My Final College Paper

A Thesis
Presented to
Department of Statistical Science
Duke University

Huijia Yu

May 2018

Approved for the
Bachelor of Science in Statistical Science

____

Merlise Clyde

____

Committeemember O. Name

____

Committeemember T. Name

____

Mine Cetinkaya-Rundel, DUS

Acknowledgements

I want to thank a few people.

Preface

This is an example of a thesis setup to use the reed thesis document class (for LaTeX) and the R bookdown package, in general.

Table of Contents

List of Tables

List of Figures

Abstract

The preface pretty much says it all.

Second paragraph of abstract starts here.

Dedication

You can have a dedication here if you wish.

Introduction

Chapter 1

Abstract

Chapter 2

Introduction

Chapter 3

Literature Review

P-values have been the reason behind lack of reproducibility in scientific discoveries, causing concern and leading to proposals of new ways to define significance. Benjamin et. al have shown that the Bayes factor equivalents for commonly used p-values only correspond to “weak” evidence in the Bayes factor characterization. (Benjamin et al., 2017) They suggest reducing the p-value threshold in studies with less power, but acknowledge that hypothesis testing with thresholding is still an issue. Another approach is suggested by Selke et al , who propose two calibrations of the p-value: as the lower bound of the Bayes factor under any alternative hypothesis, and as a posterior probability of the type 1 error in a Bayesian framework (Sellke, Bayarri, & Berger, 2001).

This problem has become a major issue in replicated studies, an effect known as the “winner’s curse” (Zöllner & Pritchard, 2007) or the Beavis effect (S. Xu, 2003). Zollner and Pritchard first define this in the context of genome-wide association scans (GWAS), which use stringent thresholds for significance, resulting in inflated effect sizes after selection, especially since these are calculated with the same data. Thus, replication studies underestimate the sample size necessary and do not have enough power to detect an effect. Zollner and Pritchard suggest a conditional-likelihood based method to address this issue. They propose a computational algorithm to maximize over the the likelihood of the parameters conditional on the significance association at level α, which results in less biased coefficient estimates (albeit with larger variance) and sample size estimates centered at the true value.(Zöllner & Pritchard, 2007)

Zhong and Prentice also propose a similar method, but use a different parametrization and an asymptotic approximation instead of a computational one to find the estimators, which is more computationally efficient (Zhong & Prentice, 2008). Ghosh et al. also define an approximate conditional likelihood, and propose two more estimators (other than the MLE): the mean of the (normalized) conditional likelihood, which can be interpreted as a posterior mean of the parameters under a flat prior, and a “compromise” estimator which is the average of the mean and MLE (Ghosh, Zou, & Wright, 2008). The combination estimator proves to have the most stable MSE accross the range of true values for the parameters. Their approach only requires summary statistics, so they further apply it to published datasets. The results are similar for the three conditional likelihood approaches, which have been applied to other studies such as Palmer and Peter .

Another method proposed to create bias-reduced estimates uses bootstrap resampling to correct for both the thresholding effect and the ranking effect, which is not addressed in the conditional likelihood methods because of the difficulty of specifying joint likelihoods for correlated variables (Sun et al., 2011). By using a sample-split approach, the detection and estimation datasets can be virtually independent. This is repeated multiple times in order to reduce variance in the results. The main drawback of this approach is its computational intensity.

Several authors have also proposed shrinkage-based methods in the effect detection step. Bacanu and Kendler use a soft threshold method to scale statistics such that their sum of squares do not overestimate the true mean and then find "suggestive" signals in a GWAS context by setting a threshold. This method does not address the winner's curse directly, but provides a subset of the genome which can be futher analyzed or used in future studies (Bacanu & Kendler, 2013). Bigdelli et al. propose shrinking coefficient estimates by drawing a comparison between "winner's curse adjustments" for effect sizes and multiple testing approaches for p-values, since both are used on the tail of their respective distributions. Their method transforms False Discovery Rate (FDR) adjusted p-values into the corresponding Z-score and uses that as the estimator.(Bigdeli et al., 2016) Both Bigdelli and Bacanu assume the data is normally distributed. Storey and Tibshirani , on the other hand, propose to adjust the value used for significance testing rather than the coefficients, choosing the FDR value as an alternative to the p-value.

Multiple Bayesian methods have also been proposed: Xu et al use a Bayesian approach to a logistic regression, selecting a spike and slab prior for the mean and an inverse gamma prior for the variance. (L. Xu, Craiu, & Sun, 2011) A beta prior for the proportion of each component in the prior, and the hyperparameters were estimated empirically. They also propose a Bayesian Model Average approach, which they recommend for instances with little prior information. Their results show that the Bayesian models had smaller variance than conditional likelihood methods, but still do not address the "ranking effect" from Sun (Sun et al., 2011), or implement a fully Bayesian approach because of the dependence on the threshold α.

Ferguson et al propose an Empirical Bayes approach, which estimate the prior density distribution with the data (Ferguson, Cho, Yang, & Zhao, 2013). This is a nonparametric estimate, but still depends on other specifications such as the number of bins, type of splines, etc. Using the empirical prior, the posterior is then calculated, from which the estimate and pseudo-Bayesian credible intervals are derived by considering the 5% and 95% points. This method resulted in better estimates in the higher density regions, but performed worse than conditional likelihood methods on the tails. Thus, the authors propose a combined method, which calculates both the empirical Bayes and the conditional likelihood confidence intervals, and picks the shortest one. One possible problem with this approach is the use of non-HPD intervals, which could change the tail behavior.

Jiang and Yu apply the Bayesian framework to power calculations specifically, defining "Bayesian power" as the marginal probability of finding significance in a replicated study given the original and the data. They also use a spike and slab prior, but estimate the hyperparameters empirically. The resulting power estimators are improved, but lead to downwards bias in the effect size.(Jiang & Yu, 2016)

Chapter 4

Simulation Study

4.1 Normal Example

4.1.1 Data Generation

To test the hypothesis $H_0 : mu = 0$ versus $H_1 : mu \neq 0$, a fixed proportion (set at 0.5) of null vs. alternative hypotheses are generated. For each hypothesis H_i, let $mu_i = 0$ in the null scenario and $mu_i \sim N(0, 1)$ in the alternative. The data Y_i is generated from a normal distribution with mean mu_i and known variance 1, with sample size 100. If Y_i is not significant at $alpha = .05$, it is sampled again from the same distribution until the sufficient statistic is significant. This is done in order to properly compare the Bayesian approach with the frequentist one, which is conditional on the data being significant.

4.1.2 Conditional Likelihood

```r
cond.posterior<- function(Y, n.samp){
  cond.likelihood<- function(Y, mu){
    ybar = mean(Y)
    N = length(Y)
    #(abs(ybar-mu)/(sqrt(1/N))>1.96)*
    dnorm(ybar, mu, sqrt(1/N))/
      (pnorm((-1.96*sqrt(1/N)), mu, sqrt(1/N))
       +1-pnorm((1.96*sqrt(1/N)), mu, sqrt(1/N)))
    #not symmetric other than mu = 0
  }
  #x<- seq(-2,2,.01)
  #plot(x, cond.likelihood(Y, x), type="l")
  #metropolis hasting with flat prior
  c=1
  mu <- 0
  MU <- NULL
  for(s in 1:n.samp){
    mu.star <- rnorm(1, mu, c)
    r = cond.likelihood(Y,mu.star)/cond.likelihood(Y,mu)
    if(runif(1)<r){
      mu <- mu.star
    }
    MU <- c(MU, mu)
  }
  #plot(MU, type="l")
  #ggplot(data = data.frame(MU))+geom_density(aes(x=MU))
  (MU)
}
```

Let B indicate that the data is significant at the level α. The conditional likelihood $L(\mu|B) = \frac{P(Y|\mu)P(B|Y,\mu)}{P(B|\mu)} = \frac{P(Y|\mu)}{\int_{\text{significant Y}} P(t|\mu)dt}$. In this case, the sufficient statistic can be used to simplify: since $\bar{Y} \sim N(\mu, \sigma^2/n)$, the conditional likelihood for the simulated data is equivalent to $L(\mu|\bar{Y}) = \frac{\phi((\bar{Y}-\mu)/\sqrt{n})}{\Phi(Z_{\alpha/2}-\mu\sqrt{n})+\Phi(Z_{1-\alpha/2}-\mu\sqrt{n})}$.

Since this likelihood is difficult to integrate analytically, the confidence intervals were estimated by treating the conditional likelihood as if it were a posterior distribution with an improper prior $\pi(\mu) = 1$, and obtaining the HPD (highest posterior density) region covering 95%. Sampling was done through a Metropolis-Hastings algorithm.

4.1.3 Posterior Distribution

```
getposterior <- function(Y,n.samp, pi = .5){
  n = length(Y)
  odds = (1-pi)/pi
  #bf.approx <- -exp(1)*p*log(p)
  bf <-  (n+1)^(-.5)*exp(n^2*mean(Y)^2/(2*(n+1)))
  alt.prob <- odds*bf/(1+odds*bf)
  #alt.prob.approx <- odds*bf.approx/(1+odds*bf.approx)

  #draws from posterior-flip a coin (ber w prob P(H given Y) and then use that to get
  draws = sapply(runif(n.samp), function(x)  {
    ifelse(x<(1-alt.prob), rnorm(1, 0, 0), rnorm(1,mean(Y)*n/(n+1), sqrt(1/(n+1))))
  })
  #ggplot(data = data.frame(draws))+geom_density(aes(x=draws))
  (list(alt.prob=alt.prob, draws=draws))

}
```

Let $\delta_a(x)$ be the Dirac delta function: $\delta_a(x) = 1$ for $x = a$ and $\delta_a(x) = 0$ otherwise. In the Bayesian case, the prior was set to a spike and slab prior, with the "spike" being a point mass at 0: $\pi(\mu|H = H_0) = \delta_0(\mu)$, and the "slab" part corresponding to a unit information prior: $\pi(\mu|H = H_1) \sim N(0, 1)$. The full prior is a mixture model $\pi(\mu|\xi) = (1 - \xi)\delta_0(\mu) + \xi\phi(\mu)$. In this case, $\xi = 0.5$ is a constant. Note that this is also the true data generating model.

The marginal posterior distribution is $P(\mu|Y) = P(H_0|Y)P(\mu|Y, H_0) + P(H_1|Y)P(\mu|Y, H_1)$. The separate posteriors for μ are: $P(\mu|Y, H_0) = \delta_0(\mu)$, $P(\mu|Y, H_1) \sim N(\frac{n}{n+1}\bar{Y}, \frac{1}{n+1})$. The posterior for the alternative hypothesis can be calculated using its bayes factor, BF and the prior odds, $\pi = \frac{(1-\xi)}{\xi}$: $P(H_1|Y) = \frac{\pi BF}{1+\pi BF}$. For this example, the prior odds are 1 (because the probability of $H_1 = \xi = 0.5$). The bayes factor $BF = \frac{L(\bar{Y}|H_1)}{L(\bar{Y}|H_0)} = \sqrt{n+1}exp(\frac{n^2}{2(n+1)}(\bar{Y})^2)$. This result comes from the fact that the marginal likelihood $L(\bar{Y}|H_1) \sim N(0, \frac{n}{n+1})$.

Putting these pieces together results in the marginal posterior for μ, which can be used to generate samples to calculate HPD credible intervals.

