BGmix

Alex Lewin*, Natalia Bochkina†

April 21, 2009

*Centre for Biostatistics, Department of Epidemiology and Public Health,
Imperial College London
†School of Mathematics, University of Edinburgh
http://www.bgx.org.uk/alex/
a.m.lewin@imperial.ac.uk

Contents

1 Overview

The `BGmix` package implements the models used in Lewin et al. (2007) for finding differential expression for genes in two or more samples. When there are two samples, a 3-component mixture is used to classify genes as over-expressed, under-expressed and non-differentially expressed, and

gene variances are modelled exchangeably to allow for variability between genes. The model is fully Bayesian, estimating the proportion of differentially expressed genes and the mixture parameters simultaneously. The model can also be run with unstructured priors, for use with multi-class data.

Several different parametric models are possible. An important part of the analysis is to check if the model is a reasonable fit to the data, and we do this via predictive checks.

The analysis is carried out using Markov Chain Monte Carlo. Convergence of the output can be checked using the `coda` package available from CRAN. We also provide a function to plot the trace of the parameters as part of the `BGmix` package.

Two alternatives are provided for assessing error rates. With the mixture model, an estimate of the false discovery rate (FDR) based on posterior probabilities can be calculated. For unstructured priors, a tail posterior probability method (Bochkina & Richardson (2007)) can be used.

The input to the model can be expression data processed by any algorithm. We provide a function `readBGX` to read in the output from the package `BGX`, which is a fully Bayesian hierarchical model for obtaining gene expression measures (Hein et al. (2005)).

2 Data format

Data input to `BGmix` consists of sample mean and sample variance for each gene, under each experimental condition. Three R objects are required as arguments to the `BGmix` function:

- `ybar`: a matrix, whose columns correspond to experimental conditions and rows correspond to genes. Each column contains sample means for all genes under one condition.
- `ss`: a matrix, whose columns correspond to experimental conditions and rows correspond to genes. Each column contains sample variances for all genes under one condition. Sample variances must be the unbiased estimates, i.e. divide by no. replicates - 1 (this is the default for the standard R `var` function).
- `nreps`: a vector containing the number of replicates in each condition.

Note that for a paired design, the data is treated as having only one 'condition', and `ybar` is then the mean *difference* between the two experimental conditions.

The data must be transformed so that Normal sampling errors are a reasonable assumption (eg. with a log or shifted-log transform), and normalised if necessary.

3 Experimental design

The first level of the model can be written as a regression for each gene:

$$\bar{y}_g = X^T.\beta_g + \epsilon_g \tag{1}$$

where $\bar{y}_g$ is the vector of sample means for different conditions and β_g is a vector of effect parameters. Different parametrisations can be achieved using the 'design' matrix X. At most 1 effect

parameter can have a mixture prior. (This will generally be the differential expression parameter.)

By default, genes have a separate variance parameter for each condition (σ^2_{gc}). However, a more general variance structure can be used, for instance each gene can have one variance across all conditions (σ^2_g).

The `BGmix` function takes four arguments relating to the parametrisation:

- `xx`: design matrix X. The dimensions of X must be no. effects x no. conditions.
- `jstar`: label of the effect which has the mixture prior. Labels start at 0, since this is passed to C++. If all parameters are fixed effects, set `jstar` = -1.
- `ntau`: the number of variance parameters for each gene.
- `indtau`: label for each condition indicating which variance grouping that condition belongs to. The length of `indtau` must be the same as the number of conditions.

The defaults for these parameters are those for the **differential expression, unpaired data** case (see below).

3.1 Differential expression, unpaired data

For unpaired data β_{g1} is the overall mean for gene g and β_{g2} is the differential expression parameter. Here $X = \begin{pmatrix} 1 & 1 \\ -1/2 & 1/2 \end{pmatrix}$, `jstar` = 1.
Two variance structures are commonly used: for gene variances per condition (σ^2_{gc}), use `ntau` = 2, `indtau` = 0:1. For one variance across all conditions (σ^2_g), use `ntau` = 1, `indtau` = 0.

3.2 Differential expression, paired data

For paired data there is only one condition and one effect, which is the differential expression parameter. Here $X = 1$, `jstar` = 0, `ntau` = 1, `indtau` = 0.

