Local Tutorial for GOALS

Emily Winn-Nunez

2023-10-20

Local Tutorial for GOALS

This script demonstrates how to calculate local GOALS values given a data set for each covariate. Specifically, this file (1) shows how to calculate the Gaussian Kernel, (2) how to precompute before calling the function to calculate GOALS and (3) Calculating the GOALS matrix and plotting local GOALS values

NOTE: This script is based on a simple (and small) genetics example where we simulate genotype data for n individuals with p measured genetic variants. We randomly assume that three of the predictors $j = \{23, 24, 25\}$ are causal and have true association with the generated (continuous) phenotype y for half of the samples, and for the other half, we assume that four of the predictors $j = \{22, 23, 24, 25\}$ are causal. We then assume that the j predictive markers explain a fixed $H^2\%$ (phenotypic variance explained; PVE) of the total variance in the response $V(y)$. This parameter H^2 can alternatively be described as a factor controlling the signal-to-noise ratio. The parameter ρ represents the proportion of H^2 that is contributed by additive effects versus interaction effects. Namely, the additive effects make up $\rho\%$, while the pairwise interactions (epistatic effects) make up the remaining $(1 - \rho)\%$.

```
### Load in the C++ BAKR functions ###
sourceCpp("BAKRGibbs.cpp")
#Register cores
cores = detectCores()
registerDoParallel(cores=cores)
```

Data Simulation

This code chunk shows how to simulate the data with different variable importance before combining into one data set.

```
set.seed(11151990)

n = 1e3; p = 25; pve=0.8; rho=0.5;
fn = "Epistatic_Raw Split"

### Define the Number of Causal SNPs
ncausal = 3

### Simulate Synthetic Data ###
maf <- 0.05 + 0.45*runif(p)
X   <- (runif(n*p) < maf) + (runif(n*p) < maf)
X   <- matrix(as.double(X),n,p,byrow = TRUE)
Xmean=apply(X, 2, mean); Xsd=apply(X, 2, sd); Xc=t((t(X)-Xmean)/Xsd)
s=c(23:25)
```

```
Xstar=X

#Marginal Effects Only
Xmarginal=Xstar[,s]
beta1=rep(1,ncausal)
y_marginal=c(Xmarginal%*%beta1)
beta1=beta1*sqrt(pve*rho/var(y_marginal))
y_marginal=Xmarginal%*%beta1

#Pairwise Epistatic Effects
Xepi=cbind(Xstar[,s[1]]*Xstar[,s[3]],Xstar[,s[2]]*Xstar[,s[3]])
beta2=c(1,1)
y_epi=c(Xepi%*%beta2)
beta2=beta2*sqrt(pve*(1-rho)/var(y_epi))
y_epi=Xepi%*%beta2

#Error Terms
y_err=rnorm(n)
y_err=y_err*sqrt((1-pve)/var(y_err))

y=c(y_marginal+y_epi+y_err); #Full Model
colnames(X) = paste("SNP",1:ncol(X),sep="")

#Now four snps contribute, equally in additivity and interaction. Change pve=0.8
### Simulate Synthetic Data ###
maf <- 0.05 + 0.45*runif(p)
X2   <- (runif(n*p) < maf) + (runif(n*p) < maf)
X2   <- matrix(as.double(X2),n,p,byrow = TRUE)
X2mean=apply(X2, 2, mean); X2sd=apply(X2, 2, sd); X2c=t((t(X2)-X2mean)/X2sd)
s=c(22:25)
Xstar=X2

#Marginal Effects Only
Xmarginal=Xstar[,s]
beta1=rep(1,4)            #ncausal here has to be 2.
y2_marginal=c(Xmarginal%*%beta1)
beta1=beta1*sqrt(pve*rho/var(y_marginal))
```

```
## Warning in beta1 * sqrt(pve * rho/var(y_marginal)): Recycling array of length 1 in vector-array arith
##   Use c() or as.vector() instead.
```

```
y2_marginal=Xmarginal%*%beta1

#Pairwise Epistatic Effects
Xepi=cbind(Xstar[,s[1]]*Xstar[,s[3]],Xstar[,s[2]]*Xstar[,s[4]])
beta2=c(1,1)
y2_epi=c(Xepi%*%beta2)
beta2=beta2*sqrt(pve*(1-rho)/var(y2_epi))
y2_epi=Xepi%*%beta2

y2=c(y2_marginal+y2_epi+y_err); #Full Model
colnames(X) = paste("SNP",1:ncol(X),sep="")
```

```
#Combine the two
X=rbind(X,X2)
Xc=rbind(Xc,X2c)
y=c(y,y2)
```

Run GOALS

```
n = dim(X)[1] #Sample size
p = dim(X)[2] #Number of markers or genes
sigma2 = 1e-3

### Find the Approximate Basis and Kernel Matrix; Choose N <= D <= P ###
B = GaussKernel(t(X)); diag(B)=1
A <- B + sigma2*diag(1,nrow=n,ncol=n)
A_svd <- svd(A)
Ainv = nearPD(A_svd$v%*%diag(1/A_svd$d, nrow=n, ncol=n)%*%t(A_svd$u))$mat
Aiy <- Ainv%*%y
BAiy <- B %*% Aiy
IAiB <- (diag(1,nrow=n, ncol=n)-Ainv%*%B)
BIAiB <- B%*%IAiB

#Calculate Delta

delta = ComputeESAFast(X, B, as.vector(Aiy), as.vector(BAiy), cores=cores)
```