While true HPD intervals can be disjoint, the intervals calculated for this experiment are actually the shortest continuous segments covering .95. This corresponds to the HPD interval under the assumption that the distribution is not severely multimodal, and is what most packages in R use to estimate HPD intervals. However, since this posterior is actually a mixture model, a large enough posterior probability for H_1 could lead to a difference between the true HPD credible interval and the calculated one.

```r
HPD <-function(post, prob  = .95){

  #HPD interval- copied from BAS/coda
  obj <- as.matrix(post)
  vals <- apply(obj, 2, sort)
  if (!is.matrix(vals))
    stop("obj must have nsamp > 1")
  nsamp <- nrow(vals)
  npar <- ncol(vals)
  gap <- max(1, min(nsamp - 1, round(nsamp * prob)))
  init <- 1:(nsamp - gap)
  inds <- apply(vals[init + gap, , drop = FALSE] - vals[init,
                                              , drop = FALSE], 2, which.
  (cbind(vals[cbind(inds, 1:npar)], vals[cbind(inds +
                                         gap, 1:npar)]))
  #look into this and check about continuity of cdf, etc
}
```

4.1.4 Results

```r
all <- function(H, N, n.samp, interval,alpha){
  mu <- ifelse(H==0, 0, rnorm(1, 0, 1)) ### normally distributed mu
  Y <- rnorm(N, mu, 1)
  count=0
  while(abs(abs(mean(Y))/sqrt(1/N)-qnorm(1-alpha/2))>interval){ #if Z is not in (1.94,
    if(count>1000){ #had to add this bc it wouldn't run
      return (c(H, mu , mean(Y), NA  ,
                NA  , NA  ,
                NA , NA ,
                NA , NA , NA, NA  , NA  ,
                NA , NA, NA, NA ))
    }

    Y <- rnorm(N, mu, 1)
    count<- count+1
  }
  post <- getposterior(Y,  n.samp)
  alt.prob = post$alt.prob
  cred <- HPD(post$draws)
  cred.lower = cred[1]
  cred.upper = cred[2]
  bayes.cov =  (cred.upper>=mu&&cred.lower<=mu)

  cond <-cond.posterior(Y, n.samp)
  conf <- HPD(cond)
  conf.lower = conf[1]
  conf.upper = conf[2]

  freq.cov =  (conf.upper>=mu&&conf.lower<=mu)
  naive.cov <- mean(Y)+1.96*sqrt(1/N)>=mu&&mean(Y)-1.96*sqrt(1/N)<=mu
  expected.cov <- .95*alt.prob+(conf.upper>=0&&conf.lower<=0)*(1-alt.prob)


  bayes.est = mean(post$draws)
  bayes.median.est = median(post$draws)
  bayes.mode.est = as.numeric(names(sort(-table(post$draws)))[1])
  cond.mean.est = mean(cond)
  d <- density(cond)
  cond.mode.est = d$x[which.max(d$y)]

  (c(H, mu , mean(Y), alt.prob  ,
     cred.lower  , cred.upper  ,
     conf.lower , conf.upper ,
     bayes.cov , freq.cov , naive.cov, expected.cov,
     bayes.est, bayes.median.est,
     bayes.mode.est,cond.mean.est,cond.mode.est))

}


N=100; n.samp = 10000; n.sim=1000; pi = .5

results <-data.frame(t(apply(matrix(as.numeric(runif(n.sim)<pi)),1, function(x) a
colnames(results)<- c("H", "mu" , "Ybar", "alt.prob" ,
        "cred.lower"  , "cred.upper"  ,
        "conf.lower" , "conf.upper" ,
        "bayes.cov" , "freq.cov" ,  "naive.cov", "expected.cov",
        "bayes.est","bayes.median.est","bayes.mode.est",
        "cond.mean.est","cond.mode.est")
```

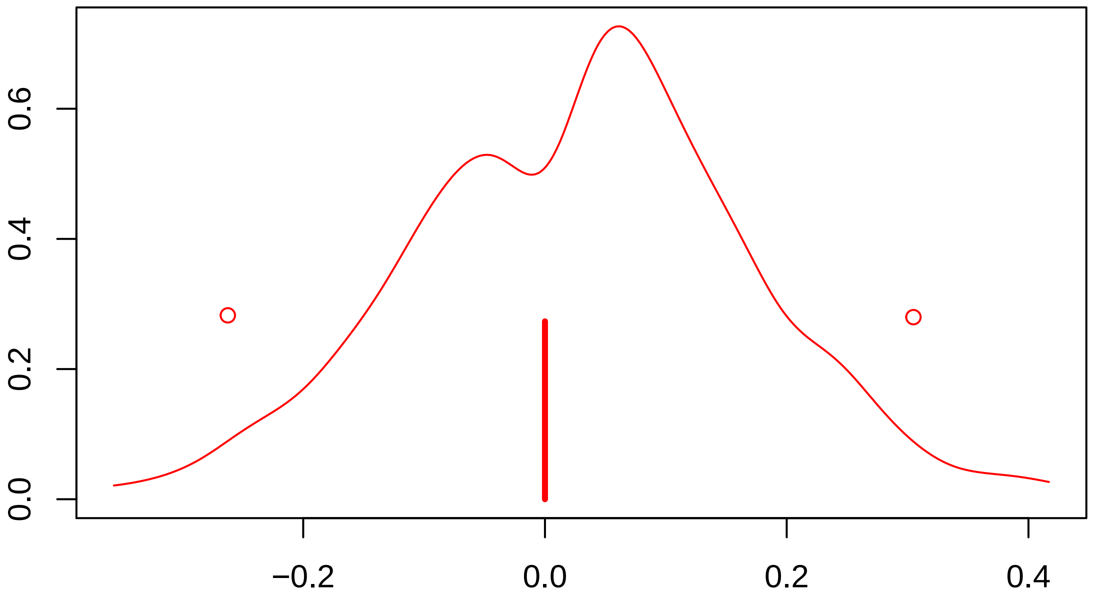
Estimators

```
##plot of estimators or mse
require(reshape2)
```

```
Loading required package: reshape2
```

```
require(knitr)
require(ggplot2)
#load("normalresults.RData")
#results <- fullresults
fullresults<-results
results<-na.omit(results)
estimators <- data.frame(bayes = abs(results$bayes.est-results$mu),
              naive = abs(results$Ybar-results$mu),
              bayes.median = abs(results$bayes.median.est-results$mu),
              bayes.mode = abs(results$bayes.mode.est-results$mu),
              cond.mean = abs(results$cond.mean.est-results$mu),
              cond.mode = abs(results$cond.mode.est-results$mu))

ggplot(data = melt(estimators), aes(x=variable, y=value)) + geom_boxplot()+labs(x
```

```
No id variables; using all as measure variables
```



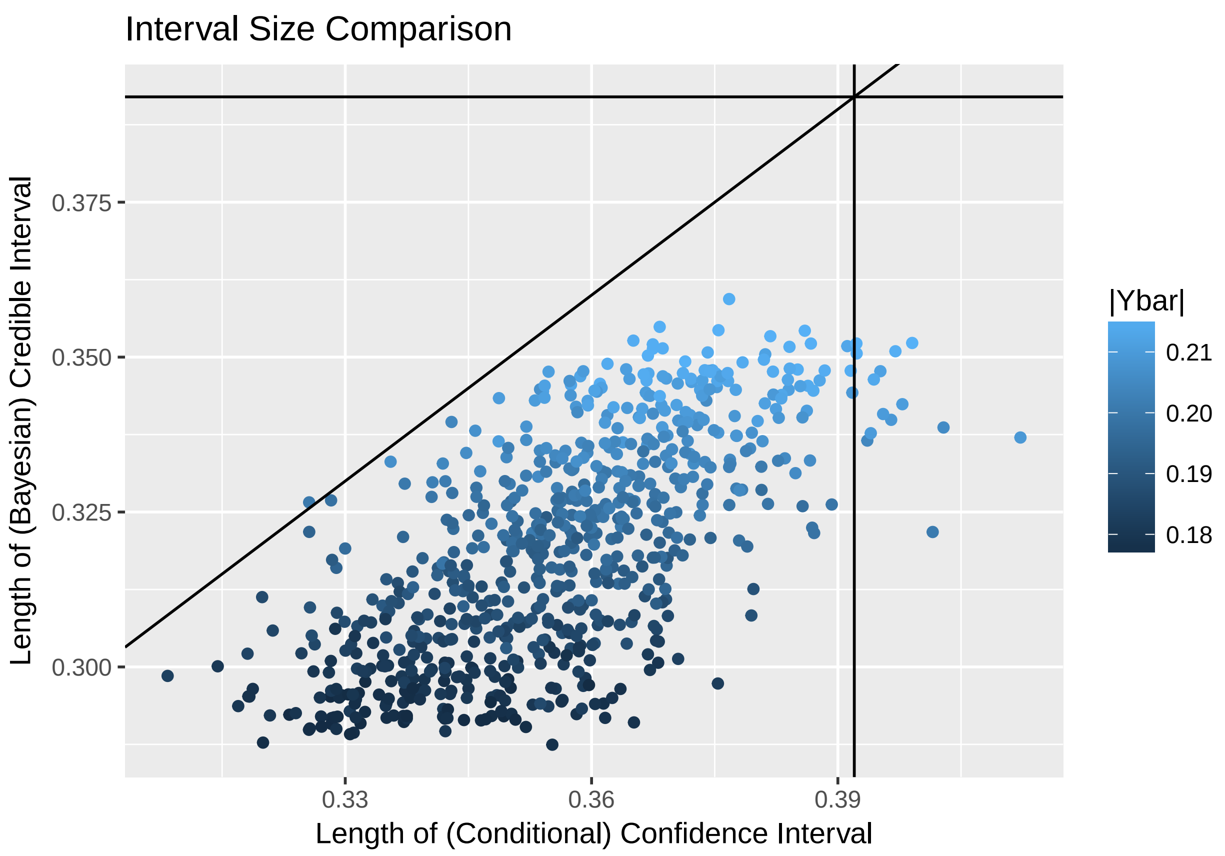
```
kable(c(bayes = sqrt(sum(estimators$bayes^2)), bayes.median= sqrt(sum(estimators$bayes.r
```

	RMSE
bayes	3.827613
bayes.median	4.365067
bayes.mode	4.365067
cond.mean	3.938311
cond.mode	3.943529
naive	4.971864

The conditional likelihood mode (i.e. MLE) has the smallest bias (absolute error) for μ out of the frequentist estimators, while the bayesian median and mode (which end up being the same) the smallest bias in the bayesian framework. The RMSE for the bayesian estimator (mean of the posterior) is the lowest, followed by the conditional mean and mode.

Credible and Confidence Intervals

```
##plot of credible interval sizes
ggplot(data = results)+ geom_point(aes(x = conf.upper-conf.lower, y = cred.upper-cred.lo
```



The lines mark the $y = x$ line, and the length of naive confidence intervals (which are constant for fixed number of samples) on the x and y axes.

The largest values for the significant statistic also correspond to the largest intervals in both cases. Note that the conditional likelihood confidence intervals are almost always larger than the credible intervals, but still mostly smaller than the naive ones.

Coverage

The marginal coverage of the (frequentist conditional likelihood) confidence interval C, $P(\mu \in C|Y) = P(\mu \in C|H_0)P(H_0|Y) + P(\mu \in C|H_1)P(H_1|Y)$ will be significantly higher than .95 for the cases in which $0 \in C$, since $P(\mu \in C|H_1) = 0.95$ by definition, and $P(\mu \in C|H_0) = I_{0\in C}$. In this experiment, the expected coverage is 0.98 for intervals with 0, and only 0.38 for those that do not contain 0.

However, conditioning on the alternative hypothesis does not lead to an empirical coverage of 95%.

We can see that both methods are still significantly better than the naive one.

```r
#table of expected vs empirical coverage

results <- na.omit(results)
b <- which(results$H==0)

kable(t(data.frame(naive = c(mean(results$naive.cov),mean(results[b,]$naive.cov),
```

Table 4.1: Naive method

	Do not reject null	Reject null
0	0.2893617	0.4184397
1	0.1588652	0.1333333

Table 4.2: Conditional Likelihood Method

	Do not reject null
0	0.7078014
1	0.2921986

```
        conditional = c(mean(results$freq.cov), mean(results[b,]$freq.cov), mean(results
        bayesian = c(mean(results$bayes.cov), mean(results[b,]$bayes.cov), mean(results
      col.names = c("Unconditional Coverage","Coverage Conditional on H0", "Coverage Con
      row.names = TRUE,
      caption = "Empirical Coverage for 95% Confidence/Credible Intervals")
```

\begin{table}
\caption{Empirical Coverage for 95% Confidence/Credible Intervals}

	Unconditional Coverage	Coverage Conditional on H0	Coverage Conditional on H1
naive	0.6212766	0.5911824	0.6941748
conditional	0.8765957	1.0000000	0.5776699
bayesian	0.8765957	1.0000000	0.5776699

\end{table}

```
#kable(c(naive = , conditional = ,bayesian = mean(results[b,]$bayes.cov)), col.names =
#kable(c(naive =c(0) , conditional = mean(results[-b,]$freq.cov),bayesian = mean(resul
```

Hypothesis Rejection(??)

