# SNP-wise Algorithm


## Stuart Brabbs


The log10 Bayes Factor function:

```
log10BF = function(g,y,sigmaa) {
  p=dim(g)[2]
  n=dim(g)[1]
  if (is.null(dim(g)[2])){
  g = g - mean(g)
  y = y - mean(y)
  n=length(g)
  X = g
  invnu = 1/sigmaa^2
  invOmega = invnu + t(X) %*% X
  B = (t(X) %*% cbind(y))/invOmega
  invOmega0 = n
  return(-0.5*log10(det(invOmega)) + 0.5*log10(invOmega0) - log10(sigmaa) - (n/2)*(log10
(t(y- X %*% B) %*% y) - log10(t(y) %*% y)))
  }
  else {
    g = scale(g, scale = FALSE)
    y = scale(y, scale = FALSE)
    X = g
    invnu = diag(rep(1/sigmaa^2, p))
    invOmega = invnu + t(X) %*% X
    B = solve(invOmega, t(X) %*% cbind(y))
    invOmega0 = n
    return(-0.5*log10(det(invOmega)) + 0.5*log10(invOmega0) - p*log10(sigmaa) - (n/2)*(l
og10(t(y- X %*% B) %*% y) - log10(t(y) %*% y - n*mean(y)^2)))
  }
}
```

The general weight function, which returns for each model the log10 posterior score, the posterior model probability (PMP), the log10 normalizing constant, and the log10 Bayes Factor:

```
snpwise_weights = function(X,y,sigmaa,priorpi) {
  models <- c()
  log10post <- c()
  log10bf <- c()

  #get number of parameters
  if (is.null(dim(X)[2])) {
    par <- 1
  }
  else {
    par <- dim(X)[2]
  }

  if (missing(priorpi)){
    p <- par
  }
  else {
    p <- 1/priorpi
  }

  #get all possible parameter combinations
  l <- rep(list(0:1),par)
  combs <- expand.grid(l)
  m <- dim(combs)[1]

  for (i in c(1:m)){
    numlist <- c()
    incl <- c()
    numincl <- 0

    for (j in c(1:par)){
      if (combs[i,j]==1){
        incl <- cbind(incl, X[,j])
        numincl <- numincl + 1
      }
    }
    numlist <- c(numlist, numincl)

    #if only one variable in model
    if (numincl==1) {
      for (k in c(1:par)){
        if (combs[i,k]==1){
          varin <- k
        }
      }
      newmod <- paste0("x", varin)

      #get weight
      prior <- (1/p)*(1-1/p)^(par-1)
      lbf <- log10BF(X[,varin], y, sigmaa)
      log10bf <- c(log10bf, lbf)
      post <- log10(prior) + lbf
    }
```

```
    #if multiple variables in model
    else if (numincl != 0){
      newmod <- ""
      for (j in c(1:par)){

        if (combs[i,j]==1){
          var <- paste0("x", j)
          newmod <- paste(newmod, sep =",", var)
        }
      }
      newmod <- sub('.', '', newmod)
      #get weight
      prior <- (1/p)^numincl * (1-1/p)^(numincl-1)
      lbf <- log10BF(incl, y, sigmaa)
      log10bf <- c(log10bf, lbf)
      post <- log10(prior) + lbf
    }

    #if null model
    else {
      log10bf <- c(log10bf, 0)
      post <- log10((1-1/p)^(p))
      newmod <- "NULL"
    }

    models <- c(models, newmod)
    #add weight to list
    log10post <- c(log10post, post)
  }

  #find log10 normalizing constant
  maxlog10post <- max(log10post)
  log10norm <- maxlog10post + log10(sum(10^(log10post-maxlog10post)))
  wts <- 10^((log10post)-log10norm)
  log10weights <- log10(wts)

  #return dataframe of each model and its posterior weight
  toreturn <- data.frame(models, log10post, wts, log10norm, log10bf)
  names(toreturn)[1] <- "models"
  names(toreturn)[2] <- "log10postscores"
  names(toreturn)[3] <- "pmps"
  names(toreturn)[4] <- "log10norm"
  names(toreturn)[5] <- "log10bayesfactor"

  return(toreturn)
}
```

