Description of metahdep: an R package for hierarchical dependence in meta-analysis

John R. Stevens[1] and Gabriel Nicholas[2]

April 22, 2010

1. Assistant Professor of Statistics, Department of Mathematics and Statistics, and Center for Integrated Biosystems, Utah State University (`http://www.stat.usu.edu/~jrstevens`)
2. MS Biostatistician, Department of Biostatistics and Medical Informatics, University of Wisconsin at Madison

Contents

1 Introduction

The *metahdep* package provides tools to conduct a meta-analysis of effect size estimates, particularly when hierarchical dependence is present (Stevens and Taylor, 2009). The package includes convenient commands for the meta-analysis of gene expression studies. We assume the reader is familiar with meta-analysis of effect size estimates, and here adopt the framework used in Stevens and Taylor (2009). This package vignette provides a guide for the use of these tools, using two examples (Sections 2 and 3). The first example, in Section 2, shows the main utility of the *metahdep* package for the meta-analysis of gene

expression studies, which motivated the development of this package (to save computation time). Some familiarity with gene expression technology will help with this first example, but is not necessary for the use of the tools in this package, as demonstrated by the second example, in Section 3, which involves foreign language learning.

Note: If you use this package for gene expression studies, please cite Stevens and Nicholas (2009). If you use this package for general applications, please cite Stevens and Taylor (2009).

2 Meta-Analysis of Gene Expression Studies

The tools in the *metahdep* package can be used to systematically combine the results of multiple gene expression studies, accounting for both sampling and hierarchical dependence among studies, and accounting for fundamental differences between studies (Stevens and Nicholas, 2009). Sampling dependence occurs when two (or more) effect size estimates are based on the same sample of data. Hierarchical dependence occurs when two (or more) studies are conducted by the same lab or research team. The *metahdep* package is specifically designed for cases where each study reports for each gene an effect size estimate of differential expression between the same two conditions. In the example that follows, we describe a hypothetical group of similar gene expression studies, and refer to a "Tissue" difference among studies. In reality this could be any covariate describing differences among studies, and multiple covariates are possible.

For purposes of illustration in this vignette, we suppose that four separate studies have been conducted, each using microarrays to look for genes exhibiting differential expression from the same control condition to the same treatment condition. Study 1 conducted experiments using two tissues (Tissue 0 and Tissue 1), testing for differential expression (control vs. treatment) in each tissue, using the hgu133a Affymetrix GeneChip. Study 2 was conducted by the same lab as Study 1 (call it Lab 1), and tested for differential expression (control vs. treatment) in Tissue 0, using the hgu95a Affymetrix GeneChip. Study 3 was conducted by another lab (call it Lab 2), testing for differential expression (control vs. treatment) in Tissue 0, using the hgu95av2 Affymetrix GeneChip. Finally, Study 4 was conducted by Lab 3, testing for differential expression (control vs. treatment) in Tissue 1 using the hgu133b Affymetrix GeneChip. Table 1 summarizes the five comparisons or tests of differential expression conducted by these three labs.

Before the results of these studies can be passed to the meta-analysis functions of the *metahdep* package, the data must first be formatted. As this formatting is nontrivial, Subsection 2.1 below illustrates the necessary steps. Subsection 2.2 below shows how to pass these formatted results to the main meta-analysis functions of the *metahdep* package.

Study	Comparison	Lab	Tissue	Chip	# Control Arrays	# Treatment Arrays
1	1	1	0	hgu133a	5	5
1	2	1	1	hgu133a	5	5
2	3	1	0	hgu95a	3	3
3	4	2	0	hgu95av2	2	2
4	5	3	1	hgu133b	9	11

Table 1: Summary of sample gene expression studies that are candidates for a meta-analyis, as each experiment tested for differential expression between the same control and treatment conditions.

2.1 Formatting the Gene Expression Data for Meta-Analysis

The first object we create to aid in formatting the data in preparation for meta-analysis is a data frame linking the probeset identifiers for the same gene on the different chip versions. This data frame is meant to help "resolve the many-to-many relationships between probes and genes" (Ramasamy et al. 2008). The data frame must have columns named "chip", "old.name", and "new.name." Probesets sharing a common "new.name" will be combined in the meta-analysis, even if they are on the same array. Probesets not appearing in the data frame will not be used in the meta-analysis.