Due to the nature of p-values, an $\alpha = 0.05$ corresponds to a posterior probability $P(H_1|Y)$ of only 0.4 for $N = 100$. This means that the 95% credible interval for $\mu|Y$ will contain 0 every time. In terms of hypothesis testing, if we consider the strategy of rejecting the null when the interval does not contain 0, this level for α leads to no rejections.

```
##confusion matrices
kable(table(results$H,abs(results$Ybar)/sqrt(1/100)<qnorm(1-0.05/2))/dim(results)[1], c
```

```
kable(table(results$H,results$conf.upper>=0&results$conf.lower<=0)/dim(results)[1], col
```

Table 4.3: Bayesian Mixture Model

	Do not reject null
0	0.7078014
1	0.2921986

```
kable(table(results$H,results$cred.upper>=0&results$cred.lower<=0)/dim(results)[1
```

Despite never rejecting the null, the conditional likelihood and the bayesian methods both perform better than the naive one in terms of "predicting" accurately. The naive method is especially problematic in that it has a higher Type 1 error (false positives) than true positives OR true negatives in the region of the data. **## Double Exponential Simulation with Adjusted P-Values**

Chapter 5

Hierarchical Model Simulation

This simulation generates data from a hierarchical logistic model as follows: If there is a true association, μ is a (fixed) nonzero value and $\beta_j \sim N(\mu, \sigma^2)$ where σ^2 is fixed. Otherwise, $\mu = \beta_j = 0, \forall j$. The observed data Y_{ij} is binary, and has $P(Y_{ij}|\beta_j) = \frac{e^{\beta_j}}{1+e^{\beta_j}}$.

The j index corresponds to the "group" to which the observation belongs.

```
knitr::opts_chunk$set(echo = TRUE)
knitr::opts_chunk$set(cache = TRUE)
library("R2jags")
```

```
Loading required package: rjags

Loading required package: coda

Linked to JAGS 4.2.0

Loaded modules: basemod,bugs


Attaching package: 'R2jags'

The following object is masked from 'package:coda':

    traceplot
```

```
set.seed(1)
epsilon=.01
#fix everything (even betas)
mu.p53=.5
phi.p53= 10
observations=1000
n.sites=7
beta.p53.fixed = rnorm(n.sites,mu.p53,phi.p53^(-.5)) #"fixed" because seed
beta.p53.fixed.1 <-beta.p53.fixed
Y <-site <- rep(NA, observations*n.sites)
for(i in 1:n.sites){
    Y[((i-1)*observations+1):(i*observations)]<-rbinom(observations, 1, exp(beta.p53.fix
    site[((i-1)*observations+1):(i*observations)]<- rep(i, observations)
}
sim.data1.1<-list(CaseCon=Y, site=site,  J=n.sites*observations ,n.sites=n.sites)

Y <-site <- rep(NA, observations*n.sites)
for(i in 1:n.sites){
  Y[((i-1)*observations+1):(i*observations)]<-rbinom(observations, 1,.5) #beta=0
  site[((i-1)*observations+1):(i*observations)]<- rep(i, observations)
}
sim.data1.0<-list(CaseCon=Y, site=site,  J=n.sites*observations ,n.sites=n.sites)
#this might just be 0?

set.seed(1)
mu.p53=.1
phi.p53= 10
observations=100
n.sites=7
beta.p53.fixed = rnorm(n.sites,mu.p53,phi.p53^(-.5)) #"fixed" because seed
beta.p53.fixed.2 <-beta.p53.fixed
Y <-site <- rep(NA, observations*n.sites)
for(i in 1:n.sites){
    Y[((i-1)*observations+1):(i*observations)]<-rbinom(observations, 1, exp(beta.p53.fix
    site[((i-1)*observations+1):(i*observations)]<- rep(i, observations)
}
sim.data1.2<-list(CaseCon=Y, site=site,  J=n.sites*observations ,n.sites=n.sites)

source("HPD.R")
#############
```

Since there is uncertainty regarding whether or not there is a true association, this can be modeled as a probability ξ with a beta prior. Then an inverse gamma prior is chosen for variance σ^2 and a spike and slab prior conditioning on the probability of association for mu: $\pi(\mu|\xi) = (1-\xi)\delta_0(\mu) + \xi\phi(\mu)$.

There are a couple of changes that can be made in the model implementation to to obtain MCMC samples:

1. The point mass can be approximated using a very concentrated normal distribution, centered at 0. Thus, $\pi(\mu|\xi) = (1-\xi)N(\mu, 0, \epsilon) + \xi N(\mu, 0, 1)$.

2. The probability ξ can give rise to a latent variable drawn from a bernoulli, which is then used to parametrize the distribution of μ. That is, $\pi(\mu|\iota) = (1-\iota)N(\mu, 0, \epsilon) + \iota N(\mu, 0, 1)$, and $P(\iota = 1) = \xi$.

However, due to the sequential nature of MCMC sampling, these seemingly identical models produce different results.

Models that do not use the latent variable must specify the mixture distribution in JAGS directly. This is done through the zeroes and ones trick, which use properties of the Poisson and Bernoulli distributions in order to allow for arbitrary distributions. In these scenarios, the model from which a sample comes from is also not evident. To determine whether a small value is truly from the concentrated normal, we can compute the probability of the sample for the two distributions and choose the model that results in a larger one.

not sure how relevant this is, maybe will remove different implementations once things work out

```
#model
p53.test.ind = function() {
  for (j in 1:J) {
    CaseCon[j] ~ dbern(theta[j])
    logit(theta[j]) <-  beta.p53[site[j]]  }

  for (l in 1:n.sites) {
    beta.p53.1[l] ~ dnorm(mu.p53, phi.p53)
    beta.p53[l] <- beta.p53.1[l]*(mu.p53.notzero)
  }
  #zeroes trick
  C<-1000
  epsilon<-0.01
  tau<- pow(epsilon,-2)
  #if mu geq 0 and mu leq 0, prior is from point mass
  #mu in (-epsilon, epsilon)
  mu.p53.notzero<- step(abs(mu.p53)-epsilon) #LR and then decide
  L<- (1-mu.p53.notzero)*(1-pind)+ #(dnorm(mu.p53,0,tau)*(1-pind))+ #
    (dnorm(mu.p53,0,1)*pind)
  #need pind for mixing

  phi<- -log(L)+C
  zero~dpois(phi)
  mu.p53 ~ dunif(-10,10) #not sure how big this interval should be, just picked
 
  phi.p53 ~ dgamma(1, .05)
  #    phi.p53 ~ dgamma(2, .02)
  sigma.p53 <- pow(phi.p53, -.5)

  #assoc~dbern(pind)
  pind ~ dbeta(.5,.5)
}
```

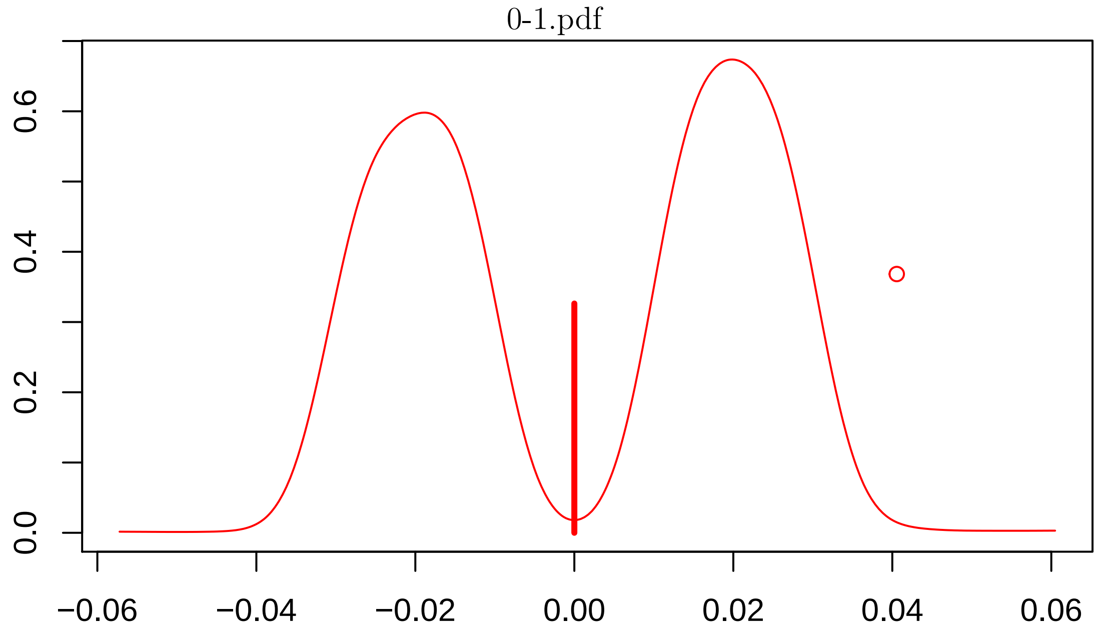
```
p53.test.approx= function() {
  for (j in 1:J) {
    CaseCon[j] ~ dbern(theta[j])
    logit(theta[j]) <-  beta.p53[site[j]]  }

  for (l in 1:n.sites) {
    beta.p53.1[l] ~ dnorm(mu.p53, phi.p53)
    beta.p53[l] <- beta.p53.1[l]*(mu.p53.notzero)
  }
  #zeroes trick
  C<-1000
  epsilon<-0.01
  tau<- pow(epsilon,-2)
  #if mu geq 0 and mu leq 0, prior is from point mass
  #is not zero if prob is greater
  mu.p53.notzero<- step(temp) #should come from this?
  temp<-dnorm(mu.p53,0,1)-dnorm(mu.p53,0,tau)
  L<- (dnorm(mu.p53,0,tau)*(1-pind))+ #(1- mu.p53.notzero)*(1-pind)+ #
    (dnorm(mu.p53,0,1)*pind)
  #need pind for mixing

  phi<- -log(L)+C
  zero~dpois(phi)
  mu.p53 ~ dunif(-10,10) #not sure how big this interval should be, just picked one la

  phi.p53 ~ dgamma(1, .05)
  #    phi.p53 ~ dgamma(2, .02)
  sigma.p53 <- pow(phi.p53, -.5)

  #assoc~dbern(pind)
  pind ~ dbeta(.5,.5)
}


p53.test.latent= function() {
  for (j in 1:J) {
    CaseCon[j] ~ dbern(theta[j])
    logit(theta[j]) <-  beta.p53[site[j]]  }

  for (l in 1:n.sites) {
    beta.p53.1[l] ~ dnorm(mu.p53, phi.p53)
    beta.p53[l] <- beta.p53.1[l]*(mu.p53.notzero)
  }
  #zeroes trick
  C<-1000
  epsilon<-0.01
  tau<- pow(epsilon,-2)
  #if mu geq 0 and mu leq 0, prior is from point mass
  #is not zero if prob is greater
  mu.p53.notzero<- assoc
  temp<-dnorm(mu.p53,0,1)-dnorm(mu.p53,0,tau)
  #L<- (dnorm(mu.p53,0,tau)^(1-assoc))*(dnorm(mu.p53,0,1)^(assoc))
  #need pind for mixing

  #phi<- -log(L)+C
  #zero~dpois(phi)
  #mu.p53 ~ dunif(-10,10) #not sure how big this interval should be, just picked
  mu.p53<- mu1.p53*assoc
  mu1.p53 ~ dnorm(0,1)
  phi.p53 ~ dgamma(1, .05)
  #    phi.p53 ~ dgamma(2, .02)
  sigma.p53 <- pow(phi.p53, -.5)

  assoc~dbern(pind)
  pind ~ dbeta(.5,.5)
}
```

```
p53.fulljags1.0.a = jags(data=sim.data1.0, ####this
                    inits=NULL, parameters.to.save =c("pind", "mu.p53", "phi.p53",
                    model = p53.test.approx)
```

```
module glm loaded

Compiling model graph
   Resolving undeclared variables
   Allocating nodes
Graph information:
   Observed stochastic nodes: 7000
   Unobserved stochastic nodes: 11
   Total graph size: 14076

Initializing model
```

```
#par(mar=rep(1,4))
#plot(as.mcmc(p53.fulljags1.0.a))
HPDM(p53.fulljags1.0.a$BUGSoutput$sims.list$mu.p53, e=epsilon)
```

```
Potentially multimodal column vectors:
 1

Column 1 multimodal intervals:  (-0.036,0.041)
```



```
          Lower     Upper
[1,] -0.03566506 0.0405582
```

```
mean(p53.fulljags1.0.a$BUGSoutput$sims.list$mu.p53)
```

```
[1] 0.001147017
```

```
median(p53.fulljags1.0.a$BUGSoutput$sims.list$mu.p53)
```

```
[1] 0.001183041
```

