Diabetes Example

First lets load the data and coerce them to be dataframes.

```
set.seed(8675309)
source("yX.diabetes.train.txt")
diabetes.train = as.data.frame(diabetes.train)

source("yX.diabetes.test.txt")
diabetes.test = as.data.frame(diabetes.test)
colnames(diabetes.test)[1] = "y"
```

BMA using BAS

Next we will load the library BAS (download first if it is not in your list of packages)

```
library(BAS)
```

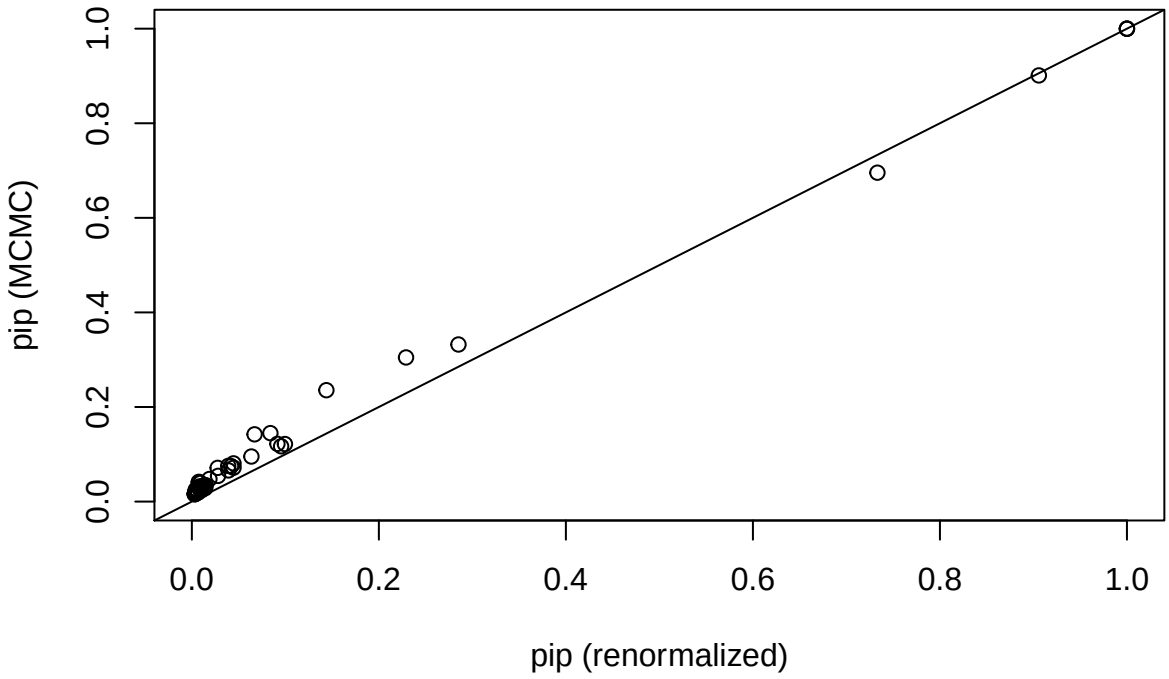
The following will run a MCMC sampler that combines a Random Walk Metropolis algorithm with a random swap step. The algorithm stops when the number of iterations exceeds `MCMC.iterations` or `n.models` have been visited. This will save every 10th model. The `initprobs` argument uses the p-values from OLS to compute an approximate Bayes Factor. These are used internally to sort the variables which provides improved memory usage.

```
diabetes.bas = bas.lm(y ~ ., data=diabetes.train,
                      method="MCMC",
                      n.models = 150000,
                      thin = 10, initprobs="eplogp")
```

```
## Warning in bas.lm(y ~ ., data = diabetes.train, method = "MCMC", n.models =
## 150000, : We recommend using the implementation using the Jeffreys-Zellner-
## Siow prior (prior='JZS') which uses numerical integration rahter than the
## Laplace approximation
```