3.3 Multi-class data

If one fixed effect per condition is required, set X to be the identity matrix and `jstar` = -1. For gene variances per condition (σ^2_{gc}), use `ntau` = no. conditions, `indtau = 0:(nconds-1)`. For one variance across all conditions (σ^2_g), use `ntau` = 1, `indtau` = 0.

4 Example: How to run the model

At a minimum, you must consider the data set and experimental design parameters in order to run the model (see Sections 2 and 3).

We demonstrate BGmix on a small simulated data set. This consists of 8 replicates of 1000 genes in 2 experimental conditions. We look for differential expression between the two conditions, with

an unpaired design.

First read in the data:

```
> library(BGmix)
> data(ybar, ss)
```

The default experimental design parameters are those for unpaired differential expression, so these can be left out here. The following command fits BGmix using a mixture of a point mass at zero for the null distribution and a Gamma and a reflected Gamma for the alternatives.

```
> outdir <- BGmix(ybar, ss, nreps = c(8, 8), niter = 1000, nburn = 1000)
```

```
[1] "Mixture prior on comp. 2"
[1] "delta ~ Gamma, MH"
[1] "eta (scale of Gamma) updated"
[1] "lambda (shape of Gamma) not updated"
[1] "Normal Likelihood"
[1] "tau ~ Gamma"
[1] "a (prior for tau) is updated"
[1] "trace output for parameters"
[1] "no trace for predicted data"
 Burn-in:
10 20 30 40 50 60 70 80 90 100 110 120 130 140 150 160 170 180 190 200 210 220 230 240 250 260
 Main up-dates:
10 20 30 40 50 60 70 80 90 100 110 120 130 140 150 160 170 180 190 200 210 220 230 240 250 260
[1] "Output directory is ./run.1"
```

The function `BGmix` returns the output directory name. The output directory contains several types of file:

- summary of model options (summary.txt)
- posterior means (mean*.txt)
- probability of being classified in the null component (prob-class.txt)
- trace of posterior distribution (trace*.txt)
- predictive p-values (pval*.txt)

Output data can be read into R with the functions `ccSummary` , `ccParams` (this reads in posterior means and classification probabilities), `ccTrace` and `ccPred`.

5 Plotting the results

First read in posterior means:

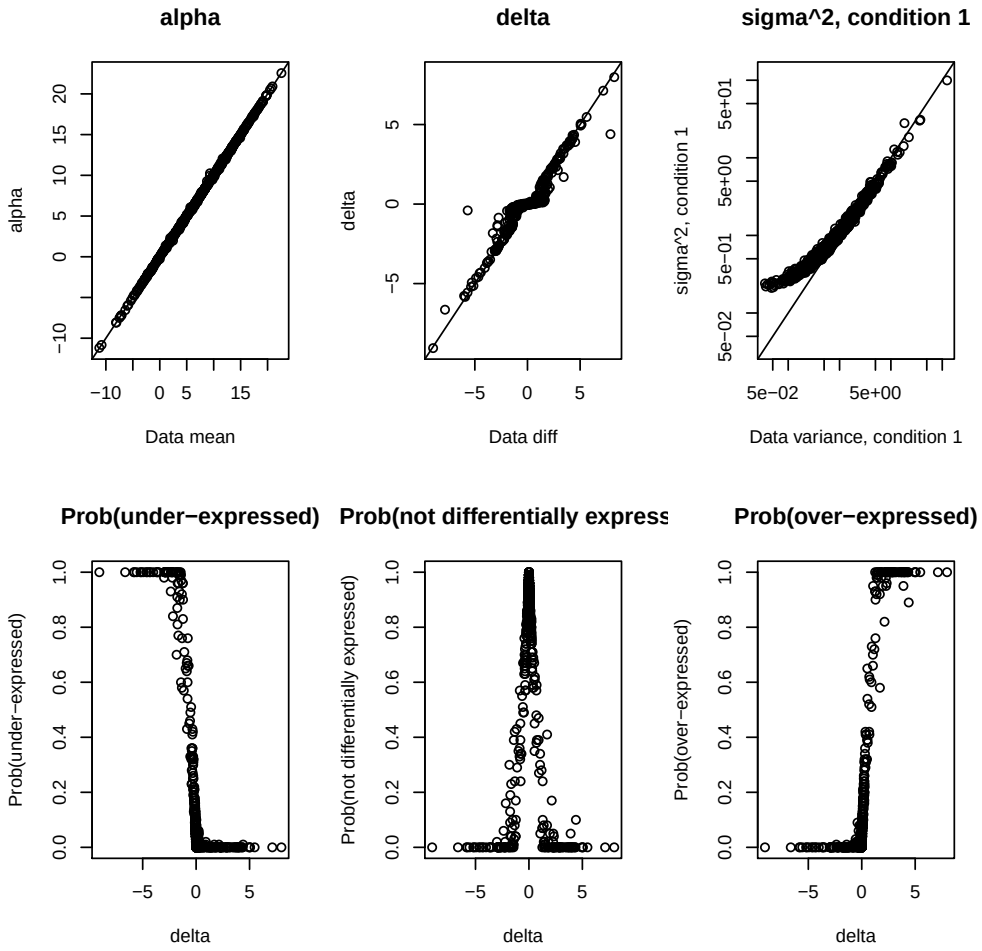
```
> params <- ccParams(file = outdir)
```

```
[1] "got beta"
[1] "got sig2"
[1] "got zg"
```

