SNP-wise General Weights2.0

Stuart Brabbs

We first redefine the log10BF function (note that we rename log10BF to emphasize that it is indeed in base 10).

```
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)*(log10(t(y- X %*% B) %*% y) - log10(t(y) %*% y - n*mean(y)^2)))
  }
}
```

We then redefine the snpwise_weights function to do several things: add a π parameter that allows us to make all priors smaller or larger based on previous knowledge (e.g., $\pi = 10^{-3}$ if we generally expect 1 in 1000 SNPs to be significant), redefine the "post" variable as "log10post", and include a normalizing constant in the output weights:

```
snpwise_weights = function(X,y,priorpi,sigmaa) {
  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] <- "weights"
  names(toreturn)[4] <- "log10norm"
  names(toreturn)[5] <- "log10bayesfactor"

  return(toreturn)
}
```

We first test this function without inputting any prior π, and thus defaulting to 1/p. We can also do a sanity to confirm that the weights sum to 1:

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

test1 <- snpwise_weights(X, y, sigmaa = 0.5)
test1
```

```
##     models log10postscores      weights log10norm log10bayesfactor
## 1     NULL      -0.5282738 6.049175e-26  24.69003         0.000000
## 2       X1      12.1398224 2.817036e-13  24.69003        12.969126
## 3       X2      20.7256458 1.085465e-04  24.69003        21.554950
## 4    X1,X2      22.0699362 2.398315e-03  24.69003        23.200270
## 5       X3       5.5119809 6.636679e-20  24.69003         6.341285
## 6    X1,X3      24.6366134 8.842668e-01  24.69003        25.766947
## 7    X2,X3      19.8770641 1.538275e-05  24.69003        21.007398
## 8 X1,X2,X3      23.7439183 1.132109e-01  24.69003        25.527465
```

```
sum(test1$weights)
```

```
## [1] 1
```

We can next test the function with the same data as above, but setting $\pi = 10^{-3}$:

```
test2 <- snpwise_weights(X, y, priorpi = 10^(-3), sigmaa = 0.5)
test2
```

```
##     models log10postscores      weights log10norm log10bayesfactor
## 1     NULL      -0.4345118 5.912632e-21  19.79371         0.000000
## 2       X1       9.9682571 1.494685e-10  19.79371        12.969126
## 3       X2      18.5540806 5.759346e-02  19.79371        21.554950
## 4    X1,X2      17.1998355 2.547582e-03  19.79371        23.200270
## 5       X3       3.3404156 3.521342e-17  19.79371         6.341285
## 6    X1,X3      19.7665126 9.393020e-01  19.79371        25.766947
## 7    X2,X3      15.0069633 1.634015e-05  19.79371        21.007398
## 8 X1,X2,X3      16.5265956 5.406152e-04  19.79371        25.527465
```

```
sum(test2$weights)
```

```
## [1] 1
```

With this parameter set, we can see that the models with multiple variables are penalized much more than before. Where in the first example the largest weighted model was X1,X3, followed by X2, they are now reversed.