XDE: A Bayesian hierarchical model for analysis of differential expression in multiple studies

Robert Scharpf, Andrew Nobel, Giovanni Parmigiani, Håkon Tjelmeland

October 17, 2016

1 Introduction

There are many publicly available high throughput gene expression studies that address comparable biological questions with similar patient populations. For economical and practical reasons many of these studies have a relatively small number of biological replicates. To improve the statistical power it is of interest to combine observed data from several microarray studies, potentially measured with different technologies. However, variation in the measured gene expression levels is caused not only by the biological conditions of interest and natural variation in gene expression in different samples of the same type, but also to a great extent by technological differences between studies. To successfully combine observed data from different studies, it is therefore essential to filter out the technological effects. The R package *XDE* fits a Bayesian Hierarchical model for estimation of differential expression in 2 or more studies. Alternative methods for assessing differential expression are provided. Other R packages developed for the cross-study analysis of differential expression are *GeneMeta* **?**, *metaArray* **?**, and *rankProduct* **?**.

After reading this vignette, one should be able to

- create an instance of the class `ExpressionSetList`
- create an instance of the class `XdeParameter` that contains default options for the MCMC
- fit the Bayesian hierarchical model to two or more studies stored as an `ExpressionSetList` class
- plot MCMC chains to assess convergence
- compute posterior averages for concordant and discordant differential expression
- generate alternative cross-study summaries of differential expression

See **?** for a more detailed discussion of the Bayesian hierarchical model

2 The `ExpressionSetList` Class

Presently the software for fitting the Bayesian hierarchical model requires that each study be represented as an `ExpressionSet` **?** and that the same features be present in each study (the `featureNames` of

the studies must be identical). Therefore, for the meta-analysis of several platforms, platform-specific annotations must be mapped to a common reference identifier. Mapping identifiers in each platform is a non-trivial stop in any cross-study analysis. The R packages *funcBox* (not yet publicly available) and *MergeMaid* have been developed to facilitate the annotation process and the merging of multiple studies, respectively **?**.

For the purposes of this vignette, *XDE* provides an example dataset of a single study that was split into three artificial datasets.

```
> library(XDE)
> data(expressionSetList)
> xlist <- expressionSetList
> class(xlist)
```

```
[1] "ExpressionSetList"
attr(,"package")
[1] "XDE"
```

The original study is described in **?**. The processed data was mapped to unigene identifiers and made available in the experimental data package *lungExpression* (http://www.bioconductor.org) to facilitate the reproducibility of the analyses described in **?**.

The function `validObject` checks that the each element in `ExpressionSetList` is a valid `ExpressionSet` and that the `featureNames` are the same in each element.

```
> validObject(xlist)
```

```
[1] TRUE
```

In order to assess differential expression across multiple studies, one must define a dichotomous covariate in the `phenoData` of each `ExpressionSet` element in the `ExpressionSetList` object. This covariate must have an identical name in each study. In this vignette, we will use *XDE* to quantify differential expression between adenocarcinomas and squamous carcinomas. The binary covariate "adenoVsquamous" has been defined in each of the `ExpressionSet` elements for this purpose. The following statement must evaluate to `TRUE`:

```
> stopifnot(all(sapply(xlist, function(x, label){ label %in% varLabels(x)}, label="adenoVsquamous")))
```

3 The `XdeParameter` Class

There are many features in our implementation of the hierarchical Bayesian model that can be modified:

- hyperparameters for the Bayesian hierarchical model
- the seed for generating random values in the MCMC
- the starting values for the MCMC chains

- tuning parameters for Metropolis-Hastings proposals
- the number of updates per MCMC iteration (each parameter can have a different number of updates)
- selection of MCMC chains that are to be written to log files

All of the above features are provided in the `XdeParameter` class. The attributes provided when initializing an instance of class `XdeParameter` work well in most instances. The `XdeParameter` vignette provides a more detailed description of how the attributes can be modified, as well as a brief description of the Bayesian model and the algorithm for the MCMC. See **?** for a more detailed description of the Bayesian model.