In this vignette example, Affymetrix GeneChip versions hgu133a, hgu133b, hgu95a, and hgu95av2 are used. Various approaches are possible to match probesets across array versions based on common gene content; see Ramasamy et al. (2008) for a discussion. The optimal resolution of this common gene content issue is beyond the scope of this *metahdep* package. The code chunk below uses Entrez Gene IDs as probesets' "new.name" and creates a data frame in the necessary format, discarding probesets that do not map to an Entrez Gene ID. As with other possible approaches, this Entrez Gene ID mapping can allow multiple probesets to share the same "new.name" on the same array version. Package users may choose to keep only one probeset for each "new.name" per array version. This choice could be made based on sequence or biological information, or it could be based on experimental data, as suggested in Ramasamy et al. (2008); for example, the `findLargest` function in the *genefilter* package could be used to identify the probesets with the largest effect size for each Entrez Gene ID. Similarly, Rhodes et al. (2002) chose the probesets with the smallest P-values for each gene ID. Even if multiple probesets are allowed to map to the same "new.name" on an array version, however, their effect size estimates can still be treated as hierarchically dependent by the *metahdep* package.

Whichever strategy is employed to match gene content across array versions, the final data frame must have the same column names (and meanings) as the sample object created here.

```
> library(AnnotationDbi)
> library(hgu133a.db)
```

```
> library(hgu133b.db)
> library(hgu95a.db)
> library(hgu95av2.db)
> EID.hgu133a <- unlist(mget(mappedkeys(hgu133aENTREZID), hgu133aENTREZID))
> EID.hgu133b <- unlist(mget(mappedkeys(hgu133bENTREZID), hgu133bENTREZID))
> EID.hgu95a <- unlist(mget(mappedkeys(hgu95aENTREZID), hgu95aENTREZID))
> EID.hgu95av2 <- unlist(mget(mappedkeys(hgu95av2ENTREZID), hgu95av2ENTREZID))
> chip <- c(rep("hgu133a", length(EID.hgu133a)), rep("hgu133b",
+     length(EID.hgu133b)), rep("hgu95a", length(EID.hgu95a)),
+     rep("hgu95av2", length(EID.hgu95av2)))
> EID <- c(EID.hgu133a, EID.hgu133b, EID.hgu95a, EID.hgu95av2)
> newnames <- data.frame(chip = chip, old.name = names(EID), new.name = EID)
```

For convenience (specifically to reduce the size of the sample objects created by this vignette), we use only a subset of the genes. In the following code, we randomly select a subset of 1000 Entrez Gene IDs. Note that we also identify the subsets of corresponding probeset ID's on the different array versions used among our studies of interest.

```
> set.seed(1234)
> use.new.name <- sample(unique(newnames$new.name), 1000)
> use.hgu133a.gn <- names(EID.hgu133a[is.element(EID.hgu133a, use.new.name)])
> use.hgu133b.gn <- names(EID.hgu133b[is.element(EID.hgu133b, use.new.name)])
> use.hgu95a.gn <- names(EID.hgu95a[is.element(EID.hgu95a, use.new.name)])
> use.hgu95av2.gn <- names(EID.hgu95av2[is.element(EID.hgu95av2,
+     use.new.name)])
> HGU.newnames <- newnames[is.element(EID, use.new.name), ]
```

The resulting `HGU.newnames` data frame is equivalent to the `HGU.newnames` data frame that is provided as sample data with the *metahdep* package.

Next, an effect size estimate (and associated variance estimate) for each probeset in each study is needed. There are a number of possible ways to calculate a meaningful effect size estimate for gene expression data; see Ramasamy et al. (2008) for a brief discussion. We adopt the estimate proposed by Hu et al. (2006), and based on the probe-level model test statistic of Bolstad (2004). This model is fit using the *affyPLM* package for Bioconductor; see the Appendix of this vignette for a summary, including the estimation of the variance and covariance of effect size estimates.

It is necessary to estimate the covariance between effect size estimates in cases such as Study 1, where two treatment vs. control comparisons are made, one for each tissue, using data from the same study. The Appendix of this vignette summarizes this covariance estimation in the probe-level model context. This type of sampling dependence can be accounted for in the meta-analysis, as discussed in Stevens and Taylor (2009).