The output of `ccParams` is a list of vectors and matrices corresponding to the different model parameters. These are easily plotted using standard R functions. For an unpaired differential expression design some standard plots are included in the package. These show smoothing of parameters and classification of genes into different mixture components:

```
> plotBasic(params, ybar, ss)
```

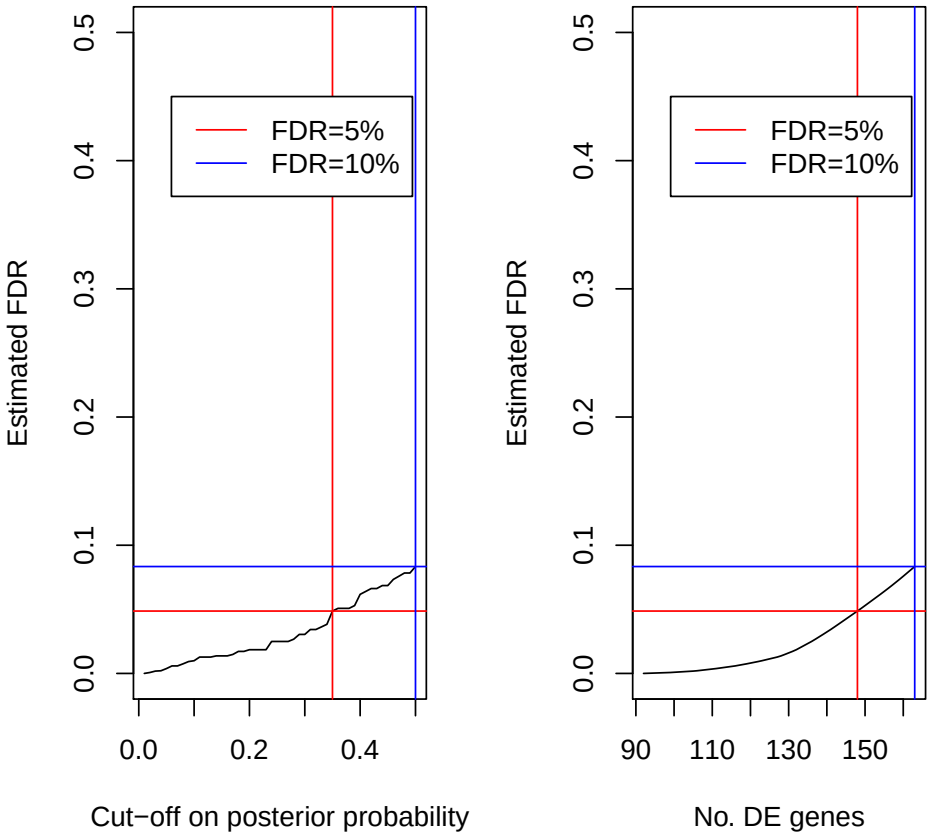
```
[1] "These plots are designed for differential expression model."
```



The estimated FDR (false discovery rate) can also be plotted:

```
> par(mfrow = c(1, 2))
> fdr <- calcFDR(params)
> plotFDR(fdr)
```

```
[1] "No. DE genes for FDR<=5% is  148"
[1] "No. DE genes for FDR<=10% is  163"
```