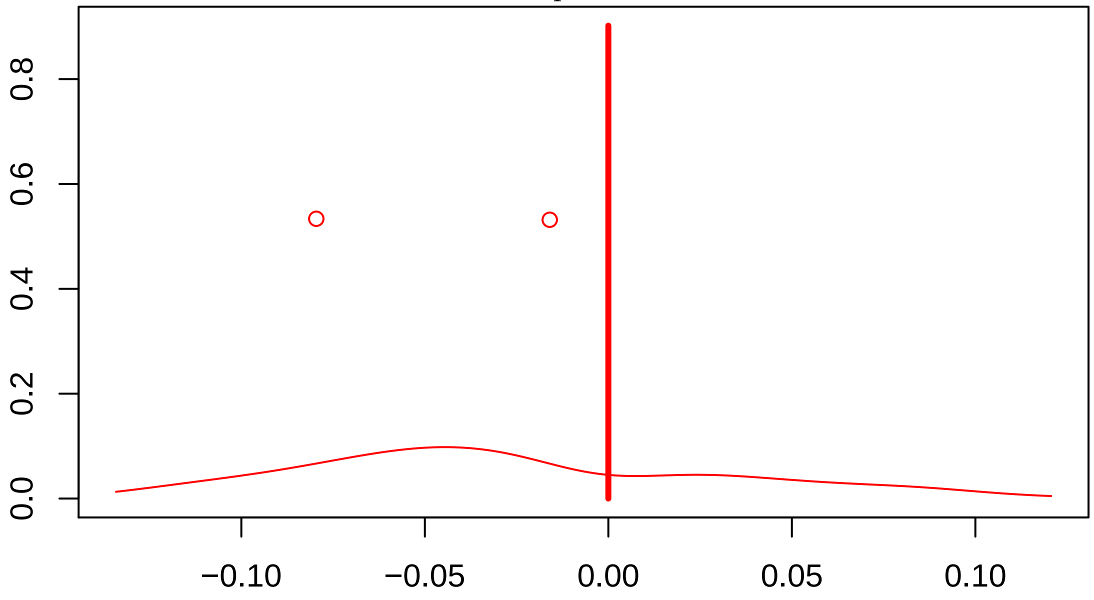
```
p53.fulljags1.0.i = jags(data=sim.data1.0, ####this
                    inits=NULL, parameters.to.save =c("pind", "mu.p53", "phi.p53","mu.p5
                    model = p53.test.ind)
```

```
Compiling model graph
   Resolving undeclared variables
   Allocating nodes
Graph information:
   Observed stochastic nodes: 7000
   Unobserved stochastic nodes: 11
   Total graph size: 14068

Initializing model
```

```
#par(mar=rep(1,4))
#plot(as.mcmc(p53.fulljags1.0.i))
HPDM(p53.fulljags1.0.i$BUGSoutput$sims.list$mu.p53, e=epsilon)
```

```
Potentially multimodal column vectors:
 1

Column 1 multimodal intervals:  (-0.08,-0.016) (-0.01,0.01)
```

0-2.pdf



```
           Lower        Upper
[1,] -0.07958173 -0.01596594 -0.009995439 0.009995481
```

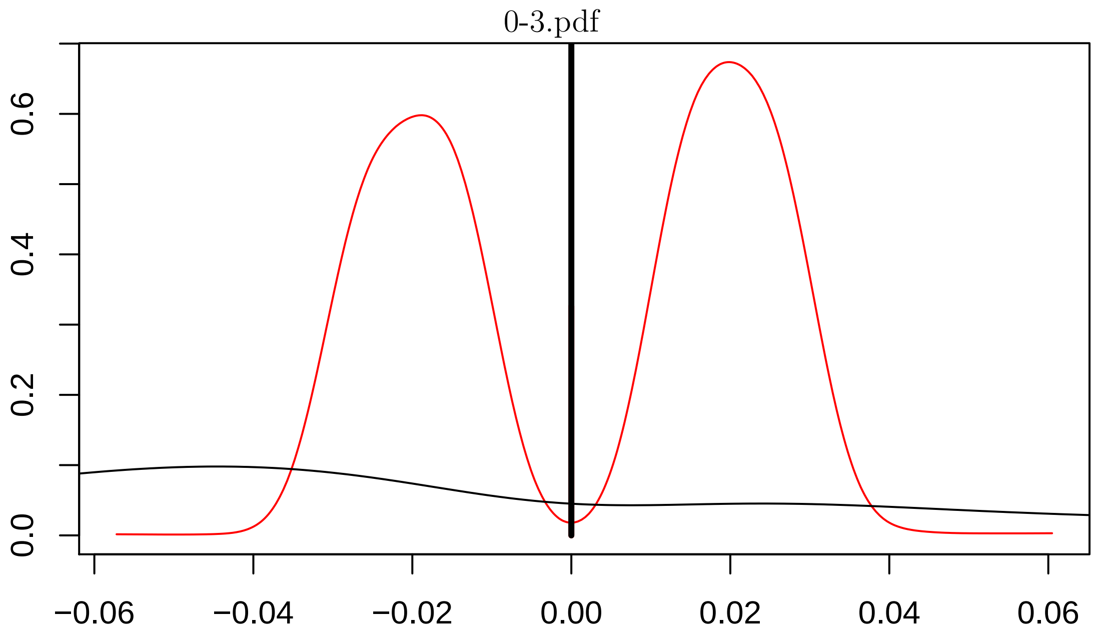
```
mean(p53.fulljags1.0.i$BUGSoutput$sims.list$mu.p53)
```

```
[1] -0.002581946
```

```
median(p53.fulljags1.0.i$BUGSoutput$sims.list$mu.p53)
```

```
[1] -0.0003995983
```

```
plotvar(p53.fulljags1.0.a$BUGSoutput$sims.list$mu.p53,e=epsilon)
plotvar(p53.fulljags1.0.i$BUGSoutput$sims.list$mu.p53,e=epsilon,newplot=FALSE)
```



```
#glm
p53.fullglm1.0 = glm(CaseCon ~ factor(site), data=sim.data1.0,  ####also this
               family=binomial, x=T)

coef(p53.fullglm1.0)
```

```
  (Intercept) factor(site)2 factor(site)3 factor(site)4 factor(site)5
  -0.10810516    0.11210517    0.07610243    0.21219905    0.02405572
factor(site)6 factor(site)7
   0.07610243    0.12810583
```

```
confint(p53.fullglm1.0)
```

```
Waiting for profiling to be done...

                    2.5 %     97.5 %
(Intercept)   -0.23242434 0.01593609
factor(site)2 -0.06328383 0.28763842
factor(site)3 -0.09932072 0.25162370
factor(site)4  0.03675574 0.38791507
factor(site)5 -0.15146721 0.19960978
factor(site)6 -0.09932072 0.25162370
factor(site)7 -0.04727711 0.30365352
```

```
p53.fullglm1.0.simple = glm(CaseCon ~ 1, data=sim.data1.0,  ####also this
               family=binomial, x=T)
coef(p53.fullglm1.0.simple)
```

```
(Intercept)
-0.01828622
```

Overall the Bayesian and frequentist results on this (nonassociated) dataset are very similar. Using a parameter vs an index does produce significantly different results at this stage. Even while using the same prior for pind, the index-based model has a greater amount of "0"s. However, this does not change the posterior of pind (ξ). On the other hand, the normal approx at least shifts it towards 0 (which is what we expect).

```
p53.fulljags1.1.a = jags(data=sim.data1.1, ####this
                         inits=NULL, parameters.to.save =c("pind", "mu.p53", "phi.p53",
                         model = p53.test.approx)
```

```
Compiling model graph
   Resolving undeclared variables
   Allocating nodes
Graph information:
   Observed stochastic nodes: 7000
   Unobserved stochastic nodes: 11
   Total graph size: 14076

Initializing model
```

```
#par(mar=rep(1,4))
#plot(as.mcmc(p53.fulljags1.1.a))
HPDM(p53.fulljags1.1.a$BUGSoutput$sims.list$mu.p53, e=epsilon)
```

```
         Lower     Upper
[1,] 0.2814818 0.6950824
```

```
mean(p53.fulljags1.1.a$BUGSoutput$sims.list$mu.p53)
```

```
[1] 0.4837337
```

```
median(p53.fulljags1.1.a$BUGSoutput$sims.list$mu.p53)
```

```
[1] 0.4825983
```

```
p53.fulljags1.1.i = jags(data=sim.data1.1, ####this
                         inits=NULL, parameters.to.save =c("pind", "mu.p53", "phi.p53",
                         model = p53.test.ind)
```

```
Compiling model graph
   Resolving undeclared variables
   Allocating nodes
Graph information:
   Observed stochastic nodes: 7000
   Unobserved stochastic nodes: 11
   Total graph size: 14068

Initializing model
```

```
#par(mar=rep(1,4))
#plot(as.mcmc(p53.fulljags1.1.i))
HPDM(p53.fulljags1.1.i$BUGSoutput$sims.list$mu.p53, e=epsilon)
```

```
         Lower     Upper
[1,] 0.3022243 0.7036912
```

```
mean(p53.fulljags1.1.i$BUGSoutput$sims.list$mu.p53)
```

```
[1] 0.4848036
```

```
median(p53.fulljags1.1.i$BUGSoutput$sims.list$mu.p53)
```

```
[1] 0.4826411
```

