SNP-wise 2.0

Stuart Brabbs

We continue to experiment with the previous "SNP-wise" regression algorithm, first looking more closely at the marginal likelihoods (via Bayes factors) and then combining with the prior. We first redefine the logBF function to remove σ_d^2, and removing the dominance effect terms from the matrices, and precenter variables so that we can remove the intercepts as well, thus removing σ_μ^2 as well. We then take off the leading 0 from the variance matrix, as it corresponded to the intercepts:

```
logBF = function(g,y,sigmaa) {
  p=dim(g)[2]
  n=dim(g)[1]
  if (is.null(dim(g)[2])){
  g = scale(g, scale = FALSE)
  y = scale(y, scale = FALSE)
  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)))
  }
}
```

We first check that the new logBF function gives believable results in a case with 4 uncorrelated causal SNPs, 100 total SNPs, and a continuous outcome:

```
set.seed(1)
geno <- sample(0:2, 100, replace=TRUE)
for (i in c(2:100)) {
  g <- sample(0:2, 100, replace=TRUE)
  geno <- cbind(geno, g)
}

y <- geno[,1] + geno[,20] + geno[,50] + geno[,95] + rnorm(100)

logBF(geno[,1], y, 0.5)
```

```
##          [,1]
## [1,] 4.65386
```

```
logBF(geno[,2], y, 0.5)
```

```
##           [,1]
## [1,] 0.5167932
```

```
logBF(cbind(geno[,1], geno[,2]), y, 0.5)
```

```
##          [,1]
## [1,] 4.366337
```

```
logBF(cbind(geno[,1], geno[,95]), y, 0.5)
```

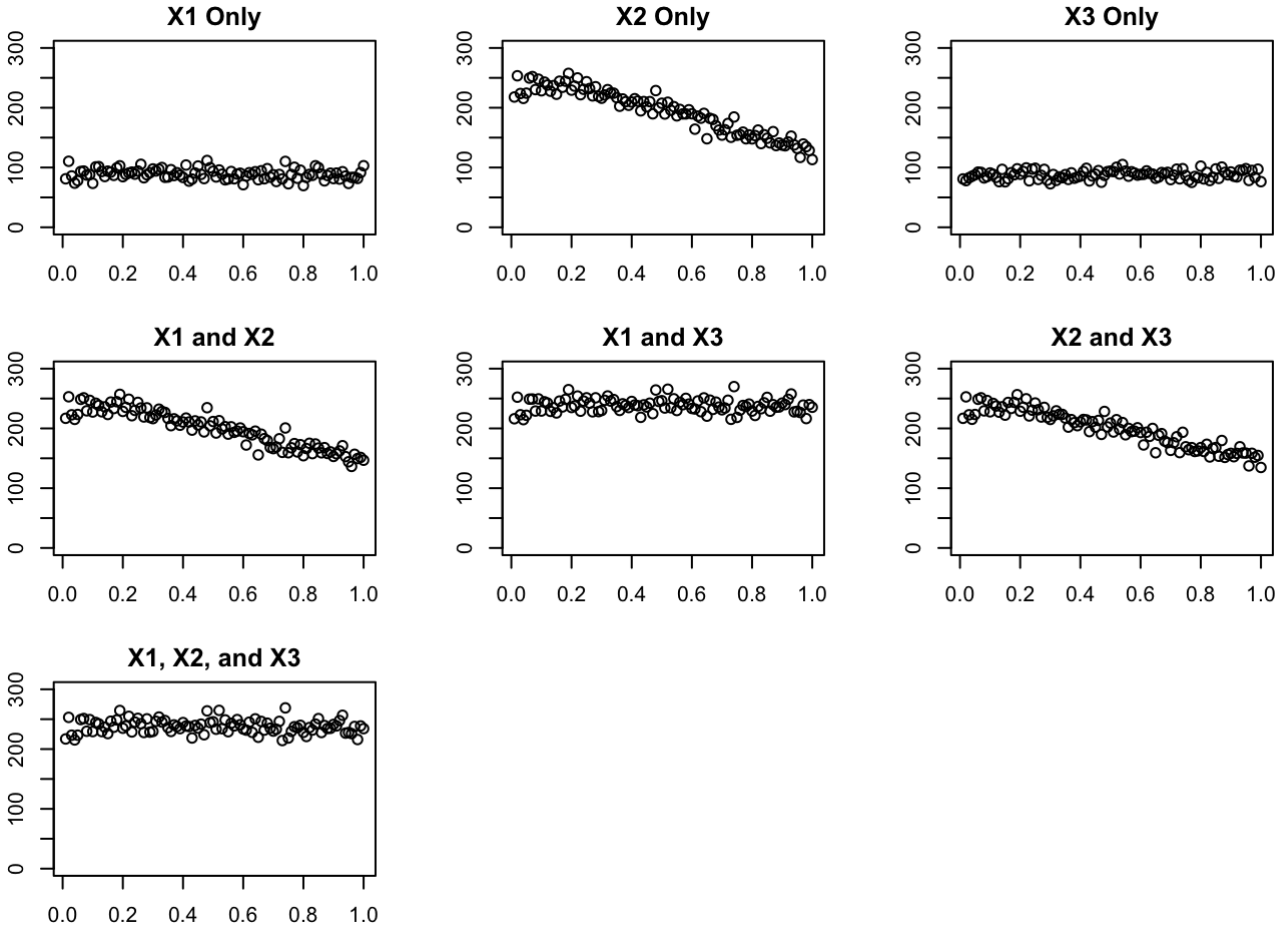
```
##          [,1]
## [1,] 9.952212
```

```
logBF(cbind(geno[,2], geno[,21]), y, 0.5)
```

```
##           [,1]
## [1,] 0.1796641
```

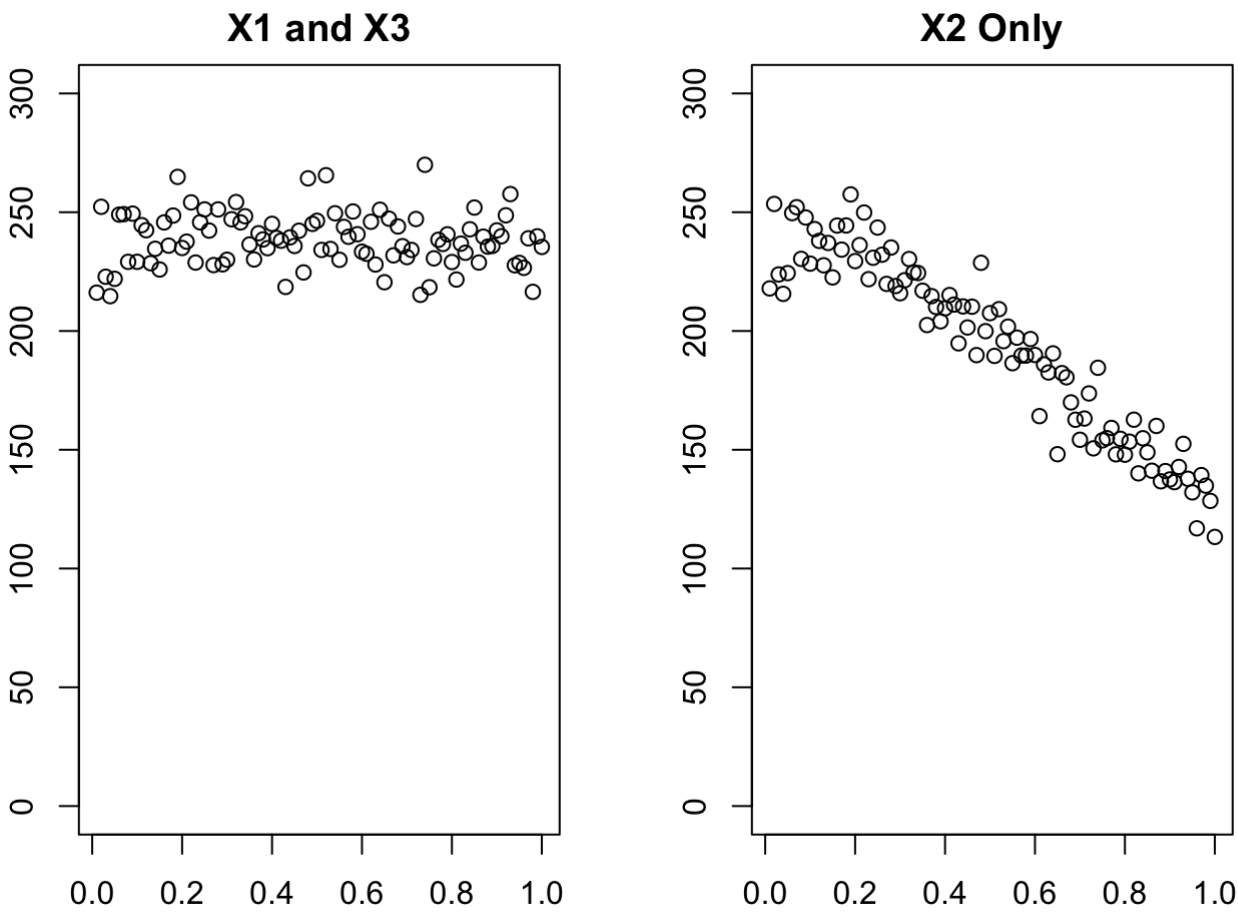
We now want to examine the Bayes factors produced by each possible (non-null) model in the "three body problem." We will cycle through 100 decreasing standard deviations for the noise in the causation from x1 and x3 to y, and plot the log Bayes factors over 100 iterations for each model:

```
set.seed(100)
mx1 <- c()
mx2 <- c()
mx3 <- c()
mx12 <- c()
mx13 <- c()
mx23 <- c()
mx123 <- c()

sds <- sort(seq(0.01, 1, by=0.01), decreasing=TRUE)

for (i in c(1:100)){
  x1 <- rnorm(1000)
  x3 <- rnorm(1000)
  x2 <- x1 + x3 + rnorm(1000, sd = sds[i])
  y <- x1 + x3 + rnorm(1000)