The effect size estimates from this probe-level model, as well as the appropriate variances and covariances, are calculated by the `getPLM.es` function. Other effect size estimates are possible, but in order to use the meta-analysis functions of the *metahdep* package, the results must be in the same format as that returned by the `getPLM.es` function. The result of a call to `getPLM.es` for each study is an object of class `ES.obj`; see the object class help page (`?"ES.obj-class"`) for more on this class.

In the sample code that follows, we use gene expression data freely available through the *ALLMLL* and *ALL* packages. The experimental design we describe does not match the actual design of the experiments used to generate these data. However, our goal here is to demonstrate the use of the meta-analysis tools in the *metahdep* package, and these publicly available data are convenient for this demonstration.

For Study 1, we have 20 arrays (from the `MLL.A` data in the *ALLMLL* package). We assume that the 20 arrays in this *AffyBatch* object are in the following order: 5 Tissue 0 Controls, 5 Tissue 1 Controls, 5 Tissue 0 Treatments, and then 5 Tissue 1 Treatments. Based on this assumed structure, we create the `ES.obj` for Study 1. Note that for this study, as well as for the remaining studies, we restrict our attention to the convenient subset of probe set ID's identified previously for each chip version. The two comparisons of interest (treatment vs. control for Tissue 0, and treatment vs. control for Tissue 1) are defined by the pairs of array index vectors passed to the `trt1` and `trt2` arguments. The covariate differences of these two comparisons are represented in theh `covariates` argument. Because this study was conducted by Lab 1 (see Table 1), we define the hierarchical dependence group with the `dep.grp=1` argument:

```
> library(ALLMLL)
> data(MLL.A)
> ES.exp1 <- getPLM.es(abatch = MLL.A, trt1 = list(c(1:5), c(6:10)),
+     trt2 = list(c(11:15), c(16:20)), sub.gn = use.hgu133a.gn,
+     covariates = data.frame(Tissue = c(0, 1)), dep.grp = 1)
```

For Study 2, we have 6 arrays (from the `spikein95` data in the *SpikeInSubset* package). We assume that the first 3 arrays are Tissue 0 Controls, and the last 3 are Tissue 0 Treatments. Based on this assumed structure, we create the `ES.obj` for Study 2. Unlike Study 1, here there is only one comparison of interest, so the `trt1` and `trt2` arguments have only one element each, and the `covariates` data.frame argument has only one row. This study was conducted by the same Lab 1 as Study 1, and this hierarchical dependence is indicated by the `dep.grp` argument:

```
> library(SpikeInSubset)
> data(spikein95)
> ES.exp2 <- getPLM.es(abatch = spikein95, trt1 = 1:3, trt2 = 4:6,
+     sub.gn = use.hgu95a.gn, covariates = data.frame(Tissue = 0),
+     dep.grp = 1)
```

For Study 3, we have 4 arrays (from the `Dilution` data in the *affy* package). We assume that the first 2 arrays are Tissue 0 Controls, and the last 2 are Tissue 0 Treatments. Based on this assumed structure, we create the `ES.obj` for Study 3:

```
> data(Dilution)
> ES.exp3 <- getPLM.es(abatch = Dilution, trt1 = 1:2, trt2 = 3:4,
+     sub.gn = use.hgu95av2.gn, covariates = data.frame(Tissue = 0),
+     dep.grp = 2)
```

Finally, for Study 4, we have 20 arrays (from the `MLL.B` data in the *ALLMLL* package). We assume that the first 9 arrays are Tissue 1 Controls, followed by 11 Tissue 1 Treatments. Based on this assumed structure, we create the `ES.obj` for Study 4:

```
> data(MLL.B)
> ES.exp4 <- getPLM.es(abatch = MLL.B, trt1 = 1:9, trt2 = 10:20,
+     sub.gn = use.hgu133b.gn, covariates = data.frame(Tissue = 1),
+     dep.grp = 3)
```

With the `ES.obj`'s thus created for each study, we now put them together as elements in a list, in preparation for passing the list to the `metahdep.format` function of the *metahdep* package. The list created by the following code is equivalent to the `HGU.DifExp.list` list that is provided as sample data with the *metahdep* package:

```
> HGU.DifExp.list <- list(ES.exp1, ES.exp2, ES.exp3, ES.exp4)
```