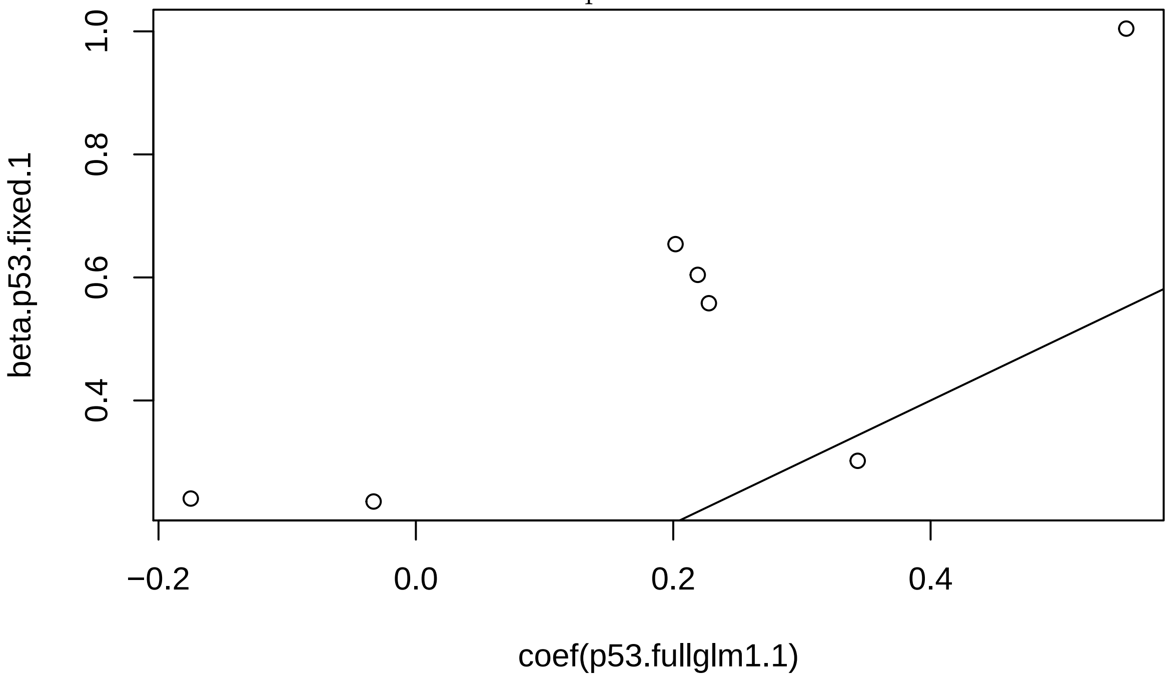
```
#glm
p53.fullglm1.1 = glm(CaseCon ~ factor(site), data=sim.data1.1,  ####also this
                family=binomial, x=T)

summary(p53.fullglm1.1)
```

```

Call:
glm(formula = CaseCon ~ factor(site), family = binomial, data = sim.data1.1,
    x = T)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-1.5735  -1.3263   0.9464   0.9563   1.1068

Coefficients:
              Estimate Std. Error z value Pr(>|z|)
(Intercept)    0.34333    0.06418   5.350 8.82e-08 ***
```

```
factor(site)2  0.22769    0.09195   2.476   0.0133 *
factor(site)3 -0.03286    0.09064  -0.363   0.7169
factor(site)4  0.55205    0.09474   5.827 5.64e-09 ***
factor(site)5  0.21903    0.09189   2.384   0.0171 *
factor(site)6 -0.17494    0.09026  -1.938   0.0526 .
factor(site)7  0.20178    0.09178   2.198   0.0279 *
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 9313.4  on 6999  degrees of freedom
Residual deviance: 9236.2  on 6993  degrees of freedom
AIC: 9250.2

Number of Fisher Scoring iterations: 4
```

```
p53.fullglm1.1.simple = glm(CaseCon ~ 1, data=sim.data1.1,  ####also this
                family=binomial, x=T)

plot(coef(p53.fullglm1.1), beta.p53.fixed.1)
lines(c(0,1),c(0,1))
```

1-1.pdf



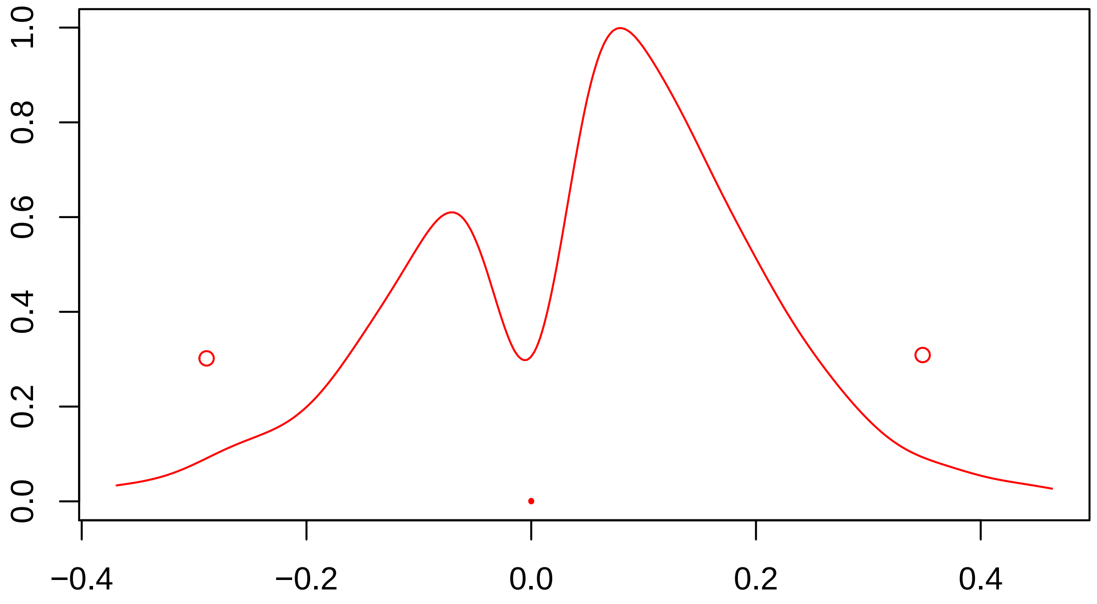
Using $\mu = 0.5$, the posterior results match the intercept. The MCMC dod not select mu = 0 at any point, but this only indicates that the data overwhelms the prior.

```
p53.fulljags1.2.a = jags(data=sim.data1.2, ####this
                         inits=NULL, parameters.to.save =c("pind", "mu.p53", "phi.p53","mu.p5
                         model = p53.test.approx)
```

```
Compiling model graph
   Resolving undeclared variables
   Allocating nodes
Graph information:
   Observed stochastic nodes: 700
   Unobserved stochastic nodes: 11
   Total graph size: 1476

Initializing model
```

```
#par(mar=rep(1,4))
#plot(as.mcmc(p53.fulljags1.2.a))
HPDM(p53.fulljags1.2.a$BUGSoutput$sims.list$mu.p53, e=epsilon)
```

```
Potentially multimodal column vectors:
 1

Column 1 multimodal intervals:  (-0.289,-0.016) (0.013,0.348)
```



```
          Lower        Upper
[1,] -0.2888934 -0.01643942 0.01344605 0.3483093
```

```r
mean(p53.fulljags1.2.a$BUGSoutput$sims.list$mu.p53)
```

```
[1] 0.04721087
```

```r
median(p53.fulljags1.2.a$BUGSoutput$sims.list$mu.p53)
```

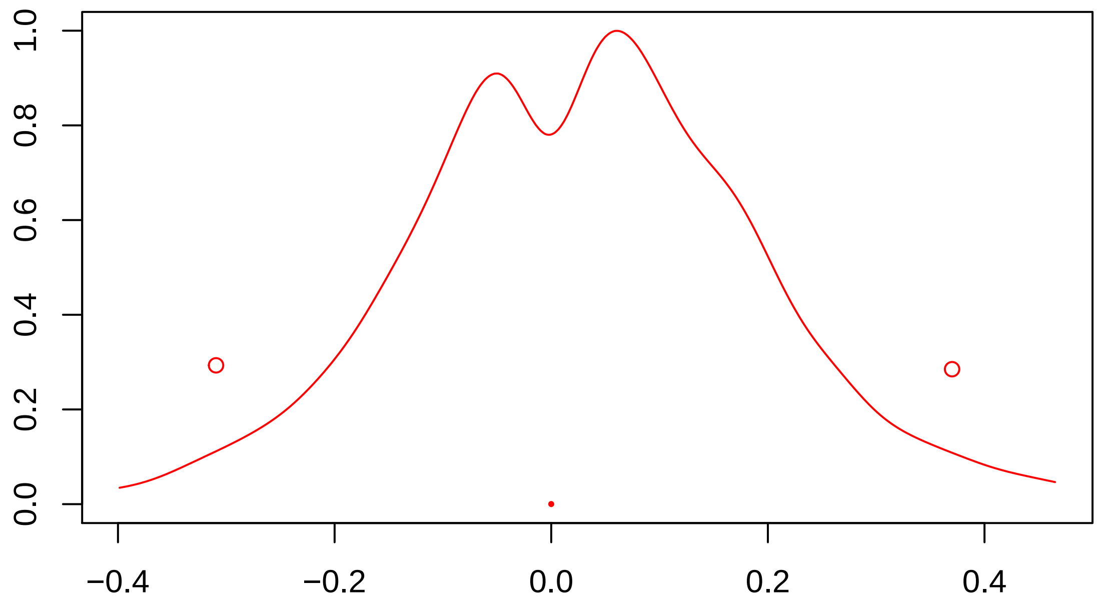
```
[1] 0.06882402
```

```r
p53.fulljags1.2.i = jags(data=sim.data1.2, ####this
                    inits=NULL, parameters.to.save =c("pind", "mu.p53", "phi.p53",
                    model = p53.test.ind)
```

```
Compiling model graph
   Resolving undeclared variables
   Allocating nodes
Graph information:
   Observed stochastic nodes: 700
   Unobserved stochastic nodes: 11
   Total graph size: 1468

Initializing model
```

```r
#par(mar=rep(1,4))
#plot(as.mcmc(p53.fulljags1.2.i))
HPDM(p53.fulljags1.2.i$BUGSoutput$sims.list$mu.p53, e=epsilon)
```

```
Potentially multimodal column vectors:
 1

Column 1 multimodal intervals:  (-0.309,-0.01) (0.01,0.37)
```



```
          Lower        Upper
[1,] -0.3094696 -0.01022024 0.01006981 0.3700762
```

```r
mean(p53.fulljags1.2.i$BUGSoutput$sims.list$mu.p53)
```

```
[1] 0.03330651
```

```r
median(p53.fulljags1.2.i$BUGSoutput$sims.list$mu.p53)
```

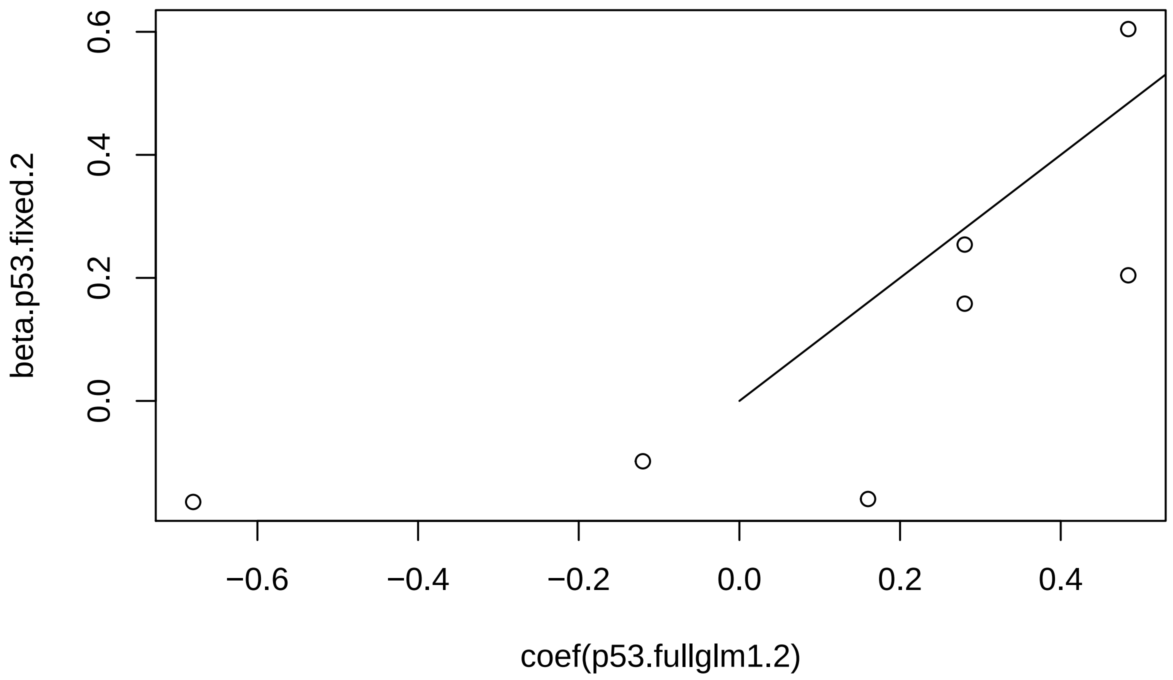
```
[1] 0.03619337
```

```r
#glm
p53.fullglm1.2 = glm(CaseCon ~ factor(site), data=sim.data1.2,  ####also this
               family=binomial, x=T)

summary(p53.fullglm1.2)
```

```

Call:
glm(formula = CaseCon ~ factor(site), family = binomial, data = sim.data1.2,
    x = T)

Deviance Residuals:
   Min       1Q   Median       3Q      Max
-1.335   -1.194    1.027    1.110    1.530

Coefficients:
              Estimate Std. Error z value Pr(>|z|)
```

```
(Intercept)    -0.1201     0.2004  -0.600   0.5487
factor(site)2   0.2805     0.2836   0.989   0.3226
factor(site)3  -0.6800     0.2948  -2.307   0.0211 *
factor(site)4   0.4841     0.2855   1.696   0.0899 .
factor(site)5   0.4841     0.2855   1.696   0.0899 .
factor(site)6   0.1601     0.2831   0.566   0.5716
factor(site)7   0.2805     0.2836   0.989   0.3226
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 970.26  on 699  degrees of freedom
Residual deviance: 947.40  on 693  degrees of freedom
AIC: 961.4

Number of Fisher Scoring iterations: 4
```

```
p53.fullglm1.2.simple = glm(CaseCon ~ 1, data=sim.data1.2,  ####also this
               family=binomial, x=T)

plot(coef(p53.fullglm1.2), beta.p53.fixed.2)
lines(c(0,1),c(0,1))
```