  mx1 <- c(mx1, logBF(x1, y, 0.5))
  mx2 <- c(mx2, logBF(x2, y, 0.5))
  mx3 <- c(mx3, logBF(x3, y, 0.5))
  mx12 <- c(mx12, logBF(cbind(x1, x2), y, 0.5))
  mx13 <- c(mx13, logBF(cbind(x1, x3), y, 0.5))
  mx23 <- c(mx23, logBF(cbind(x2, x3), y, 0.5))
  mx123 <- c(mx123, logBF(cbind(x1, x2, x3), y, 0.5))
}
par(mfrow = c(3,3))
par(mar = c(3,2,2,3))
plot(y = mx1, x = sds, ylim = c(0,300), main = "X1 Only", ylab = "log BF", xlab = "SD of
Corr. Noise")
plot(y = mx2, x = sds, ylim = c(0,300), main = "X2 Only", ylab = "log BF", xlab = "SD of
Corr. Noise")
plot(y = mx3, x = sds, ylim = c(0,300), main = "X3 Only", ylab = "log BF", xlab = "SD of
Corr. Noise")
plot(y = mx12, x = sds, ylim = c(0,300), main = "X1 and X2", ylab = "log BF", xlab = "SD
of Corr. Noise")
plot(y = mx13, x = sds, ylim = c(0,300), main = "X1 and X3", ylab = "log BF", xlab = "SD
of Corr. Noise")
plot(y = mx23, x = sds, ylim = c(0,300), main = "X2 and X3", ylab = "log BF", xlab = "SD
of Corr. Noise")
plot(y = mx123, x = sds, ylim = c(0,300), main = "X1, X2, and X3", ylab = "log BF", xlab
= "SD of Corr.")