With the list of `ES.obj`'s and the "new names" `data.frame` thus created, the `metahdep.format` function can be used to format the data in final preparation for meta-analysis. It is important to note that the same covariate(s) must be defined in each of the `ES.obj`'s to be used, even if they are constants in some `ES.obj`'s (as Tissue is in each of Studies 2, 3, and 4 here). Also, if any of the `ES.obj`'s do not have a value of `dep.grp` defined, then the `metahdep.format` function will assume that the studies (or `ES.obj` objects) are hierarchically independent, and a cautionary note will be displayed. The result of the following code is equivalent to the `HGU.prep.list` object that is provided as sample data with the *metahdep* package:

```
> HGU.prep.list <- metahdep.format(HGU.DifExp.list, HGU.newnames)
```

This resulting `HGU.prep.list` object is a list. Each element of the list is a `metaprep` object corresponding to a specific "new name" and containing the corresponding effect size estimates, covariance matrix, covariate matrix, and hierarchical dependence structure. See the object class help page (`?"metaprep-class"`) for more on this class. Section 2.2 below demonstrates how this list of `metaprep` objects is passed to the meta-analysis functions of the *metahdep* package.

2.2 Meta-Analysis of Formatted Gene Expression Studies

With a list of `metaprep` objects as returned by the `metahdep.format` function, the fixed effects, random effects, and hierarchical Bayes models can be applied to each "new name" gene individually using the `metahdep` function. In the following code, sampling dependence is accounted for in each of the models, and hierarchical dependence is accounted for in the two "`.ds`" objects returned when `delta.split=TRUE`. The non-intercept columns of the covariate matrix are centered about their means (`center.X=TRUE`) so that the Intercept term is of primary interest as the population mean effect size.

First load the package and necessary data objects (all created above). We can also select a subset of genes on which to perform the meta-analysis. In practice, biological interest would identify the subset, but we use a random subset of 50 just for convenience in our demonstration here:

```
> library(metahdep)
> data(HGU.newnames)
> data(HGU.prep.list)
> set.seed(123)
> gene.subset <- sample(HGU.newnames$new.name, 50)
```

Run fixed effects meta-analysis on each gene:

```
> FEMA <- metahdep(HGU.prep.list, method = "FEMA", genelist = gene.subset,
+     center.X = TRUE)
```

Run random effects meta-analysis on each gene, first ignoring hierarchical dependence (`delta.split=FALSE`) and then accounting for hierarchical dependence (`delta.split=TRUE`):

```
> REMA <- metahdep(HGU.prep.list, method = "REMA", genelist = gene.subset,
+     delta.split = FALSE, center.X = TRUE)
> REMA.ds <- metahdep(HGU.prep.list, method = "REMA", genelist = gene.subset,
+     delta.split = TRUE, center.X = TRUE)
```

Run hierarchical Bayes meta-analysis on each gene, first ignoring hierarchical dependence (`delta.split=FALSE`) and then accounting for hierarchical dependence (`delta.split=TRUE`):

```
> HBLM <- metahdep(HGU.prep.list, method = "HBLM", genelist = gene.subset,
+     delta.split = FALSE, center.X = TRUE, two.sided = TRUE)
> HBLM.ds <- metahdep(HGU.prep.list, method = "HBLM", genelist = gene.subset,
+     delta.split = TRUE, center.X = TRUE, two.sided = TRUE)
```

In these hierarchical Bayes meta-analysis models, we use `two.sided=TRUE` so that the posterior probabilities are converted to two-sided p-values, for more traditional interpretation, as in Stevens and Taylor (2009).

Each call to the `metahdep` function returns a `data.frame` object with summaries of the meta-analysis model called, with a row for each "new name" gene. The final chunk of code below compares the results from these five models, focusing on the test of significance of the population mean effect size (i.e., the Intercept p-value). The resulting scatterplot matrix appears in Figure 1.

```
> pvals <- data.frame(FEMA = FEMA$Intercept.pval, REMA = REMA$Intercept.pval,
+     REMA.ds = REMA.ds$Intercept.pval, HBLM = HBLM$Intercept.pval,
+     HBLM.ds = HBLM.ds$Intercept.pval)
> plot(pvals, main = "Comparison of Intercept P-values from Meta-Analysis",
+     pch = 16, cex = 0.5)
```

