Bayesian Statistics
Exercises 4.

1. Consider the following setting in the context of "multiple hypothesis tests". Let $i = 1, \dots, n$ index individuals and $j = 1, \dots, m$ index genes (or pixels in an image if you prefer). Assume we have measurements on each individual at each gene for "treatments" $k = 0, 1$. Let Y_{ijk} denote the measurement on individual i, gene j, treatment k, and let D_{ij} denote the difference between the measurements in the two treatments: $D_{ij} := Y_{ij0} - Y_{ij1}$. We will assume that D is sufficient for all our inferences, and so you can forget about Y now and work only with D: I just wanted you to understand where D might come from in principle.

   We will assume a model for D:

   $$D_{ij} | \beta, \sigma \sim N(\beta_j, \sigma_j^2) \tag{1}$$

   where $\beta = (\beta_1, \dots, \beta_m)$ is a vector of "treatment effect"s, where β_j is the effect at gene j, and $\sigma = (\sigma_1, \dots, \sigma_m)$ is a vector of standard deviation parameters. For each gene j we wish to test the null $H_j : \beta_j = 0$ (that is, that there is no treatment effect). For simplicity you can assume that $\sigma_j = 1$ is known for all j. You can also assume that $n = 10$ and $m = 1,000$.

   Assume that the true effects β_j are independent, and identically distributed, with

   $$\beta_j \sim \pi_0 \delta_0 + (1 - \pi_0) N(0, \sigma_b^2). \tag{2}$$

   where δ_0 denotes a point mass on zero. That is, $\beta_j = 0$ with probability π_0, and $\beta_j \sim N(0, \sigma_b^2)$ with probability $1 - \pi_0$.

   i) Explain how (1) and (2) together define a hierarchical model for the data and treatment effects at each gene. Given π_0, σ_b, are the β_j exchangeable? What about unconditional on π_0, σ_b? What about the data matrix D_{ij} - which aspects of this matrix are exchangeable, if any? Explain your reasoning.

   ii) Write an R function to simulate data D under this hierarchical model, for user-specified π_0 and σ_b. The function should take π_0 and σ_b as input, and return a list, with elements D (a matrix) and β (a vector).

iii) Write an R function to compute a p value p_j for each column of the data matrix D, testing $H_0 : \beta_j = 0$. This function should take as input the data matrix D and output a vector of p values. You can use any reasonable two-sided test, but state which test you use. Apply your R function to data simulated under a) $\pi_0 = 1$, b) $\pi_0 = 0.5, \sigma_b = 3$; c) $\pi_0 = 0, \sigma_b = 3$. Provide histograms of the p values in each case and comment on their distributions.

iv) Write an R function to apply the Benjamini–Hochberg rule to control FDR at a user-specified level α. This function should input a vector of p values, and a level α, and output a vector of binary (0/1) indicators, $\gamma = (\gamma_1, \ldots, \gamma_m)$ say, where $\gamma_j = 1$ indicates that the rule would reject $H_j : \beta_j = 0$.

v) Write an R function to compute the empirical False Discovery Rate (i.e. the number of false discoveries divided by the number of discoveries) for any given value for the vector β of true values of β, and the vector γ of reject decisions. That is, the function should return V/R in the notation of the notes. Remember to deal correctly with the special case of no discoveries, $R = 0$.

vi) Perform a simulation study to estimate the actual FDR ($\mathrm{E}(V/R)$) achieved by the BH rule in the three cases a), b) and c) above. In each case perform the test procedure for different levels α, and plot the estimated $\mathrm{E}(V/R)$ as a function of α (say for $\alpha = (0.05, 0.1, \ldots, 0.5)$). Comment on the results. [NOTE: to estimate the actual FDR you have to estimate $\mathrm{E}(V/R)$ where the expectation is over datasets D. To do this you will want to do a simulation study where you simulate a large number of datasets D, not just one dataset!]

vii) Repeat the simulation study, but this time estimate the pFDR instead of the FDR, and plot this as a function of α.

2. The `qvalue` package in R implements Storey's approach to estimating FDR. To install this package use

```
source("http://bioconductor.org/biocLite.R")
biocLite("qvalue")
library("qvalue")
```

The package takes a vector of p values, and outputs a list which includes an estimate of π_0 (obtained using the p values near 1) and a vector of q values. Try, for example, for a vector of p values `p`,

```
res=qvalue(p)
res$pi0
res$qvalues
```

The q value for a particular observation is an estimate of the pFDR if you reject all things that are as or more significant than that observation. You can convert the vector of q values into a list of reject decisions at a given α level (the γ vector above) using, say,

```
compute.gamma=function(q,alpha){return(q<alpha)}
```

i) Repeat the simulation study above, using `qvalue` instead of the BH procedure. Produce plots of the FDR vs the α level for `qvalue` and compare them with those obtained for BH.

ii) Perform a simulation study (e.g. by modifying the simulations you have already performed), to see how accurately `qvalue` is able to estimate the proportion of nulls π_0. Try varying π_0 from 0 to 1 for at least 3 different values of σ_b, and in each case provide plots of the true π_0 vs the estimated π_0 from `qvalue`. Comment on the results.

3. Now consider implementing an Empirical Bayes approach to this problem. To do so, given data D we will need two steps:

   A Estimate the hyper parameters π_0, σ_b in (2) by maximum likelihood. Call the estimates $\hat{\pi}_0, \hat{\sigma}_b$.

   B Compute the posterior distribution $p(\beta_j | D, \hat{\pi}_0, \hat{\sigma}_b)$ for each j.

   This question takes you through these two steps.

   i) Define $\bar{D}_j = (1/n)\sum_i D_{ij}$. Show that the vector $\bar{D} := (\bar{D}_1, \ldots, \bar{D}_m)$ is sufficient for β. That is, $p(D|\beta) \propto p(\bar{D}|\beta)$ where the constant of proportionality does not depend on β. [This means that, as far as inference for β is concerned, the likelihood $p(D|\beta)$ is equivalent to the likelihood $p(\bar{D}|\beta)$, so from now on you can treat $\bar{D}$ as your data instead of D.]

ii) Derive an expression for the log-likelihood $l(\pi_0, \sigma_b) := \log(p(\bar{D}|\pi_0, \sigma_b))$. [Hint: note that the $\bar{D}_j$ are independent given π_0, σ_b]

iii) Write an R function to compute the log-likelihood $l(\pi_0, \sigma_b)$, or alternatively $l(\theta_1, \theta_2)$ where $\theta_1 = \log(\pi_0/(1-\pi_0)), \theta_2 = \log(\sigma_b)$. [The motivation for this reparameterization is that θ_1, θ_2 can take any value on the real line.] Try using the R function optimize (or another method if you prefer) to maximize the likelihood over π_0, σ_b (or θ_1, θ_2). [You may or may not find that this works... it is a somewhat tricky numerical problem. The reparameterization may help. Alternatively if you know about the EM algorithm you can try that.]

iv) Derive the posterior distribution $\beta_j|D, \pi_0, \sigma_b$. Hint: this posterior should be a mixture of a point mass at zero and a normal distribution. It may help to first derive $p(\beta_j = 0|D, \pi_0, \sigma_b)$, and then $p(\beta_j|D, \pi_0, \sigma_b, \beta_j \neq 0)$.

v) Implement a method that computes $p(\beta_j = 0|D, \hat{\pi}_0, \hat{\sigma}_b)$. Implement another method that takes these probabilities and rejects those tests j for which this probability is $< \alpha$. Add this method to your simulation study and see how it performs. (If you are unable to get the optimization for π, σ_b to work then you can "cheat" in this step and use the true value of π, σ_b.)