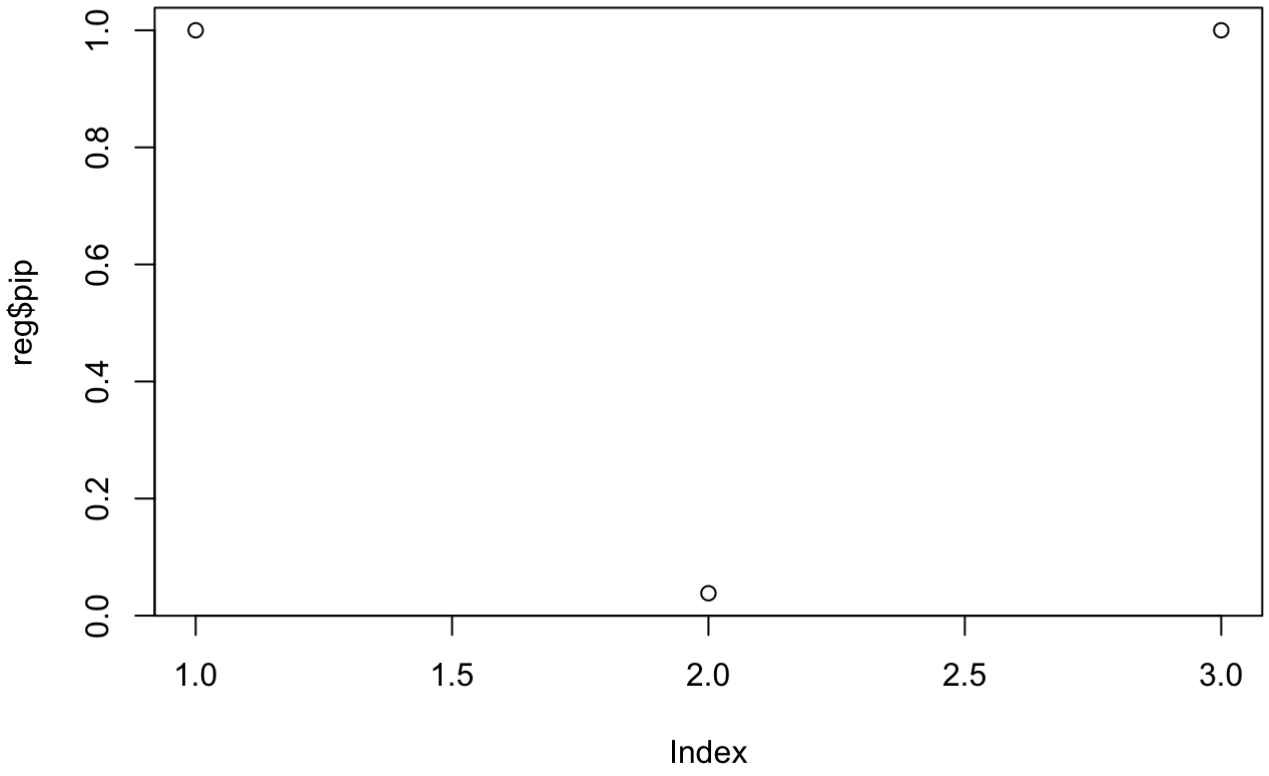
The full algorithm returns the PIP for each SNP:

```
snpwisereg = function(X,y,sigmaa, priorpi){
  normalize <- 0
  if (missing(priorpi)){
    p <- par
  }
  else {
    p <- 1/priorpi
  }
  df <- data.frame(X,y)
  snps <- dim(X)[2]
  snpnames <- c()
  pips <- rep(0,snps)
  for (i in c(1:snps)){
    colnames(df)[i] <- paste0("x",i)
    snpnames <- c(snpnames, paste0("x",i))
  }
  genweights <- snpwise_weights(X,y,sigmaa,priorpi)
  for (i in c(1:snps)){
    reg <- stepwise(df, y = colnames(df)[snps+1], include = colnames(df)[i], selection =
"forward")
    selected <- reg$variate
    selected <- unique(selected)
    if (selected[1]=="intercept"){
      selected <- selected[-1]
    }
    incl <- ""
    numincl <- 0
    for (j in c(1:snps)) {
      if (paste0("x",j) %in% selected) {
        numincl <- numincl + 1
        if (incl == "") {
          incl <- paste0("x",j)
        }
      else {
        toincl <- paste0("x",j)
        incl <- paste(incl, toincl, sep=",")
    }
  }
}
 ourrow <- filter(genweights, models == incl)
 normalize <- normalize + ourrow$pmps
 for (k in c(1:snps)){
  if (colnames(df)[k] %in% selected){
    pips[k] <- pips[k] + ourrow$pmps}
 }
}

  pips <- pips/normalize

  toreturn <- data.frame(snpnames, pips)
  names(toreturn)[1] <- "snp"
  names(toreturn)[2] <- "pip"
```

```
  return(toreturn)
}
```