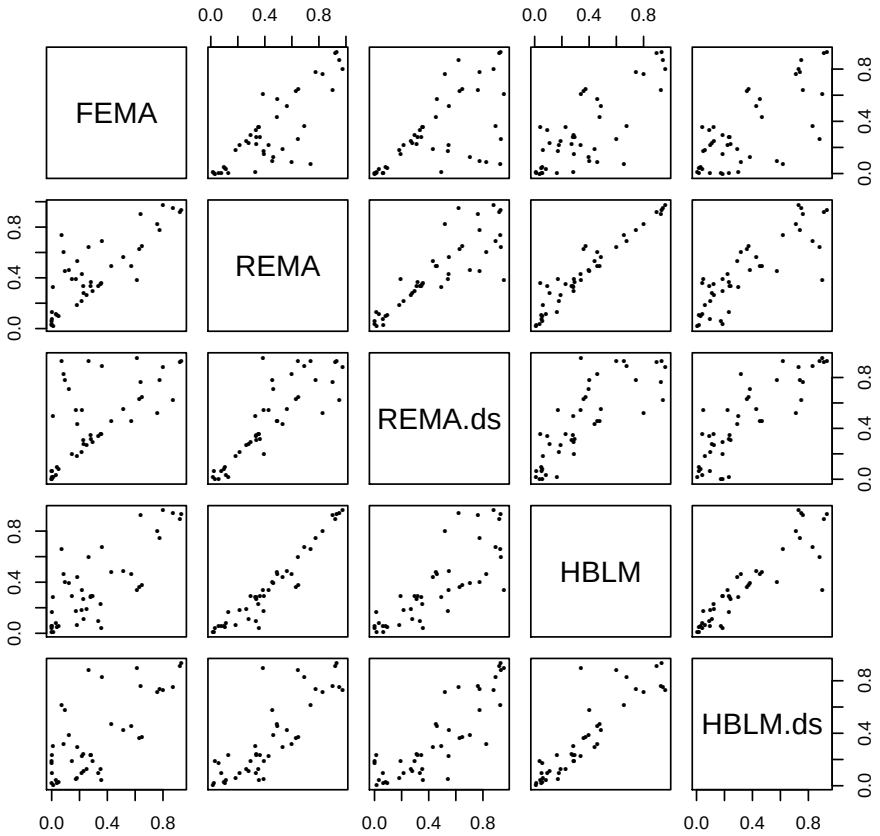

Figure 1: A comparison of the Intercept p-values (testing the significance of the population mean effect size) from several meta-analysis models applied to the gene expression data.

	n2	n1	y2	y1	s2	s1	Rec	Time	Part	Pct	dfE
Aweiss	24	24	43.50	30.00	21.47	19.42	1	2	1	2	46
Baumann #1	6	6	37.17	33.50	20.54	20.54	1	2	1	1	33
Baumann #2	7	7	61.29	51.86	20.54	20.54	1	2	2	1	33
Baumann #3	8	7	25.75	18.29	21.83	21.83	1	2	1	1	38
Baumann #4	7	7	57.71	78.00	21.83	21.83	1	2	2	1	38
Davis	23	26	28.22	11.10	8.69	9.00	1	1	2	1	47
Goyette	12	12	43.00	25.20	14.00	13.10	1	2	3	2	22
Jacobs	58	58	18.16	13.70	5.68	5.44	1	2	2	1	114
Jacobs et al.	33	27	17.30	16.40	9.00	7.30	1	1	2	1	58
Joyce #1	12	11	9.50	8.60	7.59	7.59	1	1	1	1	83
Joyce #2	17	18	14.47	10.89	7.59	7.59	1	1	1	1	83
Joyce #3	13	18	13.10	15.10	7.59	7.59	1	1	2	1	83
Knight	54	51	74.01	56.65	27.29	23.35	1	2	2	2	103
Ko	64	63	13.05	12.86	3.95	3.21	2	1	3	1	125
Kwong-Hung	55	60	6.15	5.97	1.99	1.77	2	2	3	1	113
Lou	16	17	14.63	6.24	8.85	5.14	1	1	2	1	31
Palmer	48	47	21.54	11.78	7.70	6.22	1	1	3	1	93
Stoehr	33	29	15.33	8.03	5.31	4.26	1	2	2	2	60

Table 2: Summary of the glossing data.