For this $\mu = 0.1$, the posterior appears to be in conflict with expected behavior. However, upon observing the GLM coefficients we can see that the true β_j's were quite dispersed, and that the frequentist and bayesian methods agree in expressing that there is a nonzero effect, but it can positive or negative.

```
p53.fulljags1.0.l = jags(data=sim.data1.0, ####this
                         inits=NULL, parameters.to.save =c("pind", "mu.p53", "phi.p53","mu.p5
                         model = p53.test.latent)
```

```
Compiling model graph
   Resolving undeclared variables
   Allocating nodes
Graph information:
   Observed stochastic nodes: 7000
   Unobserved stochastic nodes: 11
   Total graph size: 14056

Initializing model
```

```
#par(mar=rep(1,4))
#plot(as.mcmc(p53.fulljags1.0.l))
HPDM(p53.fulljags1.0.l$BUGSoutput$sims.list$mu.p53, e=epsilon)
```

```
     Lower Upper
[1,]     0     0
```

```
mean(p53.fulljags1.0.l$BUGSoutput$sims.list$mu.p53)
```

```
[1] 0
```

```
median(p53.fulljags1.0.l$BUGSoutput$sims.list$mu.p53)
```

```
[1] 0
```

```
#ones trick?
p53.test.latent.1= function() {
  for (j in 1:J) {
    CaseCon[j] ~ dbern(theta[j])
    logit(theta[j]) <-  beta.p53[site[j]]  }

  for (l in 1:n.sites) {
    beta.p53.1[l] ~ dnorm(mu.p53, phi.p53)
    beta.p53[l] <- beta.p53.1[l]*(mu.p53.notzero)
  }
  #zeroes trick
  C<-1e6
  epsilon<-0.01
  tau<- pow(epsilon,-2)
  #if mu geq 0 and mu leq 0, prior is from point mass
  #is not zero if prob is greater
  mu.p53.notzero<- step(temp)
  temp<-dnorm(mu.p53,0,1)-dnorm(mu.p53,0,tau) #CHANGED precision to 1/4
  L<-(dnorm(mu.p53,0,tau)*(1-pind))+ #(1- mu.p53.notzero)*(1-pind)+ #
    (dnorm(mu.p53,0,1)*pind)
  p <- L/ C
  for(i in 1:n.ones){
    ones[i] ~ dbern(p)

  }
  mu.p53 ~ dunif(-10,10) #not sure how big this interval should be, just picked o
  #mu.p53<- mu1.p53*assoc
  #mu1.p53 ~ dnorm(0,1)
  phi.p53 ~ dgamma(1, .05)
  #    phi.p53 ~ dgamma(2, .02)
  sigma.p53 <- pow(phi.p53, -.5)

  #assoc~dbern(pind)
  pind ~ dbeta(.5,.5)
}

n.ones=1
onesdata<- sim.data1.0
onesdata$ones<- rep(1,n.ones) #same
onesdata$n.ones<- n.ones
p53.fulljags1.0.l1 = jags(data=onesdata, ####this
                     inits=NULL, parameters.to.save =c("pind", "mu.p53", "phi.p53",
                     model = p53.test.latent.1)
```

```
Compiling model graph
   Resolving undeclared variables
   Allocating nodes
Graph information:
   Observed stochastic nodes: 7001
   Unobserved stochastic nodes: 10
   Total graph size: 14075

Initializing model
```

```
#par(mar=rep(1,4))
#plot(as.mcmc(p53.fulljags1.0.l1))
HPDM(p53.fulljags1.0.l$BUGSoutput$sims.list$mu.p53, e=epsilon)
```

```
     Lower Upper
[1,]     0     0
```

```
mean(p53.fulljags1.0.l$BUGSoutput$sims.list$mu.p53)
```

```
[1] 0
```

```
median(p53.fulljags1.0.l$BUGSoutput$sims.list$mu.p53)
```

```
[1] 0
```

Ones trick is still a little problematic, especially in trace of mu

```
p53.fulljags1.1.l = jags(data=sim.data1.1, ####this
                    inits=NULL, parameters.to.save =c("pind", "mu.p53", "phi.p53","mu.p5
                    model = p53.test.latent)
```

```
Compiling model graph
   Resolving undeclared variables
   Allocating nodes
Graph information:
   Observed stochastic nodes: 7000
   Unobserved stochastic nodes: 11
   Total graph size: 14056

Initializing model
```

```
#par(mar=rep(1,4))
#plot(as.mcmc(p53.fulljags1.1.l))
HPDM(p53.fulljags1.1.l$BUGSoutput$sims.list$mu.p53, e=epsilon)
```

```
         Lower     Upper
[1,] 0.2759939 0.6766949
```

```
mean(p53.fulljags1.1.l$BUGSoutput$sims.list$mu.p53)
```

```
[1] 0.4783651
```

```
median(p53.fulljags1.1.l$BUGSoutput$sims.list$mu.p53)
```

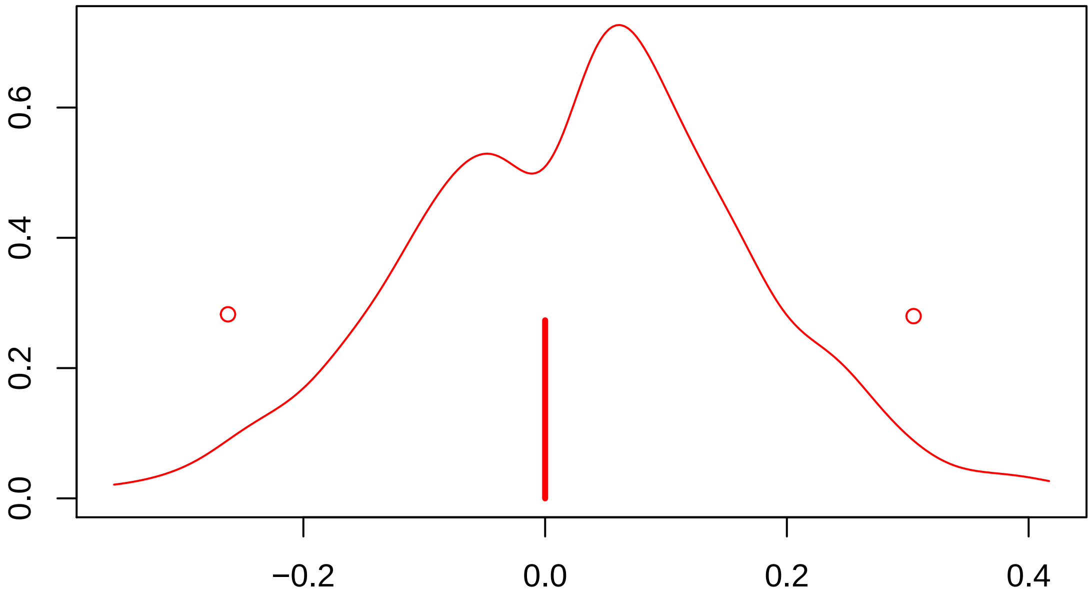
```
[1] 0.4795407
```

```
p53.fulljags1.2.l = jags(data=sim.data1.2, ####this
                    inits=NULL, parameters.to.save =c("pind", "mu.p53", "phi.p53",
                    model = p53.test.latent)
```

```
Compiling model graph
   Resolving undeclared variables
   Allocating nodes
Graph information:
   Observed stochastic nodes: 700
   Unobserved stochastic nodes: 11
   Total graph size: 1456

Initializing model
```

```
#par(mar=rep(1,4))
#plot(as.mcmc(p53.fulljags1.2.l))
HPDM(p53.fulljags1.2.l$BUGSoutput$sims.list$mu.p53, e=epsilon)
```

```
Potentially multimodal column vectors:
 1

Column 1 multimodal intervals:  (-0.262,0.305)
```



```
         Lower     Upper
[1,] -0.262373 0.3048219
```

```
mean(p53.fulljags1.2.1$BUGSoutput$sims.list$mu.p53)
```

```
[1] 0.02193662
```

```
median(p53.fulljags1.2.1$BUGSoutput$sims.list$mu.p53)
```

```
[1] 0
```

```
#precision=4
onesdata<- sim.data1.2
onesdata$n.ones<-1
onesdata$ones<- rep(1, onesdata$n.ones)
p53.fulljags1.2.l1 = jags(data=onesdata, ####this
                    inits=NULL, parameters.to.save =c("pind", "mu.p53", "phi.p53","mu.p5
                    model = p53.test.latent.1)
```

```
Compiling model graph
   Resolving undeclared variables
   Allocating nodes
Graph information:
   Observed stochastic nodes: 701
   Unobserved stochastic nodes: 10
   Total graph size: 1475

Initializing model
```

```
#par(mar=rep(1,4))
#plot(as.mcmc(p53.fulljags1.2.l1))
```

```
#with precision = .25
p53.fulljags1.2.l2 = jags(data=onesdata, ####this
                    inits=NULL, parameters.to.save =c("pind", "mu.p53", "phi.p53","mu.p5
                    model = p53.test.latent.1)
```

```
Compiling model graph
   Resolving undeclared variables
   Allocating nodes
Graph information:
   Observed stochastic nodes: 701
   Unobserved stochastic nodes: 10
   Total graph size: 1475

Initializing model
```

```
#par(mar=rep(1,4))
#plot(as.mcmc(p53.fulljags1.2.l2))
```

This last simulation is an example of Bartlett's paradox: the nonzero part of the prior is too diffuse, and so the zero component ends up being favored.

Chapter 6

Application

```r
knitr::opts_chunk$set(echo = TRUE)
knitr::opts_chunk$set(cache = TRUE)

library(rmeta)
library(lme4)
library("R2jags")
library(xtable)
```

6.1 Genome Study

- description from previous papers

```r
load("tp53.Rdata")
getdata<- function(snp.name, iter=5000, drop= NULL){

 #more iterations, coda traceplots, maybe look at the prior still- cauchy prior?
 #


if(!is.null(drop)) {
  use = !(tp53epi.wsi$site %in% drop)
  } else {
    use = rep(TRUE, nrow(tp53epi.wsi))
  }

  p53.snp = tp53geno.wsi[use, snp.name]
  tp53epi.wsi = tp53epi.wsi[use,]

  missing.geno = is.na(p53.snp)
  site.names = levels(factor(tp53epi.wsi[!missing.geno,"site"]))
  p53.data = list(CaseCon=tp53epi.wsi$casecon[!missing.geno], site=as.numeric(factor(tp5

  p53.df = data.frame(p53.data)

  J = length(p53.data$CaseCon)
  n.sites = length(unique(p53.data$site))
  p53.data$J = J
  p53.data$n.sites = n.sites

  #this is a new indicator
  p53.data$discovery.sites <- which(levels(factor(tp53epi.wsi$site))%in% drop)
  p53.data$missing.geno<-missing.geno

  return(p53.data)

}

snp.name="rs12951053n"
discovery.sitenames= c("POL", "MAY", "NCO")
#regular data
p53.data<- getdata(snp.name)
#validation data
p53.dataval<- getdata(snp.name, drop=discovery.sitenames)
```

```
source("HPD.R")
```