The default values for our `xlist` object are obtained by initializing an instance of the `XdeParameter` class:

```
> params <- new("XdeParameter", esetList=xlist, phenotypeLabel="adenoVsquamous")
> params

Instance of XdeParameter

hyperparameters:
alpha.a  beta.a    p0.a    p1.a alpha.b  beta.b    p0.b
    1.0     1.0     0.1     0.1     1.0     1.0     0.1
   p1.b
    0.1
...

updates (frequency of updates per MCMC iteration):
   nu Delta     a     b    c2
    1     1     3     3     1
...

tuning (the epsilon for Metropolis-Hastings proposals):
   nu Delta     a     b    c2
 0.01  0.01  0.04  0.04  0.01
...

output (parameters to save (0 = not saved, 1 = saved to log file):
 potential acceptance         nu     DDelta          a
         1          1          1          1          1
...

iterations:  1000
thin:  1
seed:  211075
notes:
firstMcmc:
List of 5
 $ Nu    : num [1:1500] -0.126 -0.49 0.362 0.263 0.767 ...
 $ DDelta: num [1:1500] 0.6171 -0.2698 0.3534 -0.0489 0.4996 ...
```

```
 $ A     : num [1:3] 0.6 0.641 0.531
 $ B     : num [1:3] 0.478 0.882 0.844
 $ C2    : num 0.472
...

showIterations:  TRUE
specifiedInitialValues:  TRUE
directory (where to save the MCMC chains):  ./
phenotypeLabel:  adenoVsquamous
studyNames:  study1 study2 study3
one.delta:  TRUE
```

This parameterization, presently the default, assumes that a gene is differentially expressed in all studies or in none. An alternative parametrization that allows genes to be differentially expressed in a subset of studies can be obtained by setting the argument `one.delta` to `FALSE`.

```
> params <- new("XdeParameter", esetList=xlist,
+               phenotypeLabel="adenoVsquamous", one.delta=FALSE)
```

Whether one fits the "δ_g" (`one.delta=TRUE`) or the "δ_{gp}" model (`one.delta=FALSE`), the chain written to file is of dimension G x P x I, where G is the number of genes, P is the number of studies, and S is the number of samples (in the case of the δ_g model, the value written to file for a single gene will be the same for each study. The same is true for the ξ_p parameter.)

4 Fitting the Bayesian hierarchical model

4.1 Starting values

Randomly simulated starting values By default, the first iteration of the MCMC chain stored in the slot `firstMcmc` of the `params` are simulated randomly from the priors. When the value of `burnin` is true, the output from the MCMC are not saved to file and the chain can not be monitored for convergence. By default the value of `burnin` is TRUE and only the last iteration from the chain will be available (the parameters are not written to log files).

Empirical starting values One can use empirical values for starting the chain (or specify your own starting values) by initializing an object of class XdeParameter and then specifying your own values for the first MCMC:

```
> params <- new("XdeParameter", esetList=xlist, phenotypeLabel="adenoVsquamous", one.delta=FALSE)
> empirical <- empiricalStart(xlist, phenotypeLabel="adenoVsquamous")
> firstMcmc(params) <- empirical
```

To run a burnin of 3 iterations starting from the empircal values:

```
> iterations(params) <- 3
> burnin(params) <- TRUE
> fit <- xde(params, xlist)
```

Only the first and last iterations of the MCMC are available when `burnin` is `TRUE`. The output of the `xde` is an object of class `XdeMcmc`. The object `fit` contains the last iteration from the MCMC, as well as a different seed that can be used for initiating the next chain. For instance, to run two additional iterations starting at the last iteration stored in the `fit` object, one should provide the `params`, `xlist`, and `fit` objects to the `xde` function:

```
> iterations(params) <- 2
> fit2 <- xde(params, xlist, fit)
```

When an object of class `XdeMcmc` is supplied as an argument to the `xde` function, the seed and the last iteration from the `fit` object are used to begin the next chain. Note that the results from the previous call to the `xde` would be identical to the following sequence of commands:

```
> firstMcmc(params) <- lastMcmc(fit)
> seed(params) <- seed(fit)
> fit2 <- xde(params, xlist)
```