3 Meta-Analysis of Foreign Language Learning Studies

The statistical methods used by the *metahdep* package are described in Stevens and Taylor (2009), which focused on a meta-analytic review of 13 experiments with 18 study reports all involving the effect of native-language (L1) vocabulary aids (e.g., glossing) on second language (L2) reading comprehension. The data from that paper are included in the *metahdep* package for purposes of demonstration. The sample code in this section of the package vignette reproduces the main results of Stevens and Taylor (2009).

A summary of the glossing data is provided in the `gloss.Table1` data frame within the `gloss` data set (see output in Table 2).

```
> library(metahdep)
> data(gloss)
> gloss.Table1
```

Based on this summary, the vector of unbiased effect size estimates can be constructed. The following code produces this vector (`theta`), which is equivalent to the vector `gloss.theta` within the `gloss` data set:

```
> attach(gloss.Table1)
> sp <- sqrt(((n1 - 1) * s1^2 + (n2 - 1) * s2^2)/(n1 + n2 - 2))
> c.correct <- (1 - 3/(4 * dfE - 1))
> theta <- c.correct * (y2 - y1)/sp
```

The covariance matrix corresponding to the vector of unbiased effect size estimates is constructed next, with some off-diagonal elements nonzero because of the sampling dependence (as described in Stevens and Taylor, 2009). First get the diagonal elements of the matrix:

```
> pE <- c.correct^2 * dfE/(dfE - 2)
> var.theta <- pE * (1/n2 + 1/n1) + (pE - 1) * theta^2
> V <- diag(var.theta)
```

As described in Stevens and Taylor (2009), sampling dependence exists among effect size estimates in both the Baumann and Joyce studies. As currently ordered, sampling dependence exists between studies 2 and 3 (Baumann), between studies 4 and 5 (Baumann), and among studies 10-12 (Joyce). The off-diagonal elements of the covariance matrix can be constructed accordingly. This resulting matrix `V` is equivalent to the matrix `gloss.V` within the `gloss` data set.

```
> V[2, 3] <- V[3, 2] <- (pE[2] - 1) * theta[2] * theta[3]
> V[4, 5] <- V[5, 4] <- (pE[4] - 1) * theta[4] * theta[5]
> V[10, 11] <- V[11, 10] <- (pE[10] - 1) * theta[10] * theta[11]
> V[10, 12] <- V[12, 10] <- (pE[10] - 1) * theta[10] * theta[12]
> V[11, 12] <- V[12, 11] <- (pE[11] - 1) * theta[11] * theta[12]
```

To account for fundamental differences between studies in the meta-analysis, the design matrix can be constructed. This is equivalent to the matrix `gloss.X` within the `gloss` data set.

```
> X <- matrix(0, nrow = 18, ncol = 6)
> X[, 1] <- 1
> X[Rec == 2, 2] <- 1
> X[Time == 2, 3] <- 1
> X[Part == 2, 4] <- 1
> X[Part == 3, 5] <- 1
> X[Pct == 2, 6] <- 1
> colnames(X) <- c("Intercept", "Rec2", "Time2", "Part2", "Part3",
+     "Pct2")
```

With the vector of unbiased effect size estimates (`theta`), the corresponding covariance matrix (`V`), and the design matrix (`X`) now defined, three classes of meta-analysis models may be considered: fixed effects, random effects, and hierarchical Bayes. The *metahdep* package has defined the `metahdep.FEMA`, `metahdep.REMA`, and `metahdep.HBLM` functions for these three model classes, respectively. In each model, three dependence structures are considered: independence (V diagonal), dependence (V not necessarily diagonal), and delta-split (also accounting for hierarchical dependence). The delta-split structure is not possible in the fixed effects model. For the delta-splitting models, dependence groups must be defined. In these data, the Baumann studies (numbered 2-5 in the current format) are one group, and the Joyce studies (numbered 10-12 in the current format) are another. The following commands define these dependence groups and then proceed to fit these possible models.

```
> dep.groups <- list(c(2, 3, 4, 5), c(10, 11, 12))
> FEMA.indep <- metahdep.FEMA(theta, diag(diag(V)), X, center.X = TRUE)
> FEMA.dep <- metahdep.FEMA(theta, V, X, center.X = TRUE)
> REMA.indep <- metahdep.REMA(theta, diag(diag(V)), X, center.X = TRUE)
> REMA.dep <- metahdep.REMA(theta, V, X, center.X = TRUE)
> REMA.ds <- metahdep.REMA(theta, V, X, center.X = TRUE, delta.split = TRUE,
+     dep.groups = dep.groups)
> HBLM.indep <- metahdep.HBLM(theta, diag(diag(V)), X, center.X = TRUE,
+     two.sided = TRUE)
> HBLM.dep <- metahdep.HBLM(theta, V, X, center.X = TRUE, two.sided = TRUE)
> HBLM.ds <- metahdep.HBLM(theta, V, X, center.X = TRUE, two.sided = TRUE,
+     delta.split = TRUE, dep.groups = dep.groups, n = 20, m = 20)
```