```



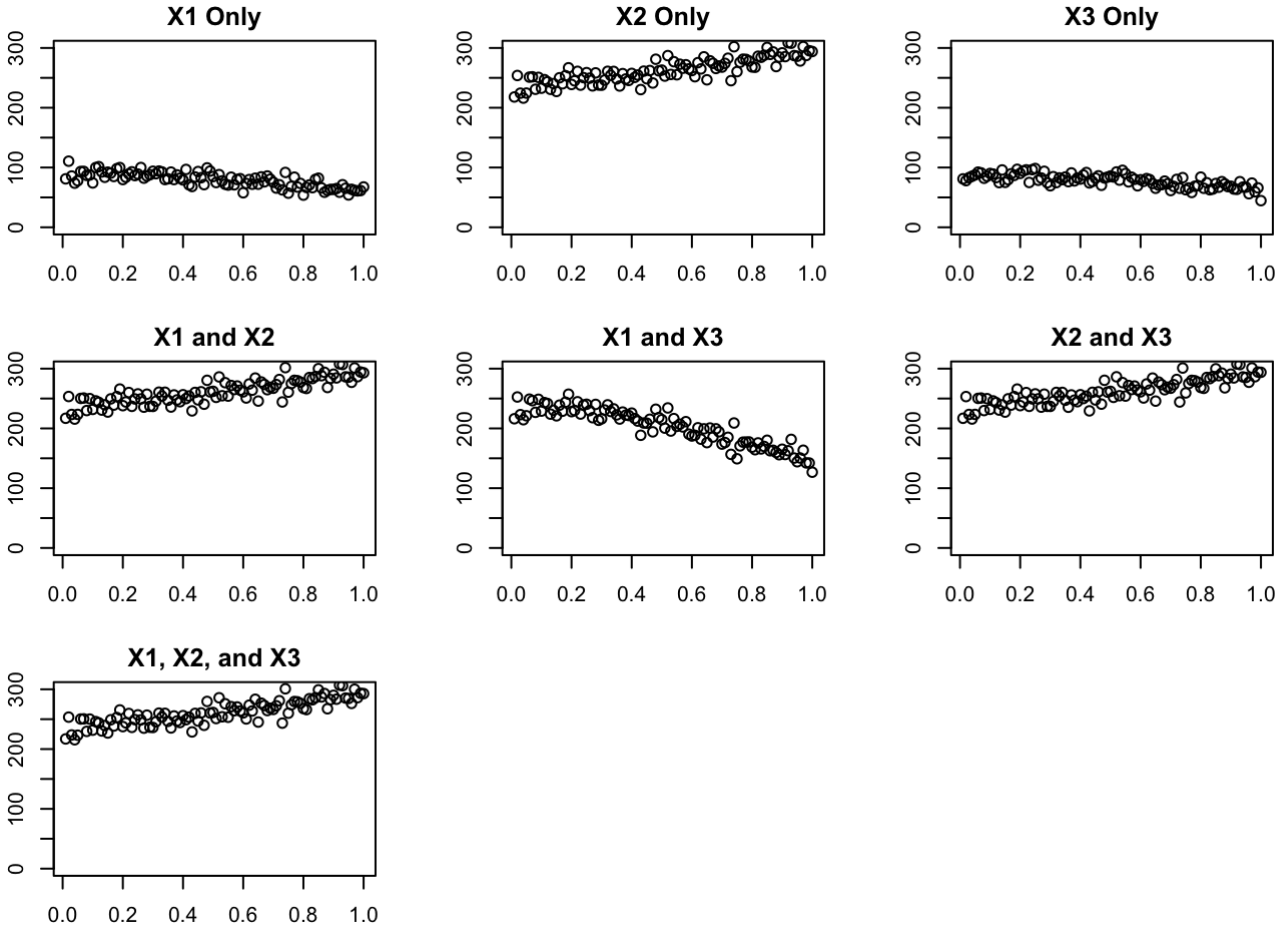
We also plot larger the two main models of interest - X1 and X3 vs. X2 alone:

```
par(mfrow = c(1,2))
par(mar = c(3,2,2,3))
plot(y = mx13, x = sds, ylim = c(0,300), main = "X1 and X3", ylab = "log BF", xlab = "SD
of Corr. Noise")
plot(y = mx2, x = sds, ylim = c(0,300), main = "X2 Only", ylab = "log BF", xlab = "SD of
Corr. Noise")
```

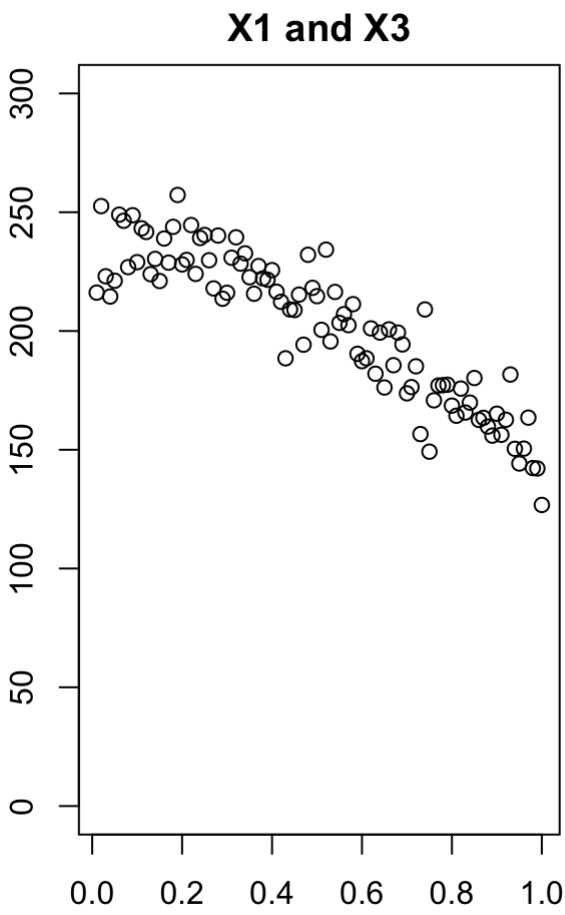


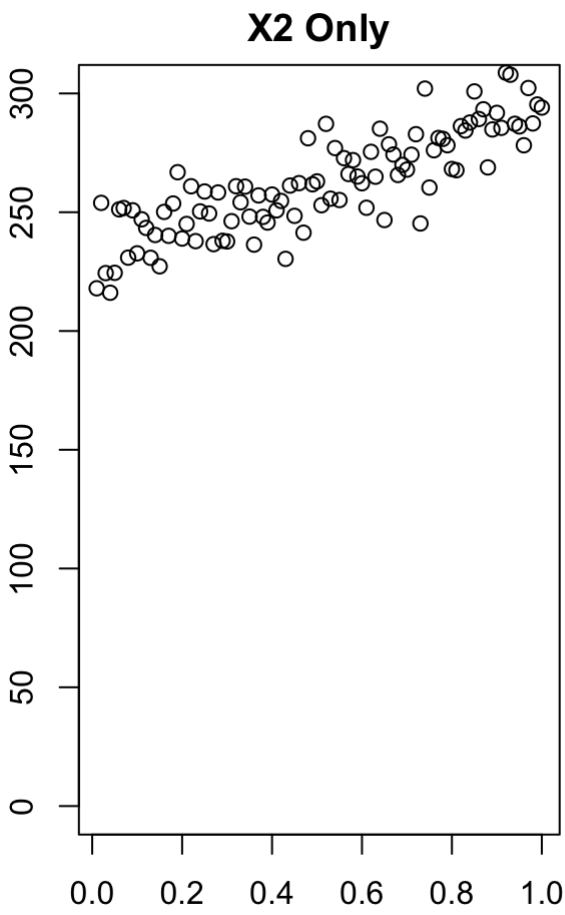
We now invert the truth, making X2 the true model, and reproduce the above plots:

```
mx1 <- c()
mx2 <- c()
mx3 <- c()
mx12 <- c()
mx13 <- c()
mx23 <- c()
mx123 <- c()

sds <- sort(seq(0.01, 1, by=0.01), decreasing=TRUE)

set.seed(100)

for (i in c(1:100)){
  x1 <- rnorm(1000)
  x3 <- rnorm(1000)
  x2 <- x1 + x3 + rnorm(1000, sd = sds[i])
  y <- x2 + rnorm(1000)

  mx1 <- c(mx1, logBF(x1, y, 0.5))
  mx2 <- c(mx2, logBF(x2, y, 0.5))
  mx3 <- c(mx3, logBF(x3, y, 0.5))
  mx12 <- c(mx12, logBF(cbind(x1, x2), y, 0.5))
  mx13 <- c(mx13, logBF(cbind(x1, x3), y, 0.5))