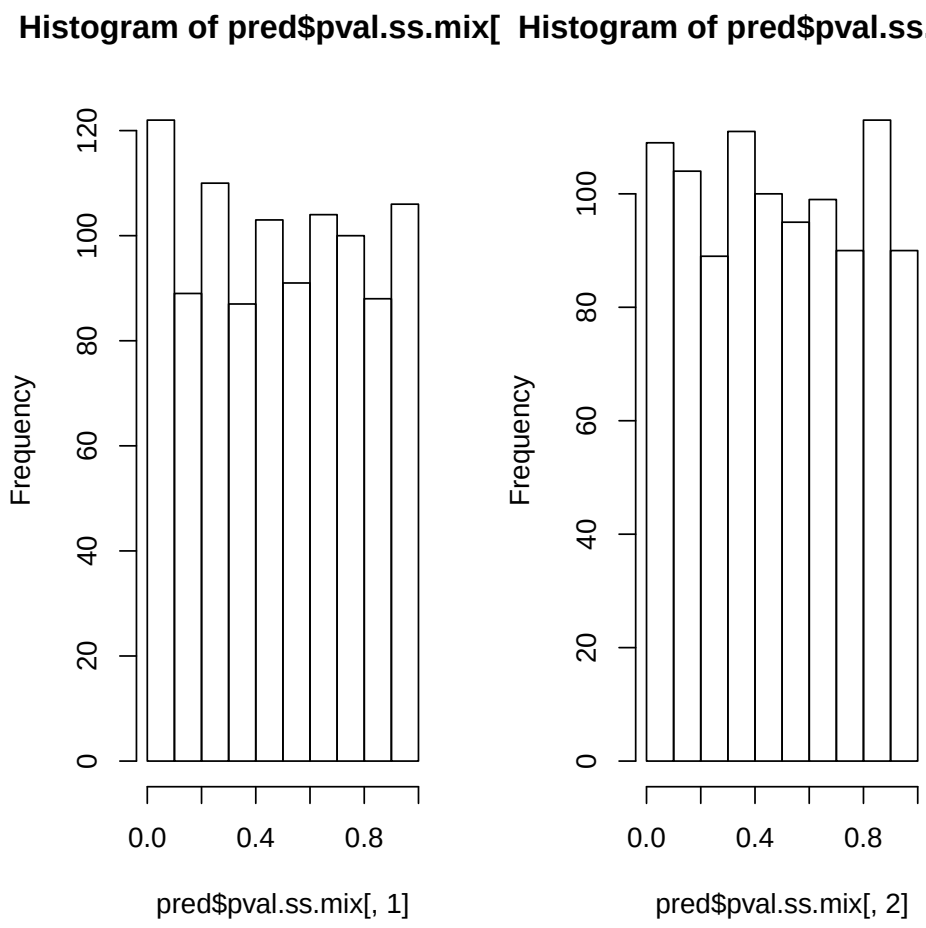
6 Predictive model checking

First read in the predictive p-values:

```
> pred <- ccPred(file = outdir)
```

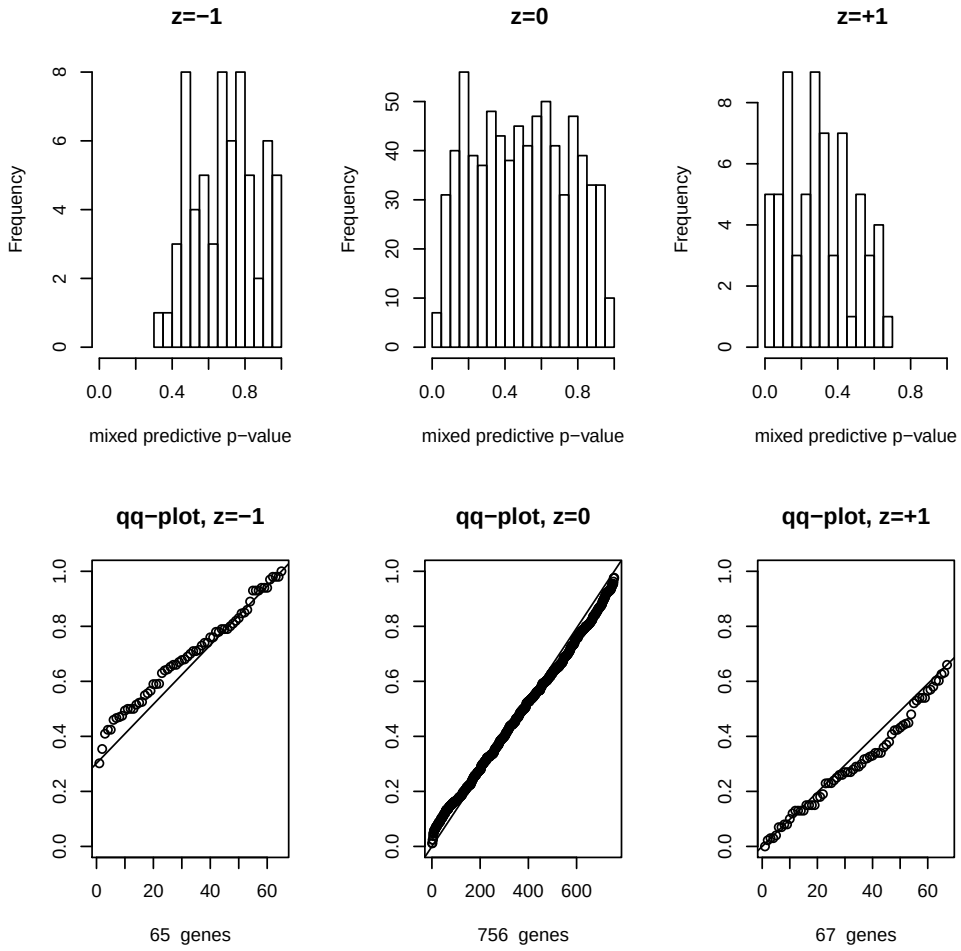
It is a good idea to look at histograms of the predictive p-values for the gene variances:

```
> par(mfrow = c(1, 2))
> hist(pred$pval.ss.mix[, 1])
> hist(pred$pval.ss.mix[, 2])
```



For mixture models, there is a specific function to plot histograms of predictive p-values corresponding to each of the mixture components.

```
> par(mfrow = c(2, 3))
> plotPredChecks(pred$pval.ybar.mix2, params$pc, probz = 0.8)
```



7 Tail posterior probability

Tail posterior probability is used to find differentially expressed genes with unstructured prior for the difference (fixed effects). It needs trace and parameters output from BGmix with `jstar = -1` (all effects are fixed):

```
> nreps <- c(8, 8)
> outdir2 <- BGmix(ybar, ss, nreps = nreps, jstar = -1, niter = 1000,
+     nburn = 1000)
> params2 <- ccParams(outdir2)
> res2 <- ccTrace(outdir2)
```

and the tail posterior probability is calculated by

```
> tpp.res <- TailPP(res2, nreps = nreps, params2, p.cut = 0.7,
+     plots = F)
```

Note that in this function the tail posterior probability is calculated only for the second contrast, assuming that it is the difference between condition 2 and condition 1 for the default contrast