The `metahdep.HBLM` function may return slightly different results for different values of the `n` and `m` arguments, as these control the step sizes in the necessary numerical integrations of the hierarchical Bayes model.

The results from these various meta-analysis models can be summarized. The following code produces the output shown in Table 3; this is a reproduction of Table 4 of Stevens and Taylor (2009).

```
> r1 <- round(c(FEMA.indep$beta.hats[1], FEMA.indep$beta.hat.p.values[1],
+     NA, NA), 4)
> r2 <- round(c(FEMA.dep$beta.hats[1], FEMA.dep$beta.hat.p.values[1],
+     NA, NA), 4)
> r3 <- round(c(REMA.indep$beta.hats[1], REMA.indep$beta.hat.p.values[1],
+     REMA.indep$tau2.hat, NA), 4)
> r4 <- round(c(REMA.dep$beta.hats[1], REMA.dep$beta.hat.p.values[1],
+     REMA.dep$tau2.hat, NA), 4)
> r5 <- round(c(REMA.ds$beta.hats[1], REMA.ds$beta.hat.p.values[1],
```

```
   Model   Structure Intercept      P tau-square varsigma
1  Fixed Independent    0.6166      0       <NA>     <NA>
2  Fixed   Dependent    0.6154      0       <NA>     <NA>
3 Random Independent     0.577 0.0017     0.2432     <NA>
4 Random   Dependent    0.5764 0.0017      0.243     <NA>
5 Random Delta-split    0.5261 0.0026     0.2936  -0.0816
6  Bayes Independent    0.5775 0.0019     0.3057     <NA>
7  Bayes   Dependent    0.5769 0.0019     0.3054     <NA>
8  Bayes Delta-split     0.628 0.0011     0.2573   0.1117
```

Table 3: Summary of results from various meta-analysis models of the glossing data.

```
+     REMA.ds$tau2.hat, REMA.ds$varsigma.hat), 4)
> r6 <- round(c(HBLM.indep$beta.hats[1], HBLM.indep$beta.hat.p.values[1],
+     HBLM.indep$tau.var + HBLM.indep$tau.hat^2, NA), 4)
> r7 <- round(c(HBLM.dep$beta.hats[1], HBLM.dep$beta.hat.p.values[1],
+     HBLM.dep$tau.var + HBLM.dep$tau.hat^2, NA), 4)
> r8 <- round(c(HBLM.ds$beta.hats[1], HBLM.ds$beta.hat.p.values[1],
+     HBLM.ds$tau.var + HBLM.ds$tau.hat^2, HBLM.ds$varsigma.hat),
+     4)
> model <- c(rep("Fixed", 2), rep("Random", 3), rep("Bayes", 3))
> dep <- c("Independent", "Dependent", rep(c("Independent", "Dependent",
+     "Delta-split"), 2))
> Table2 <- as.data.frame(cbind(model, dep, rbind(r1, r2, r3, r4,
+     r5, r6, r7, r8)))
> colnames(Table2) <- c("Model", "Structure", "Intercept", "P",
+     "tau-square", "varsigma")
> rownames(Table2) <- NULL
> Table2
```

References

[1] Bolstad, B.M. (2004), "Low-level Analysis of High-density Oligonucleotide Array Data: Background, Normalization and Summarization," PhD dissertation, U.C. Berkeley.

[2] Hu, P., Greenwood, C.M.T., and Beyene, J. (2006), "Integrative Analysis of Gene Expression Data Including an Assessment of Pathway Enrichment for Predicting Prostate Cancer," *Cancer Informatics* 2006:2 289-300.

[3] Ramasamy, A., Mondry, A., Holmes, C.C., and Altman, D.G. (2008), "Key Issues in Conducting a Meta-Analysis of Gene Expression Microarray Datasets," *PLoS Medicine* 5(9):e184.