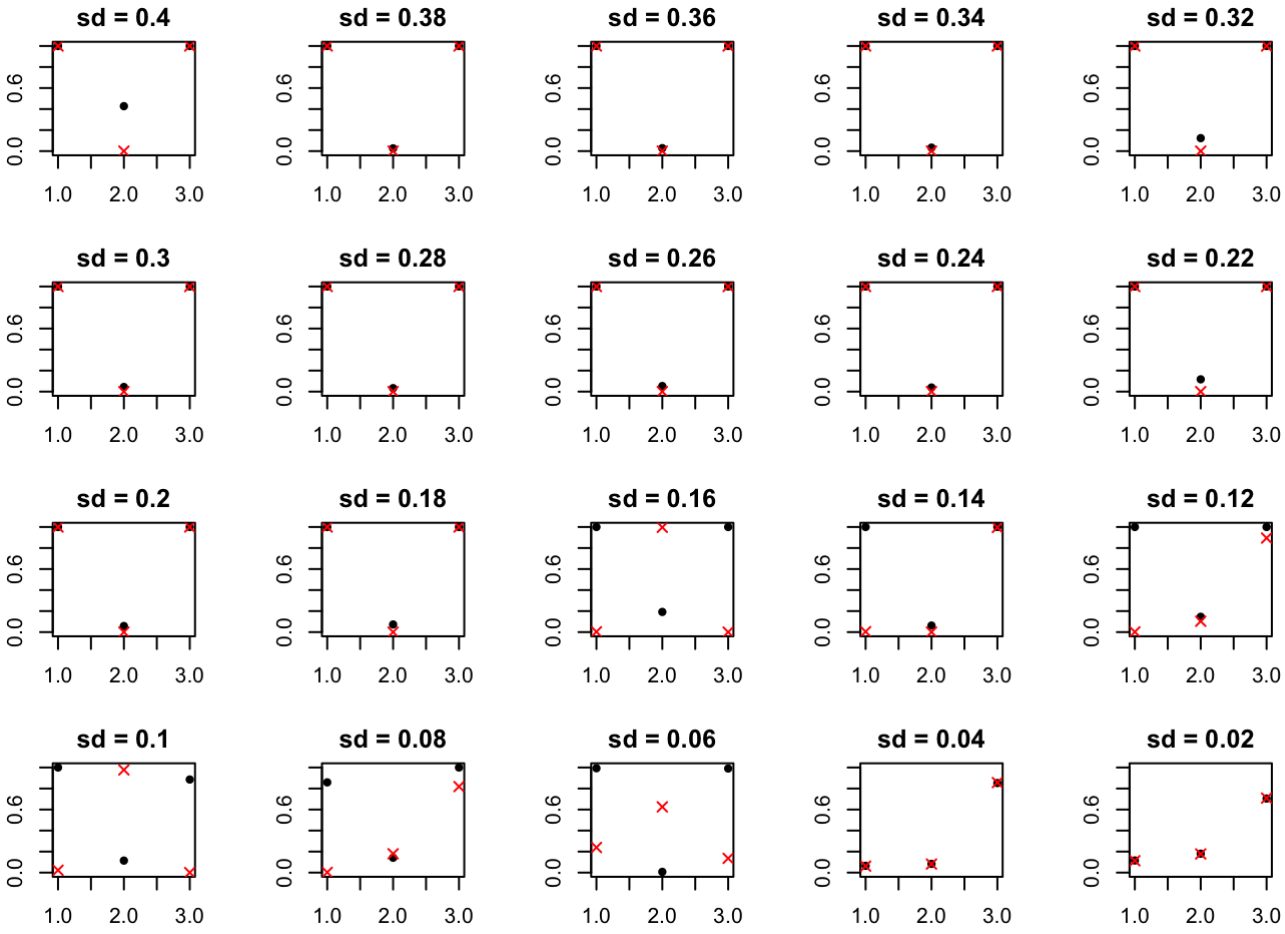
We test the algorithm on an example of the three-body problem, with X1 and X3 the true causal SNPs, and a prior of $\pi = 10^{-3}$:

```
set.seed(100)
x1 <- rnorm(100)
x3 <- rnorm(100)
x2 <- x1 + x3 + rnorm(100, sd = 0.5)
y <- x1 + x3 + rnorm(100)
X <- cbind(x1, x2, x3)

reg <- snpwisereg(X,y,0.5)
plot(reg$pip)
```




We then test the algorithm on a series of 20 different data sets of various correlations and compare the resulting PIPs with those from SuSiE, with black points being from the snpwise method and red crosses being from SuSiE:

```
set.seed(100)
sds <- sort(seq(0.02, 0.4, by=0.02), decreasing = TRUE)
par(mfrow = c(4,5))
for (i in c(1:20)){
  x2 <- rnorm(1000)
  x1 <- x2 + rnorm(1000, sd = sds[i])
  x3 <- x2 + rnorm(1000, sd = sds[i])
  X <- cbind(x1, x2, x3)
  y <- x1 + x3 + rnorm(1000)

  reg <- snpwisereg(X,y,0.5)
  reg2 <- susie(X, y, L= 2)
  title <- paste0("sd = ", sds[i])
  title2 <- paste("susie", title, sep = " ")
  par(mar = c(3,2,2,3))
  plot(reg$pip, xlab = "Predictor", ylab = "PIP", main = title, pch = 20, ylim = c(0,1))
  par(new=TRUE)
  plot(reg2$pip, ylim = c(0,1), axes = FALSE, pch = 4, col = 2)
}
```




We can see that, in this example at least, the snpwise method's PIPs remain highly accurate at lower correlations than SuSiE, though at the lowest both do begin to deviate.