One should run several thousand iterations (saving all parameters to file) to monitor convergence. In the following code chunk (not run) we save only the chains for the parameters that are not indexed by genes and/or study by setting the `output` for these parameters to zero – this step was taken to keep this package from becoming unnecessarily large. By setting the `thin` to 2, we only write every other MCMC iteration to file. In total, 1000 iterations are saved.

```
> burnin(params) <- FALSE
> iterations(params) <- 2000
> output(params)[c("potential", "acceptance",
+                  "diffExpressed",
+                  "nu", ##"DDelta",
+                  ##"delta",
+                  "probDelta",
+                  ##"sigma2",
+                  "phi")] <- 0
> thin(params) <- 2
> directory(params) <- "logFiles"
> xmcmc <- xde(params, xlist)
```

See the `XdeParameter` vignette for a more detailed discussion of the output, burnin, and thin methods. The `xmcmc` object can be loaded by:

5 MCMC diagnostics

In this section we briefly describe how to access the chains for assessing convergence of the MCMC. We refer the Ruser to the package *coda* for more detailed discussion of MCMC diagnostics **?**.

The following is a list of the chains that were saved in our run with 1000 saved iterations:

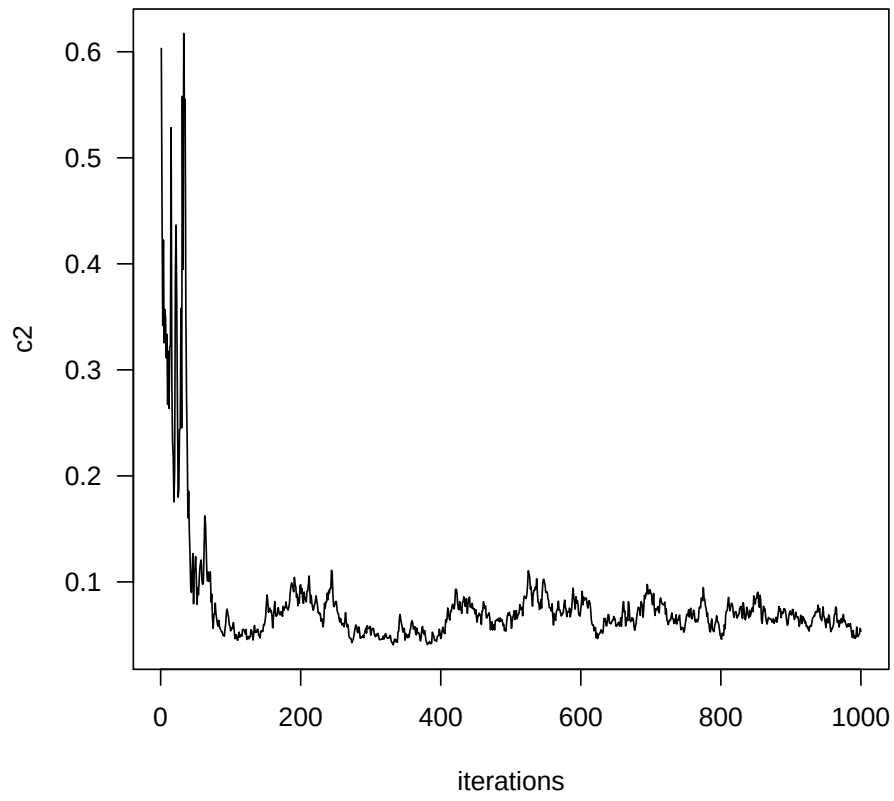
```
> output(xmcmc)[2:22][output(xmcmc)[2:22] == 1]
```

```
      a       b      c2  gamma2       r     rho      xi
      1       1       1       1       1       1       1
      t       l   theta  lambda   tau2R tau2Rho
      1       1       1       1       1       1
```