[4] Rey, W.J. (1983), *Introduction to Robust and Quasi-Robust Statistical Methods*, Springer-Verlag, Berlin.

[5] Rhodes, D.R., Barrette, T.R., Rubin, M.A., Ghosh, D. and Chinnaiyan, A.M. (2002), "Meta-Analysis of Microarrays: Interstudy Validation of Gene Expression Profiles Reveals Pathway Dysregulation in Prostate Cancer," *Cancer Research* 62:4427-4433.

[6] Stevens, J.R. and Nicholas, G. (2009), "metahdep: Meta-analysis of hierarchically dependent gene expression studies", *Bioinformatics* 25(19):2619-2620.

[7] Stevens, J.R. and Taylor, A.M. (2009), "Hierarchical Dependence in Meta-Analysis," *Journal of Educational and Behavioral Statistics* 34(1):46-73.

Appendix: Summary of Effect Size Estimate Calculation, with Variance and Covariance

For meta-analysis of gene expression data, Hu et al. (2006) demonstrated the strong performance of an effect size estimate based on Bolstad's probe-level model (Bolstad 2004). This model is fit with data from several arrays using the *affyPLM* package for Bioconductor. For each probeset, the Bolstad model returns a vector $\hat{\beta}$ of M-estimates for expression across the arrays, as well as a matrix $\hat{\Sigma}$, the estimated covariance matrix of $\hat{\beta}$. By the central limit theorem, and as in Rey (1983), the asymptotic distribution is

$$\hat{\beta} \sim N(\beta, \Sigma),$$

where β_i is the expression of the gene on array i.

Suppose that for a specific comparison j, a test of differential expression between groups of arrays is to be considered. We assume there are $n_{j,t}$ "treatment" arrays and $n_{j,c}$ "control" arrays. We define c_j to be a contrast vector with elements $1/n_{j,t}$ for "treatment" arrays, $-1/n_{j,c}$ for "control" arrays, and 0 for all other arrays in the study. Then the gene's effect size estimate for this comparison of interest j is

$$\hat{\theta}_j = \sqrt{\frac{1}{n_{j,t}} + \frac{1}{n_{j,c}}} \times \frac{c_j^T \hat{\beta}}{\sqrt{c_j^T \hat{\Sigma} c_j}}.$$

Note that

$$\frac{c_j^T \hat{\beta}}{\sqrt{c_j^T \Sigma c_j}} \sim N\left(\frac{c_j^T \beta}{\sqrt{c_j^T \Sigma c_j}}, 1\right),$$

and so

$$Var\left[\frac{c_j^T \hat{\beta}}{\sqrt{c_j^T \Sigma c_j}}\right] = 1.$$

Another comparison or test of differential expression could be considered between different groups of arrays, such as treatment vs. control for another tissue. Define c_h to be the appropriate contrast vector, constructed similar to c_j. The gene's effect size estimate for this comparison of interest h, $\hat{\theta}_h$, can be calculated similar to $\hat{\theta}_j$. Because $\hat{\theta}_j$ and $\hat{\theta}_h$ are both based on the same estimate $\hat{\beta}$, they are sampling dependent. We can calculate

$$Cov\left[\frac{c_j^T \hat{\beta}}{\sqrt{c_j^T \Sigma c_j}}, \frac{c_h^T \hat{\beta}}{\sqrt{c_h^T \Sigma c_h}}\right] = \frac{c_j^T \Sigma c_h}{\sqrt{c_j^T \Sigma c_j}\sqrt{c_h^T \Sigma c_h}}.$$

With all this, we have estimates for the variance and covariance of the gene's effect size estimates as

$$\begin{aligned} Var\left[\hat{\theta}_j\right] &\approx \frac{1}{n_{j,t}} + \frac{1}{n_{j,c}}, \\ Cov\left[\hat{\theta}_j, \hat{\theta}_h\right] &\approx \sqrt{\frac{\left(\frac{1}{n_{j,t}} + \frac{1}{n_{j,c}}\right)\left(\frac{1}{n_{h,t}} + \frac{1}{n_{h,c}}\right)}{\left(c_j^T \hat{\Sigma} c_j\right)\left(c_h^T \hat{\Sigma} c_h\right)}} \times c_j^T \hat{\Sigma} c_h. \end{aligned}$$