6.1.1 EDA

- plot of data points?
- conditional likelihood, FDR, etc as function of Y
-

```
OR.freq = function(snp.name, tp53epi.wsi,tp53geno.wsi, psdir="ps", drop=NULL,iter=5000,

  if (!is.null(drop)) {
    use = !(tp53epi.wsi$site %in% drop)}
  else {
    use = rep(TRUE, nrow(tp53epi.wsi))
  }

  p53.snp = tp53geno.wsi[use, snp.name]
  tp53epi.wsi = tp53epi.wsi[use,]
  missing.geno = is.na(p53.snp)
  site.names = levels(factor(tp53epi.wsi[!missing.geno,"site"]))

  p53.data = list(CaseCon=tp53epi.wsi$casecon[!missing.geno], site=as.numeric(fact

  p53.df = data.frame(p53.data)
  write.csv(p53.df, file=paste(snp.name, ".csv", sep=""))

  p53.full = glm(CaseCon ~ factor(site) + factor(site)*p53 + factor(site)*Age + BC
 p53.pooled = glm(CaseCon ~ factor(site) + p53 + factor(site)*Age + BC, data=p53.d
   p53.null = glm(CaseCon ~ factor(site) +  factor(site)*Age + BC, data=p53.df, fa

  p53.df$Site = factor(p53.df$site)

  p53.me = glmer(CaseCon ~ BC + p53 + Age + (1|site)  + (0 + p53 | site) + (0 + Ag

  test=anova(p53.full, p53.pooled, p53.null, test="Chi")
  coef = summary(p53.full)$coef
  ns = length(site.names)
  OR = coef[c(ns+1, (ns+4):(ns+ns+2)),1]
  x = p53.full$x
  p = predict(p53.full, type="response")
  var = solve(t(x)%*% diag(p*(1-p)) %*%x)[c(ns+1, (ns+4):(ns+ns+2)),c(ns+1, (ns+4)

  sqrt(diag(var))

  eff = matrix(0, ns,ns)
  eff[,1] = 1
  for (i in 2:ns) eff[i,i] = 1

  OR = eff %*% OR
  OR.SE = sqrt(diag(eff %*% var %*% t(eff)))

  DS = meta.summaries(OR, OR.SE, method="random", names=site.names, logscale=F)
  p.value = pnorm(-(abs(DS$summary/DS$se.summary)))*2
  BF0 = -exp(1)*p.value*log(p.value)
  return(list(snp=snp.name, DS=DS, OR=OR, SE=OR.SE, p.value=p.value, BF.Ha = 1/BF0
}
```

```
validation.sitenames = c("AUS" ,"HAW" ,"MAL" ,"NEC" ,"NHS" ,"SEA" ,"STA" ,"UCI","U

freq<-OR.freq(snp.name, tp53epi.wsi,tp53geno.wsi, psdir="ps", drop=validation.site
freq
```

```
$snp
[1] "rs12951053n"

$DS
Random-effects meta-analysis
Call: meta.summaries(d = OR, se = OR.SE, method = "random", logscale = F,
    names = site.names)
Summary effect=0.303   95% CI (0.075, 0.531)
Estimated heterogeneity variance: 0  p= 0.485

$OR
          [,1]
[1,] 0.2689085
[2,] 0.4317372
[3,] 0.0911083

$SE
[1] 0.2253285 0.1673257 0.2320561

$p.value
[1] 0.009180839

$BF.Ha
[1] 8.542625

$test
Analysis of Deviance Table

Model 1: CaseCon ~ factor(site) + factor(site) * p53 + factor(site) *
    Age + BC
Model 2: CaseCon ~ factor(site) + p53 + factor(site) * Age + BC
Model 3: CaseCon ~ factor(site) + factor(site) * Age + BC
  Resid. Df Resid. Dev Df Deviance Pr(>Chi)
1      2558     2907.6
2      2560     2909.1 -2  -1.4777  0.47766
3      2561     2915.7 -1  -6.6070  0.01016 *
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

$p53.me
Generalized linear mixed model fit by maximum likelihood (Laplace
  Approximation) [glmerMod]
 Family: binomial  ( logit )
Formula: CaseCon ~ BC + p53 + Age + (1 | site) + (0 + p53 | site) + (0 +
    Age | site)
```

```
   Data: p53.df
      AIC       BIC    logLik  deviance  df.resid
 2936.888  2977.844 -1461.444  2922.888      2561
Random effects:
 Groups Name        Std.Dev.
 site   (Intercept) 0.000e+00
 site.1 p53         1.311e-05
 site.2 Age         7.216e-03
Number of obs: 2568, groups:  site, 3
Fixed Effects:
(Intercept)           BC          p53          Age
   -1.73137      0.44222      0.29692      0.01013
```

- put this into a table
- p values,fdr, BF interpretations

6.1.2 model(s)