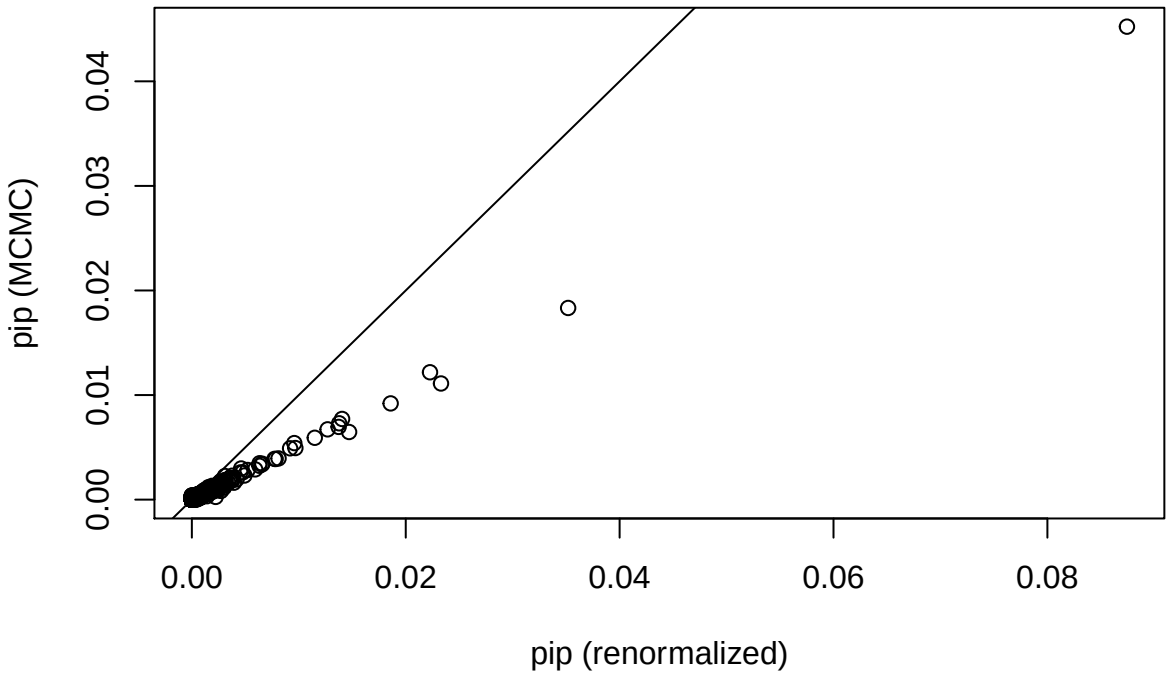
Let's check if the algorithms is even close to converging as there are 2^{64} models! The function `diagnostics` compares estimates of posterior inclusion probabilities estimated from Monte Carlo frequencies to estimates based on normalizing inclusion marginal likelihoods times prior probabilities of sampled models. These should be equal if the chain has converged.

```
diagnostics(diabetes.bas, type="pip")
```



Close, but could run longer! We can also compare the model probabilities

```
diagnostics(diabetes.bas, type="model")
```



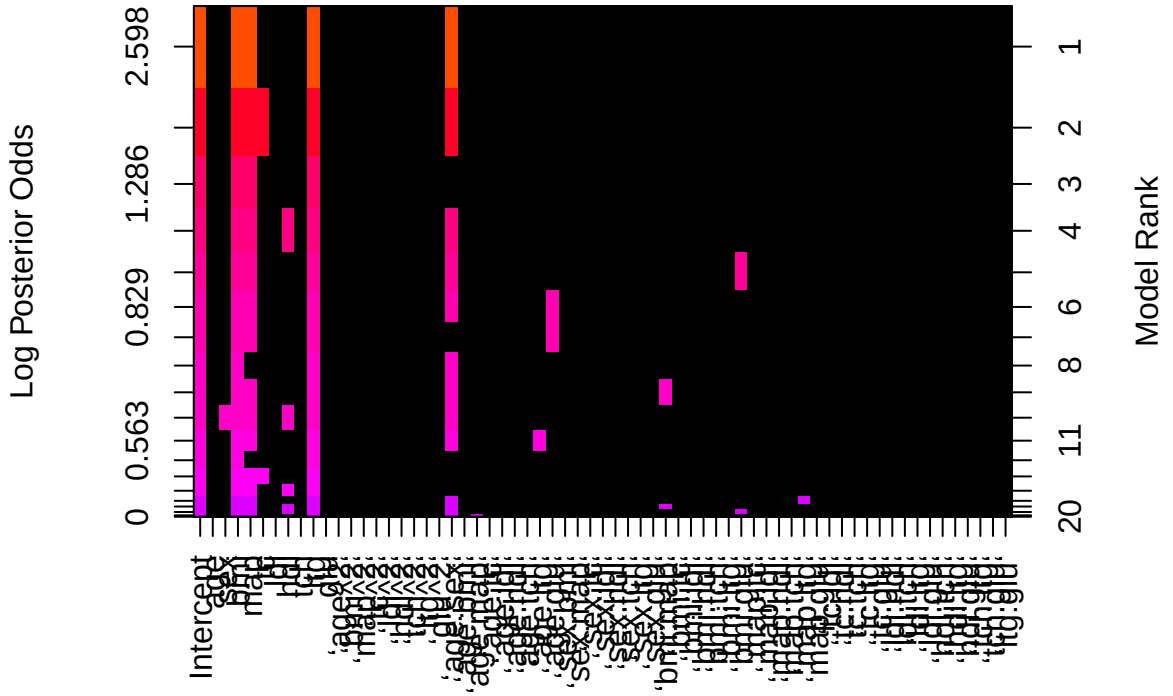
Clearly, not long enough. One problem with MCMC and model selection is that the "best" model may be visited very few times.

```
summary(diabetes.bas$freq)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    1.00    2.00    4.00   10.04    9.00 6784.00
```