The $ operator can be used to read in log files. First, we need to update the `directory` slot in the `xmcmc` with a character string indicating the path to the log files. (Note that an assignment method for directory is only available for R object of class `XdeMcmc` – typically one would not need to change the directory in the `XdeMcmc` object). In the following code chunk, we extract the chain for the c^2 parameter:

```
> pathToLogFiles <- system.file("logFiles", package="XDE")
> xmcmc@directory <- pathToLogFiles
> c2 <- xmcmc$c2
```

```
> par(las=1)
> plot.ts(c2, ylab="c2", xlab="iterations", plot.type="single")
```



Here we extract only the parameters that are not indexed by gene and platform (log files for parameters indexed by gene and platform are typically very large):

```
> getLogs <- function(object){
+         params <- output(object)[output(object) == 1]
+         params <- params[!(names(params) %in% c("nu", "phi", "DDelta", "delta", "sigma2", "diffExpre
+         names(params)
+ }
> param.names <- getLogs(xmcmc)
> params <- lapply(lapply(as.list(param.names), function(name, object) eval(substitute(object$NAME_ARG
> names(params) <- param.names
> tracefxn <- function(x, name)  plot(x, plot.type="single", col=1:ncol(x), ylab=name)
> mapply(tracefxn, params, name=names(params))
```

6 Posterior probabilities of differential expression

We refer to the posterior mean of the standardized offsets in the hierarchical model as the Bayesian effect size (BES). The BES is calculated as $\frac{\delta_g \Delta_{gp}}{c\tau\sigma^{b_p}}$ and obtained by

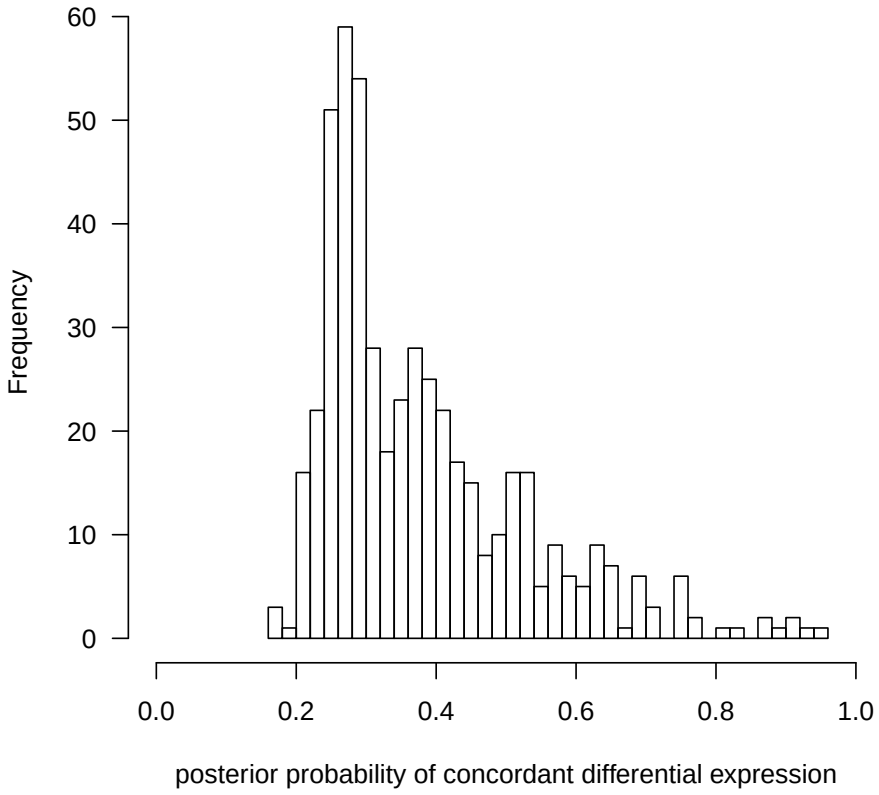
```
> bayesianEffectSize(xmcmc) <- calculateBayesianEffectSize(xmcmc)
```