- rationale for priors
- plot for mixture
- plots of cond likelihood of mu

```
p53.data.aug<-p53.data
p53.data.aug$zeroes<- rep(0,p53.data$n.sites)
p53.data.aug$zero<- 0

p53.data.normal<-p53.dataval
p53.data.normal$n.discovery<- length(p53.data.normal$discovery.sites)
p53.data.normal$zeroes<- rep(0,p53.data.normal$n.discovery)
p53.data.normal$MLE<- freq$OR[,1]
p53.data.normal$SE<- freq$SE
p = 0.00325
p53.data.normal$q<-qnorm(1-p/2)

bfdata <- p53.dataval
bfdata$p <- freq$p.value
```

```
p53.model = function() {
  for (j in 1:J) {
    CaseCon[j] ~ dbern(theta[j])
    logit(theta[j]) <- beta.site[site[j]] + beta.p53[site[j]]*p53[j] +
      beta.Age[site[j]]*Age[j] + beta.BC*BC[j]
  }
  for (k in 1:n.sites) {
    beta.site[k] ~ dnorm(mu.site, phi.site)
    beta.p53[k] ~ dnorm(mu.p53, phi.p53)
    beta.Age[k] ~ dnorm(mu.Age, phi.Age)
  }

  beta.BC ~ dnorm(0, 3)

  mu.site ~ dnorm(0, .1)
  phi.site <- pow(sigma.site, -2)
  sigma.site ~ dunif(0,5)

  mu.p53 ~ dnorm(0, .1)
  phi.p53 <- pow(sigma.p53, -2)
  sigma.p53 ~ dunif(0, 5)

  mu.Age ~ dnorm(0, .1)
  phi.Age <- pow(sigma.Age, -2)
  sigma.Age  ~ dunif(0, 5)

}
```

```
#mixture with association variable
p53.newmodel3 = function() {
  for (j in 1:J) {
    CaseCon[j] ~ dbern(theta[j])
    logit(theta[j]) <- beta.site[site[j]] + beta.p53[site[j]]*p53[j] +
      beta.Age[site[j]]*Age[j] + beta.BC*BC[j]    }

  for (l in 1:n.sites) {
    beta.site[l] ~ dnorm(mu.site, phi.site)
    beta.p53.1[l] ~ dnorm(mu.p53, phi.p53)
    beta.p53[l] <- beta.p53.1[l]*assoc
    beta.Age[l] ~ dnorm(mu.Age, phi.Age)
  }
  beta.BC ~ dnorm(0, .1)

  mu.site ~ dnorm(0, .1)
  phi.site ~ dgamma(1,.05)
  sigma.site <- pow(phi.site, -.5)

  #E[prec] 20
  #(based on range .5 to 2 for OR => range = 1.4 = 6 sigma  sigma = 1.4/6 ~= .2
  mu.p53.1 ~ dnorm(0,.1)
  mu.p53<-mu.p53.1*assoc
  phi.p53 ~ dgamma(1, .05)
  #    phi.p53 ~ dgamma(2, .02)
  sigma.p53 <- pow(phi.p53, -.5)

  mu.Age ~ dnorm(0, .1)
  phi.Age ~ dgamma(1, .05)
  sigma.Age  <- pow(phi.Age, -.5)

  assoc ~ dbern(pind)
  pind ~ dbeta(2,6)
}
```

```
#mixture using zeroes trick (without assoc latent var)
p53.newmodel4 = function() {
  for (j in 1:J) {
    CaseCon[j] ~ dbern(theta[j])
    logit(theta[j]) <- beta.site[site[j]] + beta.p53[site[j]]*p53[j] +
      beta.Age[site[j]]*Age[j] + beta.BC*BC[j]    }

  for (l in 1:n.sites) {
    beta.site[l] ~ dnorm(mu.site, phi.site)
    beta.Age[l] ~ dnorm(mu.Age, phi.Age)
    beta.p53.1[l] ~ dnorm(mu.p53, phi.p53)
    beta.p53[l] <- beta.p53.1[l]*(1-mu.p53.iszero)
  }
  beta.BC ~ dnorm(0, .1)

  mu.site ~ dnorm(0, .1)
  phi.site ~ dgamma(1,.05)
  sigma.site <- pow(phi.site, -.5)

  #zeroes trick
  C<-1000
  epsilon<-0.001
  tau<- pow(epsilon,-2)
  #if mu geq 0 and mu leq 0, prior is from point mass
  #mu in (-epsilon, epsilon)
  mu.p53.iszero<- step(epsilon-abs(mu.p53))
  L<- (mu.p53.iszero*(1-pind))+ #(dnorm(mu.p53,0,tau)*(1-pind))+
    (dnorm(mu.p53,0,1)*pind)
  #need pind for mixing

  phi<- -log(L)+C
  zero~dpois(phi)
  mu.p53 ~ dunif(-2,2) #not sure how big this interval should be, just picked one larg

  phi.p53 ~ dgamma(1, .05)
  #    phi.p53 ~ dgamma(2, .02)
  sigma.p53 <- pow(phi.p53, -.5)

  mu.Age ~ dnorm(0, .1)
  phi.Age ~ dgamma(1, .05)
  sigma.Age  <- pow(phi.Age, -.5)
  #assoc~dbern(pind)
  pind ~ dbeta(2,10)
}
```

```
#conditional likelihood prior (normal approx, using zeroes trick)
p53.normal = function() {
  for (j in 1:J) {
    CaseCon[j] ~ dbern(theta[j])
    logit(theta[j]) <- beta.site[site[j]] + mu.p53*p53[j] +
      beta.Age[site[j]]*Age[j] + beta.BC*BC[j]    }

  for (l in 1:n.sites) {
    beta.site[l] ~ dnorm(mu.site, phi.site)
    #beta.p53[l] ~ dnorm(mu.p53, phi.p53)
    beta.Age[l] ~ dnorm(mu.Age, phi.Age)
  }
  C<-1000

  for (k in 1:n.discovery){
    #zeroes trick for MLE~cond prob
    tau[k]<- pow(SE[k], -2)

    L[k]<- dnorm(MLE[k],mu.p53, tau[k])/(pnorm(-q*SE, mu.p53, tau[k]) +
                                        1-pnorm(q*SE, mu.p53, tau[k]))
    phi[k]<- -log(L[k])+C
    zeroes[k]~dpois(phi[k])
  }

  beta.BC ~ dnorm(0, .1)

  mu.site ~ dnorm(0, .1)
  phi.site ~ dgamma(1,.05)
  sigma.site <- pow(phi.site, -.5)

  #E[prec] 20
  #(based on range .5 to 2 for OR => range = 1.4 = 6 sigma  sigma = 1.4/6 ~= .2
  mu.p53 ~ dnorm(0,.1)
  phi.p53 ~ dgamma(1, .05)
  #    phi.p53 ~ dgamma(2, .02)
  sigma.p53 <- pow(phi.p53, -.5)

  mu.Age ~ dnorm(0, .1)
  phi.Age ~ dgamma(1, .05)
  sigma.Age  <- pow(phi.Age, -.5)
  #assoc~dbern(pind) *assoc
  pind ~ dbeta(2,6)
}
```

```
#eplogp approximation of BF for posterior prob of association
p53.bf.approx = function() {
  for (j in 1:J) {
    CaseCon[j] ~ dbern(theta[j])
    logit(theta[j]) <- beta.site[site[j]] + mu.p53*p53[j] +
      beta.Age[site[j]]*Age[j] + beta.BC*BC[j]    }

  for (l in 1:n.sites) {
    beta.site[l] ~ dnorm(mu.site, phi.site)
    beta.p53.1[l] ~ dnorm(mu.p53, phi.p53)
    #beta.p53[l] <- beta.p53.1[l]*(assoc)
    beta.Age[l] ~ dnorm(mu.Age, phi.Age)
  }
  beta.BC ~ dnorm(0, .1)
  mu.site ~ dnorm(0, .1)
  phi.site ~ dgamma(1,.05)
  sigma.site <- pow(phi.site, -.5)

  #E[prec] 20
  #(based on range .5 to 2 for OR => range = 1.4 = 6 sigma  sigma = 1.4/6 ~= .2
  mu.p53.1 ~ dnorm(0,.1)
  mu.p53<-mu.p53.1*assoc
  phi.p53 ~ dgamma(1, .05)
  #    phi.p53 ~ dgamma(2, .02)
  sigma.p53 <- pow(phi.p53, -.5)

  mu.Age ~ dnorm(0, .1)
  phi.Age ~ dgamma(1, .05)
  sigma.Age  <- pow(phi.Age, -.5)

  assoc~dbern(post.ind)
  prior.odds<- (1-pind)/pind
  BF<- -exp(1)*p*log(p)
  post.ind<-prior.odds*BF/(1+prior.odds*BF)
  pind ~ dbeta(2,6)
}
```

```
parameters = c("mu1.site", "mu1.p53",  "beta.BC", "beta.Age", "mu.site","mu.p53", "mu.Ag

parameters4 = c("beta.BC", "beta.Age", "mu.site","mu.p53", "mu.Age", "sigma.site", "sigm

parameters.normal<- c("beta.BC", "beta.Age", "mu.site","mu.p53", "mu.Age", "sigma.site",

p53.sim = jags(data=p53.data, inits=NULL, parameters.to.save =parameters, model = p53.mo
```

```
module glm loaded

Warning in jags.model(model.file, data = data, inits = init.values,
n.chains = n.chains, : Unused variable "discovery.sites" in data

Warning in jags.model(model.file, data = data, inits = init.values,
n.chains = n.chains, : Unused variable "missing.geno" in data

Compiling model graph
   Resolving undeclared variables
   Allocating nodes
Graph information:
   Observed stochastic nodes: 11182
   Unobserved stochastic nodes: 46
   Total graph size: 59386

Initializing model

Warning in jags.samples(model, variable.names, n.iter, thin, type = "trace", : Failed to s
Variable mu1.site not found

Warning in jags.samples(model, variable.names, n.iter, thin, type = "trace", : Failed to s
Variable mu1.p53 not found

Warning in jags.samples(model, variable.names, n.iter, thin, type = "trace", : Failed to s
Variable assoc not found

Warning in jags.samples(model, variable.names, n.iter, thin, type = "trace", : Failed to s
Variable pind not found

Warning in jags.samples(model, variable.names, n.iter, thin, type = "trace", : Failed to s
Variable phi1.site not found
```

```
Warning in jags.samples(model, variable.names, n.iter, thin, type = "trace", : Faile
Variable phi1.p53 not found
```

```r
p53.simval = jags(data=p53.dataval, inits=NULL, parameters.to.save =parameters, mo
```

```
Warning in jags.model(model.file, data = data, inits = init.values,
n.chains = n.chains, : Unused variable "discovery.sites" in data

Warning in jags.model(model.file, data = data, inits = init.values,
n.chains = n.chains, : Unused variable "missing.geno" in data

Compiling model graph
   Resolving undeclared variables
   Allocating nodes
Graph information:
   Observed stochastic nodes: 8614
   Unobserved stochastic nodes: 37
   Total graph size: 45664

Initializing model

Warning in jags.samples(model, variable.names, n.iter, thin, type = "trace", : Faile
Variable mu1.site not found

Warning in jags.samples(model, variable.names, n.iter, thin, type = "trace", : Faile
Variable mu1.p53 not found

Warning in jags.samples(model, variable.names, n.iter, thin, type = "trace", : Faile
Variable assoc not found

Warning in jags.samples(model, variable.names, n.iter, thin, type = "trace", : Faile
Variable pind not found

Warning in jags.samples(model, variable.names, n.iter, thin, type = "trace", : Faile
Variable phi1.site not found

Warning in jags.samples(model, variable.names, n.iter, thin, type = "trace", : Faile
Variable phi1.p53 not found
```

```r
p53.simnew4 = jags(data=p53.data.aug, inits=NULL, parameters.to.save =parameters4,
```

```
Warning in jags.model(model.file, data = data, inits = init.values,
n.chains = n.chains, : Unused variable "discovery.sites" in data

Warning in jags.model(model.file, data = data, inits = init.values,
n.chains = n.chains, : Unused variable "missing.geno" in data
```

```
Warning in jags.model(model.file, data = data, inits = init.values,
n.chains = n.chains, : Unused variable "zeroes" in data

Compiling model graph
   Resolving undeclared variables
   Allocating nodes
Graph information:
   Observed stochastic nodes: 11183
   Unobserved stochastic nodes: 47
   Total graph size: 59459

Initializing model

Warning in jags.samples(model, variable.names, n.iter, thin, type = "trace", : Failed to s
Variable assoc not found

Warning in jags.samples(model, variable.names, n.iter, thin, type = "trace", : Failed to s
Variable beta.p53.iszero not found
```

```
p53.simnormal = jags(data=p53.data.normal, inits=NULL, parameters.to.save =parameters.no
```

```
Warning in jags.model(model.file, data = data, inits = init.values,
n.chains = n.chains, : Unused variable "discovery.sites" in data

Warning in jags.model(model.file, data = data, inits = init.values,
n.chains = n.chains, : Unused variable "missing.geno" in data

Compiling model graph
   Resolving undeclared variables
   Allocating nodes
Graph information:
   Observed stochastic nodes: 8614
   Unobserved stochastic nodes: 28
   Total graph size: 45642

Initializing model

Warning in jags.samples(model, variable.names, n.iter, thin, type = "trace", : Failed to s
Variable beta.p53 not found

Warning in jags.samples(model, variable.names, n.iter, thin, type = "trace", : Failed to s
Variable assoc not found
```

```
p53.bfsim = jags(data=bfdata, inits=NULL, parameters.to.save =parameters, model =
```

```
Warning in jags.model(model.file, data = data, inits = init.values,
n.chains = n.chains, : Unused variable "discovery.sites" in data

Warning in jags.model(model.file, data = data, inits = init.values,
n.chains = n.chains, : Unused variable "missing.geno" in data

Compiling model graph
   Resolving undeclared variables
   Allocating nodes
Graph information:
   Observed stochastic nodes: 8614
   Unobserved stochastic nodes: 39
   Total graph size: 45659

Initializing model

Warning in jags.samples(model, variable.names, n.iter, thin, type = "trace", : Faile
Variable mu1.site not found

Warning in jags.samples(model, variable.names, n.iter, thin, type = "trace", : Faile
Variable mu1.p53 not found

Warning in jags.samples(model, variable.names, n.iter, thin, type = "trace", : Faile
Variable phi1.site not found

Warning in jags.samples(model, variable.names, n.iter, thin, type = "trace", : Faile
Variable phi1.p53 not found

Warning in jags.samples(model, variable.names, n.iter, thin, type = "trace", : Faile
Variable beta.p53 not found
```

```
p53.simnew3 = jags(data=p53.data, inits=NULL, parameters.to.save =parameters, mode
```

```
Warning in jags.model(model.file, data = data, inits = init.values,
n.chains = n.chains, : Unused variable "discovery.sites" in data

Warning in jags.model(model.file, data = data, inits = init.values,
n.chains = n.chains, : Unused variable "missing.geno" in data

Compiling model graph
   Resolving undeclared variables
   Allocating nodes
Graph information:
   Observed stochastic nodes: 11182
   Unobserved stochastic nodes: 48
   Total graph size: 59407

Initializing model
```

```
Warning in jags.samples(model, variable.names, n.iter, thin, type = "trace", : Failed to s
Variable mu1.site not found

Warning in jags.samples(model, variable.names, n.iter, thin, type = "trace", : Failed to s
Variable mu1.p53 not found

Warning in jags.samples(model, variable.names, n.iter, thin, type = "trace", : Failed to s
Variable phi1.site not found

Warning in jags.samples(model, variable.names, n.iter, thin, type = "trace", : Failed to s
Variable phi1.p53 not found
```

```r
library(dplyr)
getOR<-function(sim){

OR = as.data.frame(sim$BUGSoutput$sims.matrix)%>%select( starts_with("beta.p53"), "mu.p
exp(OR)
#colnames(OR) = c(site.names, "Overall")

sum.OR = t(apply(exp(OR), 2, function(x) {PI = HPDinterval(as.mcmc(x))
 return(c(median(x), PI[1], PI[2]))}
))
 return(sum.OR)
}

OR1<-getOR(p53.sim) #regular
OR3<-getOR(p53.simnew3) #w assoc
OR4<-getOR(p53.simnew4) #w point mass ind
ORn<-getOR(p53.simnormal) #MLEs for prior
ORbf<-getOR(p53.bfsim) #MLEs for prior
```

```r
ORtable = data.frame(original= OR1["mu.p53",], latentvar= OR3["mu.p53",],
                  zerotrick= OR4["mu.p53",] , cond= ORn["mu.p53",], bfapprox= ORbf["mu.p5
rownames(ORtable)<- c("Median", "2.5%","97.5%")
kable(ORtable)
```

	original	latentvar	zerotrick	cond	bfapprox
Median	1.183197	1.000000	1.1849025	1.159779	1.000000
2.5%	1.042845	1.000000	0.9963451	1.011020	1.000000
97.5%	1.355430	1.246941	1.3741023	1.324258	1.154497

- discussion of OR
- at least one example of a posterior?
- plot CI comparisons
- plot something like the shrinkage
- plot CI length comparison
- some notes on disjoint CI's, log vs not logged HPD

References

Bacanu, S.-A., & Kendler, K. S. (2013). Extracting actionable information from genome scans. *Genetic Epidemiology*, *37*(1), 48–59.

Benjamin, D. J., Berger, J. O., Johannesson, M., Nosek, B. A., Wagenmakers, E.-J., Berk, R., … others. (2017). Redefine statistical significance. *Nature Human Behaviour*.

Bigdeli, T. B., Lee, D., Webb, B. T., Riley, B. P., Vladimirov, V. I., Fanous, A. H., … Bacanu, S.-A. (2016). A simple yet accurate correction for winner's curse can predict signals discovered in much larger genome scans. *Bioinformatics*, *32*(17), 2598–2603.

Ferguson, J. P., Cho, J. H., Yang, C., & Zhao, H. (2013). Empirical bayes correction for the winner's curse in genetic association studies. *Genetic Epidemiology*, *37*(1), 60–68.

Ghosh, A., Zou, F., & Wright, F. A. (2008). Estimating odds ratios in genome scans: An approximate conditional likelihood approach. *The American Journal of Human Genetics*, *82*(5), 1064–1074.

Jiang, W., & Yu, W. (2016). Power estimation and sample size determination for replication studies of genome-wide association studies. *BMC Genomics*, *17*(1), 19.

Sellke, T., Bayarri, M., & Berger, J. O. (2001). Calibration of ρ values for testing precise null hypotheses. *The American Statistician*, *55*(1), 62–71.

Sun, L., Dimitromanolakis, A., Faye, L. L., Paterson, A. D., Waggott, D., Bull, S. B., … others. (2011). BR-squared: A practical solution to the winner's curse in genome-wide scans. *Human Genetics*, *129*(5), 545–552.

Xu, L., Craiu, R. V., & Sun, L. (2011). Bayesian methods to overcome the winner's curse in genetic studies. *The Annals of Applied Statistics*, *5*, 201–231.

Xu, S. (2003). Theoretical basis of the beavis effect. *Genetics*, *165*(4), 2259–2268.

Yekutieli, D. (2012). Adjusted bayesian inference for selected parameters. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, *74*(3), 515–541.

Zhong, H., & Prentice, R. L. (2008). Bias-reduced estimators and confidence intervals for odds ratios in genome-wide association studies. *Biostatistics*, *9*(4), 621–634.

Zöllner, S., & Pritchard, J. K. (2007). Overcoming the winner's curse: Estimating penetrance parameters from case-control data. *The American Journal of Human Genetics*, *80*(4), 605–615.