What variables are included?

```
image(diabetes.bas)
```



Despite this, let's use BMA for prediction and compute the MSE between the predicted and actual responses for the test data using BMA:

```
pred.bas = predict(diabetes.bas, newdata=diabetes.test, estimator="BMA")
cv.summary.bas(pred.bas$fit, diabetes.test$y)
```

```
## [1] 0.6745578
```

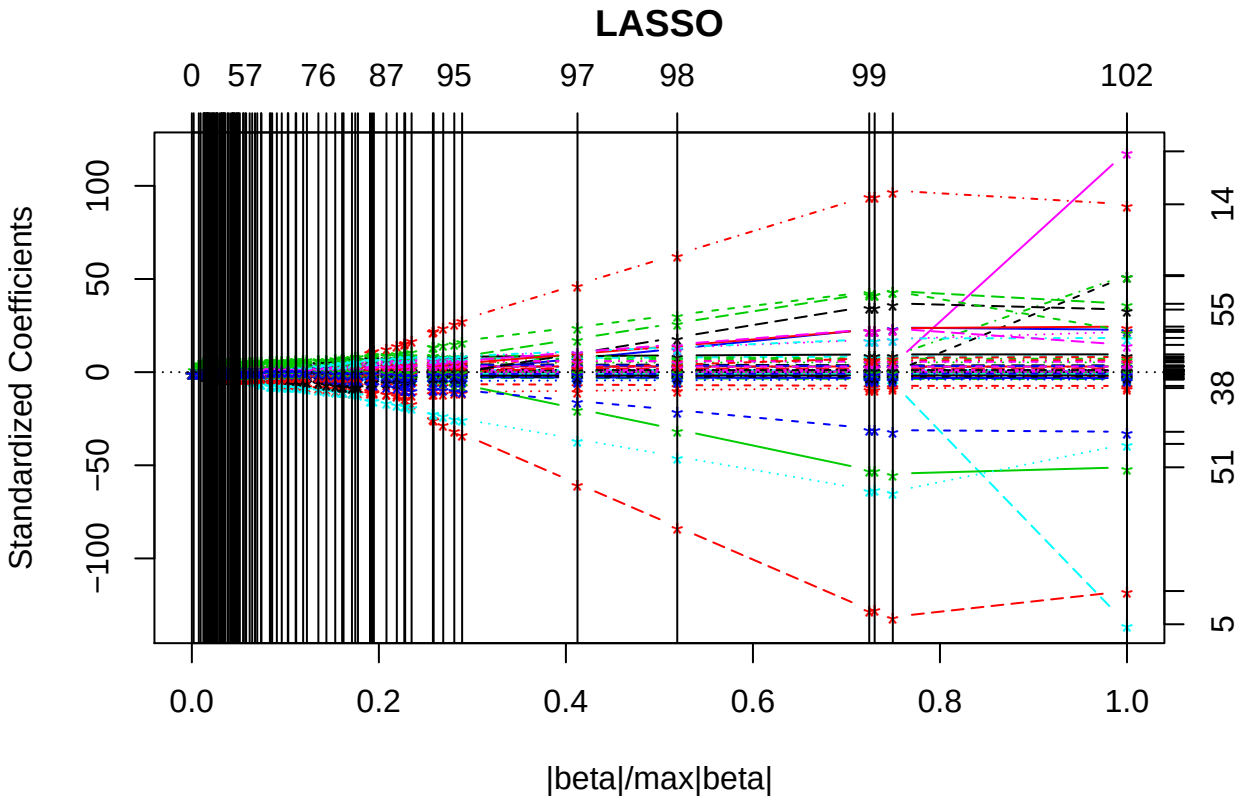
Similary for lasso, we can compute the out of sample MSE:

```
library(lars)
```

```
## Loaded lars 1.2
```

```
diabetes.lasso = lars(as.matrix(diabetes.train[, -1]), diabetes.train[,1], type="lasso")

plot(diabetes.lasso)
```




```
best = which.min(diabetes.lasso$Cp)
out.lasso = predict(diabetes.lasso, as.matrix(diabetes.test[,-1]))
cv.summary.bas(out.lasso$fit[, best], diabetes.test$y)
```

```
## [1] 0.6890253
```

Summary

I routinely use much larger numbers of MCMC iterations than I see in papers if looking at BMA or model selection (i.e. millions rather than say 10,000)

Explore different numbers of models - does it make a difference for estimating marginal inclusion probabilities or posterior inclusion probabilities? What about predictions under BMA?