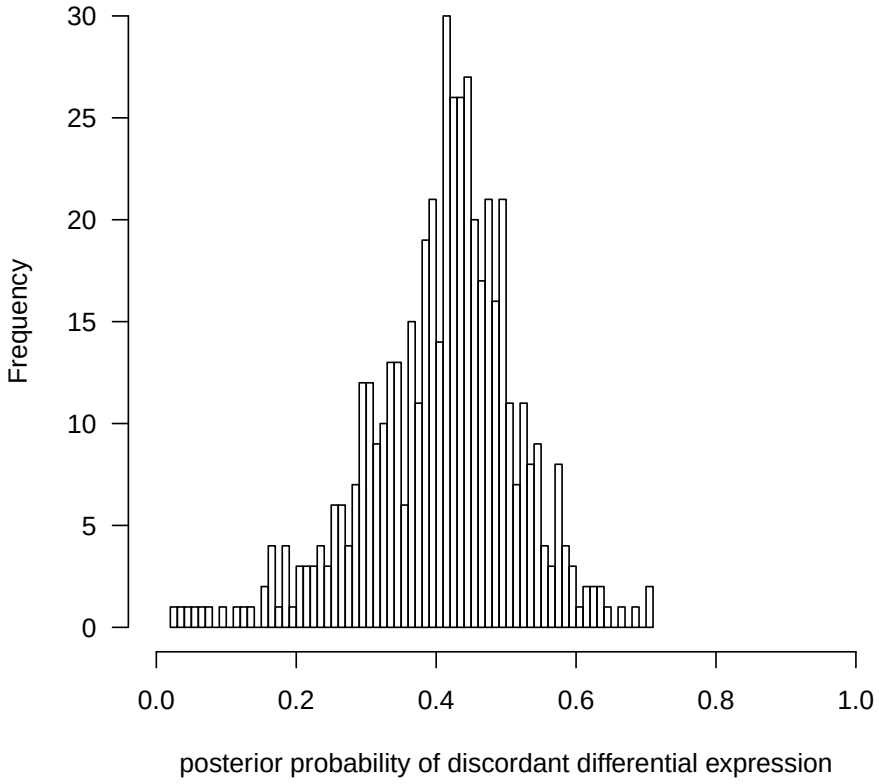
As the function `calculateBayesianEffectSize` requires that the δ, Δ, and σ^2 chains are saved, the above code chunk is not evaluated in this vignette.

```
> load(file.path(pathToLogFiles, "BES.rda"))
> load(file.path(pathToLogFiles, "postAvg.rda"))
```

Posterior averages for the probability of differential expression, concordant differential expression, and discordant differential expression are stored in the `postAvg` object.

See **?** for a discussion of how the above posterior average probabilities are computed.