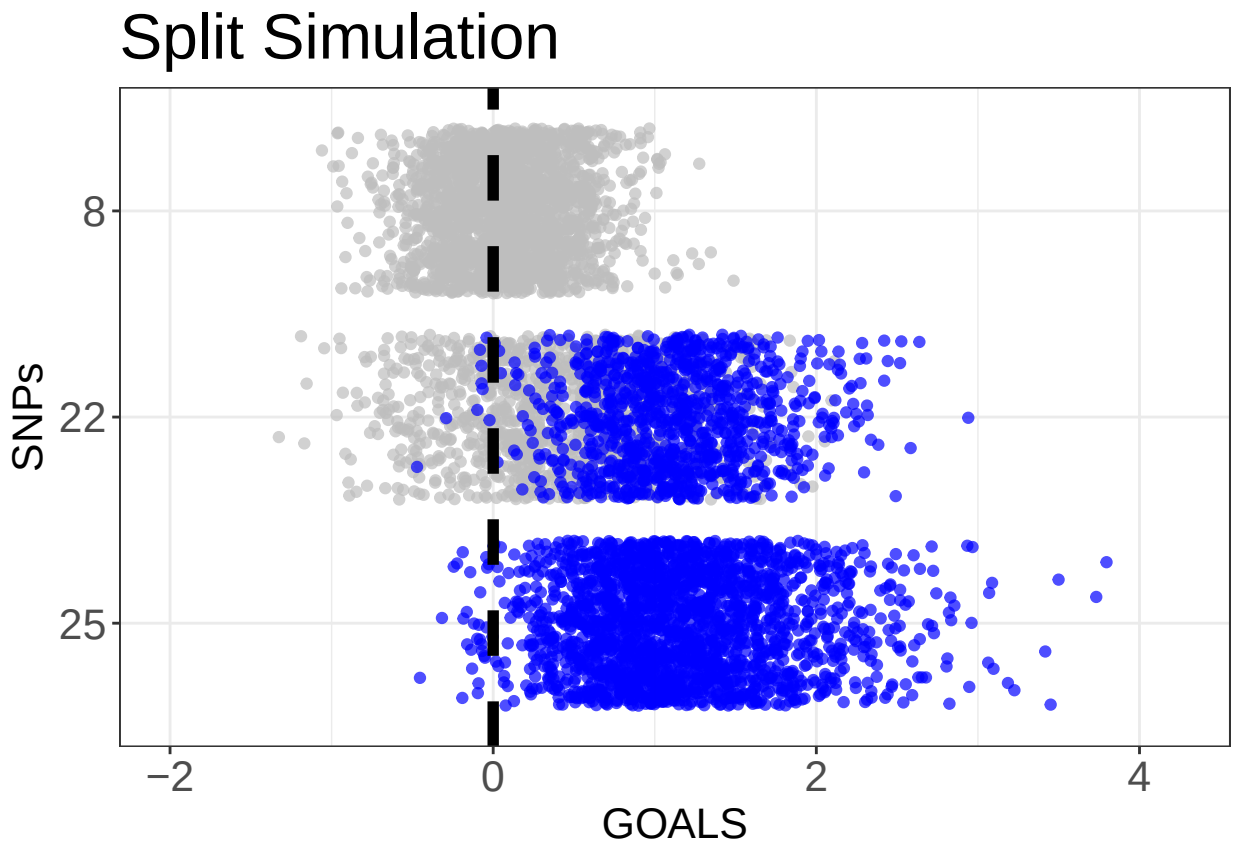
Plot results for select SNPs

Color coding to match causal with blue and noncausal with gray.

```
color_scheme = color=c(rep("gray", 1000), c(rep("blue", 1000)))

ggplot() +
  geom_jitter(data = data.frame("GOALS"=delta[,25]), aes(x="25", y=GOALS), color="blue", alpha=0.7)+
  geom_jitter(data = data.frame("GOALS"=delta[,22]), aes(x="22", y=GOALS), color=color_scheme, alpha=0.
  geom_jitter(data = data.frame("GOALS"=delta[,8]), aes(x="8", y=GOALS), color="gray", alpha=0.7)+
  scale_x_discrete(limits=c("25", "22", "8"))+               #Orders it the way you want.
  geom_hline(yintercept=0, linetype="dashed",color="black", size=2)+
  theme_bw()+
  coord_flip()+                                          #Makes the grid flip
  ggtitle("Split Simulation") +  #Add title here
  xlab("SNPs")+
  ylab("GOALS")+
  theme(axis.text.x=element_text(size=16), axis.text.y = element_text(size=16), axis.title=element_text
  ylim(c(-2,4.25))
```

```
## Warning: Using ‘size‘ aesthetic for lines was deprecated in ggplot2 3.4.0.
## i Please use ‘linewidth‘ instead.
## This warning is displayed once every 8 hours.
## Call ‘lifecycle::last_lifecycle_warnings()‘ to see where this warning was
## generated.
```


Split Simulation
SNPs
8
22
25
−2
0
2
4
GOALS