  mx23 <- c(mx23, logBF(cbind(x2, x3), y, 0.5))
  mx123 <- c(mx123, logBF(cbind(x1, x2, x3), y, 0.5))
}
par(mfrow = c(3,3))
par(mar = c(3,2,2,3))
plot(y = mx1, x = sds, ylim = c(0,300), main = "X1 Only", ylab = "log BF", xlab = "SD of
Corr. Noise")
plot(y = mx2, x = sds, ylim = c(0,300), main = "X2 Only", ylab = "log BF", xlab = "SD of
Corr. Noise")
plot(y = mx3, x = sds, ylim = c(0,300), main = "X3 Only", ylab = "log BF", xlab = "SD of
Corr. Noise")
plot(y = mx12, x = sds, ylim = c(0,300), main = "X1 and X2", ylab = "log BF", xlab = "SD
of Corr. Noise")
plot(y = mx13, x = sds, ylim = c(0,300), main = "X1 and X3", ylab = "log BF", xlab = "SD
of Corr. Noise")
plot(y = mx23, x = sds, ylim = c(0,300), main = "X2 and X3", ylab = "log BF", xlab = "SD
of Corr. Noise")
plot(y = mx123, x = sds, ylim = c(0,300), main = "X1, X2, and X3", ylab = "log BF", xlab
= "SD of Corr.")

par(mfrow = c(1,2))
```




```
par(mar = c(3,2,2,3))
plot(y = mx13, x = sds, ylim = c(0,300), main = "X1 and X3", ylab = "log BF", xlab = "SD
of Corr. Noise")
plot(y = mx2, x = sds, ylim = c(0,300), main = "X2 Only", ylab = "log BF", xlab = "SD of
Corr. Noise")
```





When attempting to adapt the SNPwise algorithm to the new logBF function, continually get the error nonconformable invnu + t(X) %*% X. Since the logBF works fine on its own with each model configuration, suspect may have to do with null model. Will work out in coming days.

```
SNPwise = function(X, y, sigmaa){
  df <- data.frame(X, y)
  snps <- dim(X)[2]
  bfs <- c()
  snpprobs <- rep(0, snps)
  for (i in c(1:snps)){
    colnames(df)[i] <- paste0("x", i)
  }
  for (i in c(1:snps)){
    reg <- stepwise(df, y = colnames(df)[snps + 1], include = colnames(df)[i], selection
= "forward")
    selected <- reg$variate[-1]
    selmatrix <- data.matrix(df[,selected])
    newbf <- logBF(selmatrix, y, sigmaa)
    for (i in c(1:snps)){
      if (colnames(df)[i] %in% selected) {
        snpprobs[i] <- snpprobs[i] + newbf
      }
    }
    bfs <- c(bfs, newbf)
  }
  normalizer <- sum(bfs)
  return(snpprobs)
}
```