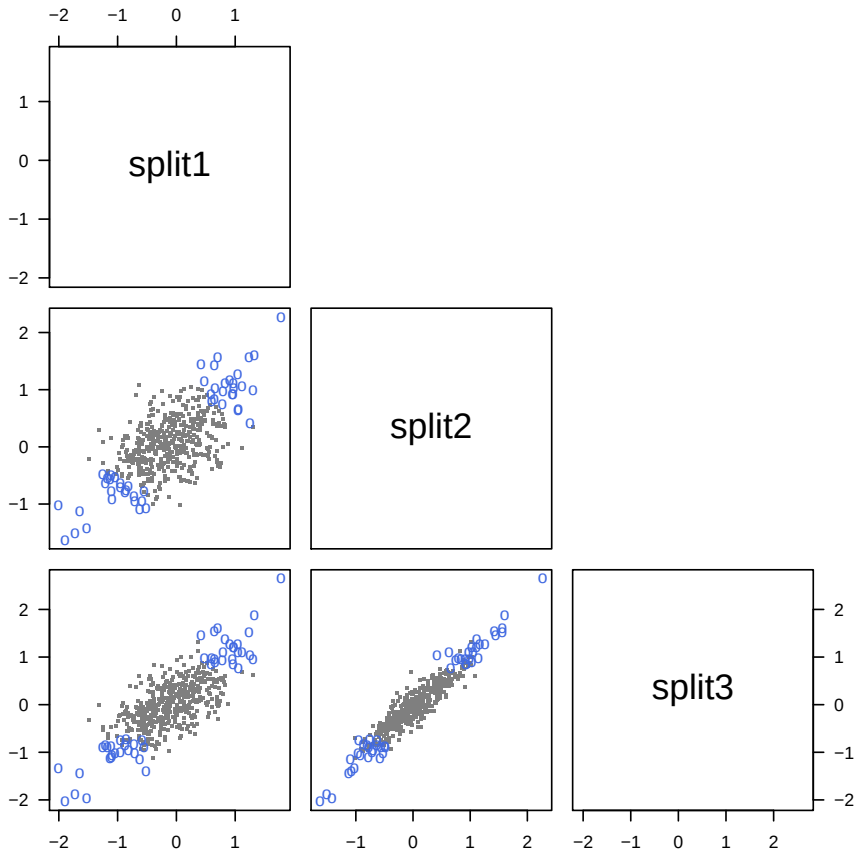

Frequency
60
50
40
30
20
10
0
0.0
0.2
0.4
0.6
0.8
1.0
posterior probability of concordant differential expression



We may wish to differentially label the study-specific statistics of effect size that have high probabilities of concordant differential expression. To do this, we order the matrix of the BES so that the genes with the highest posterior probabilities are plotted last. The function `symbolsInteresting` returns a list of graphical options to `pairs`.

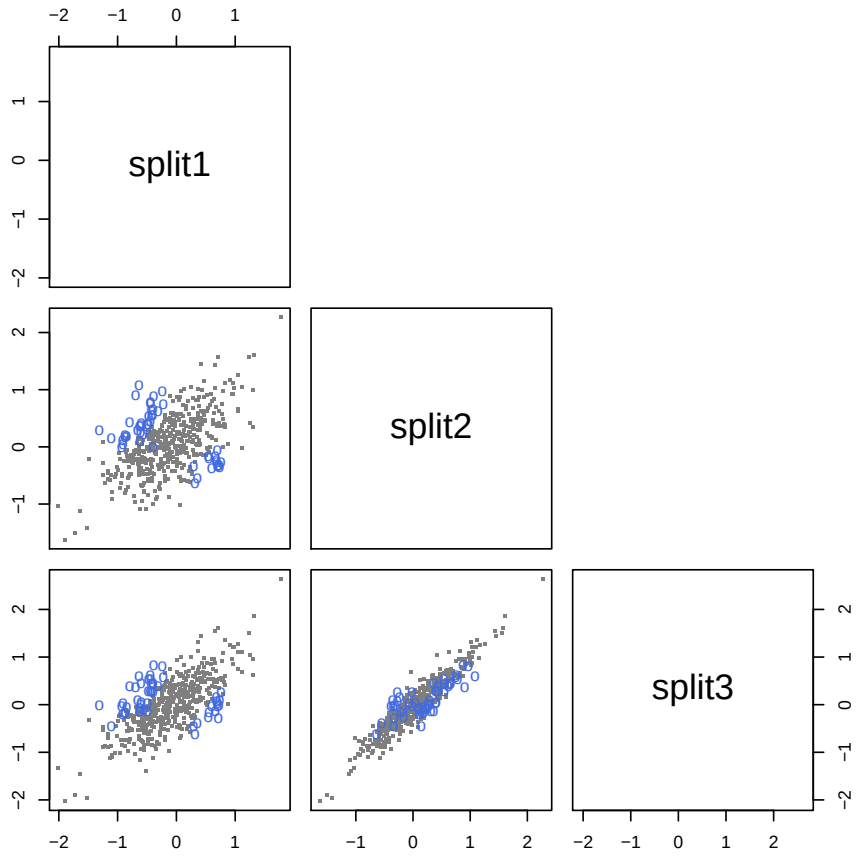
```
> op.conc <- symbolsInteresting(rankingStatistic=postAvg[, "concordant"])
> op.disc <- symbolsInteresting(rankingStatistic=postAvg[, "discordant"])
```

```
> par(las=1)
> graphics:::pairs(BES[op.conc$order, ], pch=op.conc$pch, col=op.conc$col,
+                  bg=op.conc$bg, upper.panel=NULL, cex=op.conc$cex)
```



or high probabilities of discordant differential expression

```
> graphics:::pairs(BES[op.disc$order, ], pch=op.disc$pch, col=op.disc$col, bg=op.disc$bg,
+                  upper.panel=NULL, cex=op.disc$cex)
```



