid = DataFrame(map(x -> getindex.(vector, x), eachindex(first(vector))));

r=Int(grid[case,1])
n=Int(grid[case,2]);
p=Int(grid[case,3]);
t=Int(grid[case,4]);

print("r=",r,"\n")
print("n=",n,"\n")
print("p=",p,"\n")
print("t=",t,"\n")

pheno=CSV.read("y.csv",DataFrame,delim=",",header=true,missingstrings=["NA"]);

geno=CSV.read("X.csv",DataFrame,delim=",",header=true);

cd("tmpRR")
folder=string("test_",case)
mkdir(folder)
cd(folder)

#add 1 because the first column is animal
geno=geno[:,1:(p+1)];
geno=geno[1:n,:];
genotypes=get_genotypes(geno,quality_control=false,method="RR-BLUP");

pheno=pheno[1:n,:];

start = time();

cnames_pheno=names(pheno);

model_equations=String[]

for i in 2:(t+1)
        print(i,"\n")
        tmp=""
        tmp=string(cnames_pheno[i]," = intercept + genotypes")
```

```
        push!(model_equations,tmp)
end

model_equations=join(model_equations,"; ");

#Build model
model=build_model(model_equations);

out=runMCMC(model,pheno,chain_length=1000,burnin=500)

elapsed= time()-start;
elapsed

#Write results
tmp_folder=pwd()
cd("../../")
output_file=open("times_RR_JWAS.csv","a")
results=string(case,",",n,",",p,",",t,",",r,",",elapsed);
write(output_file,results)
write(output_file,"\n")
close(output_file)
rm(tmp_folder,recursive=true)
```

The code in the file `BRR_julia.sub` is given below:

```
#!/usr/bin/bash --login
#SBATCH --job-name=JWAS_RR_4T
#SBATCH --time=04:00:00
#SBATCH --cpus-per-task=4
#SBATCH --mem=70gb
#SBATCH --constraint=intel18
#SBATCH --array=1-720

PATH=/mnt/research/quantgen/projects/BGLR/multitrait/opt/julia-1.7.2/bin/:$PATH
export PATH

cd $SLURM_SUBMIT_DIR

# run command
julia BRR.jl --case $SLURM_ARRAY_TASK_ID
```