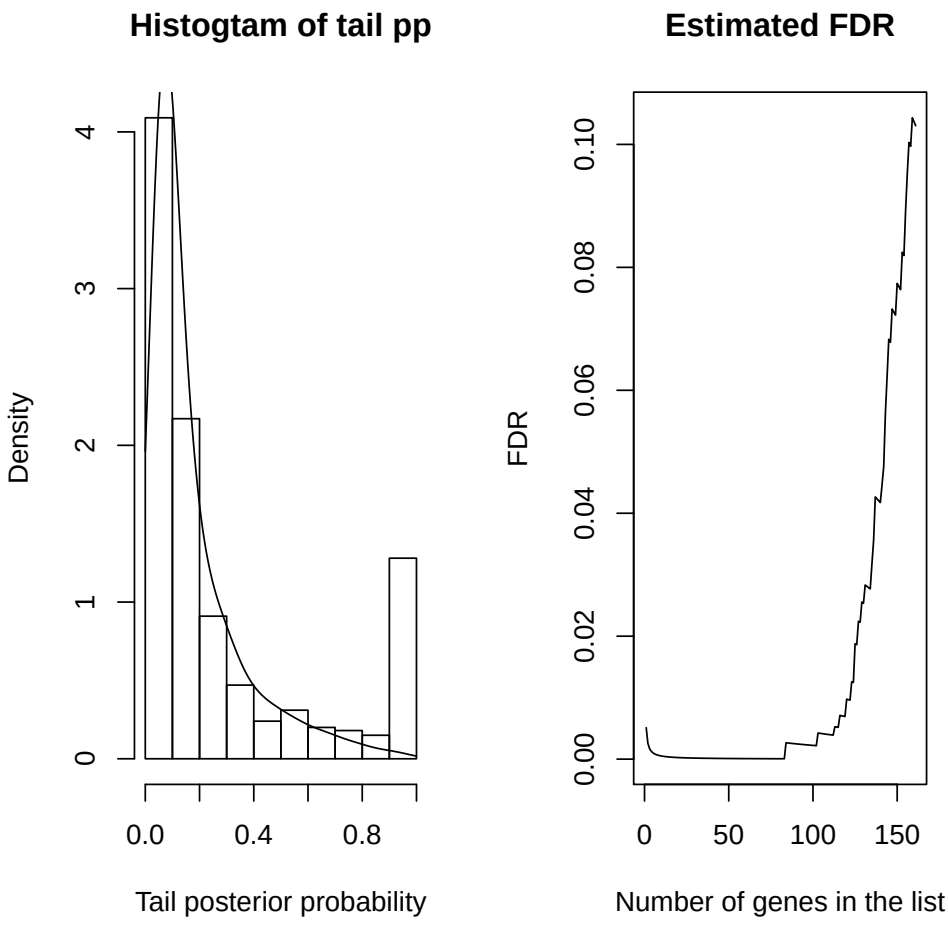
matrix X (see Section 3.1), or for the first contrast in paired data.

The returned values are the tail posterior probabilities `tpp`, estimated False Discovery Rate `FDR` and estimated proportion of non-differentially expressed genes `pi0`. FDR and `pi0` can also be estimated separately:

```
> FDR.res <- FDRforTailPP(tpp.res$tpp, a1 = params2$maa[1], a2 = params2$maa[2],
+     n.rep1 = nreps[1], n.rep2 = nreps[2], p.cut = 0.8)
> pi0 <- EstimatePi0(tpp.res$tpp, tpp.res$pp0)
```

The histogram of the tail posterior probabilities with its density under the null (no differentially expressed genes) and a graph of FDR can be plotted. (These plots can be done in function `TailPP` by setting arguments `plots = TRUE`.)

```
> par(mfrow = c(1, 2))
> histTailPP(tpp.res)
> FDRplotTailPP(tpp.res)
```



8 Model options

To run the model with a flat prior on all effects (no mixture prior), set `jstar` = -1.
There are three main choices for the mixture prior, as presented in Lewin et al. (2007). These are controlled by the option `move.choice.bz` in `BGmix`:

- Null point mass, alternatives Uniform (`move.choice.bz = 1`)
- Null point mass, alternatives Gamma (`move.choice.bz = 4`)
- Null small Normal, alternatives Gamma (`move.choice.bz = 5`)

There are two alternatives for the prior on the gene variances. These are controlled by the option `move.choice.tau` in `BGmix`:

- Inverse Gamma (`move.choice.tau = 1`)
- log Normal (`move.choice.tar = 2`)

9 Other functions

`plotTrace` plots trace plots of model parameters (useful for assessing convergence of the MCMC)

`plotCompare` produces scatter plot of two variables using the same scale for x and y axes

`plotMixDensity` plots predictive density for mixture model (Note: you must save the trace of the predicted data for this: option `trace.pred=1` in `BGmix` and option `q.trace=T` in `ccPred`)

`TailPP` plots the tail posterior probability (for use with unstructured priors on the effect parameters)

`readBGX` reads in results from the `BGX` package.

10 Acknowledgements

Thanks to Ernest Turro for invaluable help getting the package to work.

References

Bochkina, N. and Richardson, S. (2007). Tail posterior probability for inference in pairwise and multiclass gene expression data. *Biometrics* in press.

Hein, A.-M. K., Richardson, S., Causton, H. C., Ambler, G. K., and Green, P. J. (2005). BGX: a fully Bayesian gene expression index for Affymetrix GeneChip data. *Biostatistics* **6(3)**, 349–373.

Lewin, A. M., Bochkina, N. and Richardson, S. (2007). Fully Bayesian mixture model for differential gene expression: simulations and model checks. *submitted*.