7 Alternative cross-study summaries of differential expression

Study-specific estimates of effect size, such as SAM or t-statistics, can be useful to check the overall reproducibility between studies. Again using pairwise scatterplots we plot t- and SAM-statistics using different colors and plotting symbols for the genes that show high posterior probabilities of concordant differential expression.

```
> ##t <- ssStatistic(statistic="t", phenotypeLabel="adenoVsquamous", esetList=xlist)
> tt <- rowttests(xlist, "adenoVsquamous", tstatOnly=TRUE)
> if(require(siggenes)){
+   sam <- ssStatistic(statistic="sam", phenotypeLabel="adenoVsquamous", esetList=xlist)
+ }
> if(require(GeneMeta)){
+   z <- ssStatistic(statistic="z", phenotypeLabel="adenoVsquamous", esetList=xlist)
+ }
```

```
> graphics:::pairs(tt[op.conc$order, ], pch=op.conc$pch, col=op.conc$col,
```

```
+                   bg=op.conc$bg, upper.panel=NULL, cex=op.conc$cex)
>
```

```
> graphics:::pairs(tt[op.conc$order, ], pch=op.conc$pch, col=op.conc$col,
+                   bg=op.conc$bg, upper.panel=NULL, cex=op.conc$cex)
>
```

```
> graphics:::pairs(tt[op.disc$order, ], pch=op.disc$pch, col=op.disc$col,
+                   bg=op.disc$bg, upper.panel=NULL, cex=op.disc$cex)
```

Uncorrelated t and SAM statistics suggest a low level of reproducibility that may be attributable to technological differences in the platforms, probes that align to different transcripts of the same gene, or differences in the study populations. Low or non-existing reproducibility may induce a wrong borrowing of strength in the Bayesian model, whereby concordant differential expression is seen as noise and shrunk to zero. Hence, study-specific estimates of t- and SAM-statistics may be helpful in deciding whether the Bayesian model is likely to be beneficial.

If the correlation across studies is low, we suggest an unsupervised approach to gene filtering, such as integrative correlation, to select for genes that show some level of reproducibility across studies. The R packages *MergeMaid* and *genefilter* may be helpful.

For evaluating the overall differential expression, we follow the discussion of Garrett-Mayer (**?**), and combine the elements of single study statistics in a linear fashion to obtain a statistic suitable for assessing differential expression. For a more detailed discussion of how these cross-study summaries were generated for evaluating concordant and discordant differential expression, see **?**.

```
> tScores <- xsScores(tt, N=nSamples(xlist))
> samScores <- xsScores(sam, nSamples(xlist))
> zScores <- xsScores(z[, match(names(xlist), colnames(z))], N=nSamples(xlist))
> ##Concordant differential expression, we use the combined score from the random effects model direct
> zScores[, "concordant"] <- z[, "zSco"]
```

8 Session Information

The version number of R and packages loaded for generating the vignette were:

- R version 3.3.1 (2016-06-21), `x86_64-apple-darwin13.4.0`
- Locale: `C/en_US.UTF-8/en_US.UTF-8/C/en_US.UTF-8/en_US.UTF-8`
- Base packages: base, datasets, grDevices, graphics, methods, parallel, stats, utils
- Other packages: Biobase 2.34.0, BiocGenerics 0.20.0, XDE 2.20.0
- Loaded via a namespace (and not attached): AnnotationDbi 1.36.0, DBI 0.5-1, IRanges 2.8.0, Matrix 1.2-7.1, MergeMaid 2.46.0, RCurl 1.95-4.8, RSQLite 1.0.0, S4Vectors 0.12.0, XML 3.98-1.4, annotate 1.52.0, bitops 1.0-6, genefilter 1.56.0, grid 3.3.1, gtools 3.5.0, lattice 0.20-34, mvtnorm 1.0-5, splines 3.3.1, stats4 3.3.1, survival 2.39-5, tools 3.3.1, xtable 1.8-2