MAMA: an R package for Meta-Analysis of MicroArray

Ivana Ihnatova

April 29, 2013

Contents

Chapter 1

Introduction

This paper provides a user guide to the R-package MAMA. The package implements eight different methods that are designed to identify differentially expressed genes in meta-analysis of microarrays. A more comprehensive vignette can be downloaded from: `http://is.muni.cz/www/184415/MAMA_full.pdf`

In here, we will demonstrate the features of the package with an example of meta-analysis in cancer microarray data, the comparison of expression profiles in MSI (microsatelite instable) and MSS (microsatelite stable) colon cancer. We gathered three microarray data from public databases. The data are stored in `ColonData`. It is an object of new S4 class - `MetaArray`, in which separate slots for gene expression data matrices (expression profiles), clinical sample characteristics and datanames were defined.

We start with package and sample data loading.

```
> rm(list=ls(all=TRUE))
> options(width=60)
> library(MAMA)
> data(ColonData)
> ColonData

Dataset Denmark  containing 500 probes and  77 samples.
Sumarization of samples:
MSIstatus
---------
MSI MSS
 39  38
age
---
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  36.00   45.00   54.00   54.92   62.00   80.00

Dataset Australia  containing 500 probes and  36 samples.
Sumarization of samples:
MSIstatus
---------
MSI MSS
  5  31
```

```
age
---
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  50.00   57.75   66.00   66.36   73.75   85.00

Dataset Japan  containing 500 probes and  41 samples.
Sumarization of samples:
location
--------
  distal proximal  unknown
      28       11        2
MSIstatus
---------
MSI MSS
 16  25
age
---
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  40.00   47.00   53.00   53.61   60.00   65.00
```

The original data sets have been preprocessed and subsampled in order to reduce the computational complexity of the meta-analysis. All data sets have been normalized and are in log_2-scale. The corresponding sample sizes for the three datasets can be seen in outprint above. The same set of 500 gene has been selected in each dataset.

A different method is described in each of the parts below and the descriptions are independed from each other, so you can directly move to the method of your interest. A wrapper function for each method outperforms all necessary steps as one-line command.

We select the genes at significance level of 0.05 in all methods.

Chapter 2

Methods that combine p-values

Introduction

In this part we will focus on methods that combine p-values [4], [5]. These methods are inspired by Fisher's S-statistic published in 1925 [6].

Two measurements of significance of change in gene expression are the most common. These are: value of test-statistic and p-value. In here, the p-values from study-specific analysis are combined into one p-value in sense of sum of logs. Methods differ in test statistic used to calculate the study-specific p-values.

Usage

Function `metaMA` is a wrapper funcion for this method. By

```
> pval<- metaMA(ColonData, "MSIstatus", which ="pval")
```

```
    DE    IDD   Loss    IDR    IRR 
212.00  43.00  22.00  20.28  11.52 
```

we combine study specific p-values(`pval`) in `ColonData` dataset using column `"MSIstatus"` from clinical data as class labels. `pval` is a list with seven slots. *Study1* to *Study3* are numeric vectors with indices of differentially expressed genes in data sets 1 to 3. *AllIndStudies* is a vector of indices of differentially expressed genes in at least one data set. Differentially expressed genes found by meta-analysis have their indices stored in *Meta*. A slot called *TestStatistic* is a vector with test statistics in meta-analysis. The last slot *gene.names* contains the names of the genes in the analysis.

Chapter 3

Methods that combine effect sizes

Introduction

Methods that combine effect size use hierarchical model:

$$y_i = \theta_i + \epsilon_i, \epsilon_i \sim N(0, \sigma_i^2)$$

$$\theta_i = \mu + \delta_i, \delta_i \sim N(0, \tau_i^2),$$

where μ is true difference in mean expression between two classes, y_i denotes the measure effect for study i, with $i = 1, .., k$, τ^2 represents the between study variability, σ_i^2 denotes the within study variability. The analysis is different depending on whether a fixed-effect model (FEM) or a random-effect model (REM) is deemed appropriate. Under a FEM, $\tau = 0$ is assumed, otherwise a REM need to be fit. The estimates of the overall effect μ are different depending on which model is used.

Two papers dealing with effect size combination as method for meta analysis of microarray have been published [4] and [9]. They differ in effect size definition and implementation.

Method presented in [4] offers three variants of effect sizes (classical and moderated T-test) and uses explicitly random-effect model. It is implemented as two functions `EScombination` for unpaired data and `EScombination.paired` for paired data.

On the other hand, in [9] the effect size is defined as Hedge's and Olkin's g and both random-effect and fixed-effect are available. Package *GeneMeta* [10] implements this method.

Algorithm

1. Data recoding.

2. Effect size calculation in each data set.

3. Decision between random-effect model (REM) and fixed-effect model (FEM).

4. Model application.
5. Identification of differentialy expressed genes.

Usage

Because there are two different ways of implementation of this method, we will discuss them separately.

Implementation from metaMA package

```
> es <- metaMA(ColonData, "MSIstatus", which = "ES")
```

```
    DE    IDD   Loss    IDR    IRR
166.00  27.00  52.00  16.27  27.23
```

This object is a list with seven slots. *Study1* to *Study3* are indices of differentially expressed genes in data sets 1 to 3. *AllIndStudies* is a vector of indices of differentially expressed genes in at least one data set. Differentially expressed genes found by meta-analysis have their indices stored in *Meta*. A slot called *TestStatistic* is a vector with test statistics ("combined effect size") in meta-analysis. The last slot *gene.names* contains the names of the genes in the analysis (rownames of the gene expression data matrices).

Implemenetation from GeneMeta package

Function (`ES.GeneMeta`) wrapes data preparation and functions `zScore`, `ScoresFDR`.

```
> es2<-ES.GeneMeta(ColonData, "MSIstatus", nperm = 100)
```

`nperm = 100` is used only for computational complexity. A thousand of permutation is more appropriate. We get a list with two slots. Columns of the first one (`theScores`) refer to:

- *Effect_Ex_* are the unbiased estimates of the effect
- *EffectVar_Ex_* are the estimated variances of the unbiased effects
- *zSco_Ex_* are the unbiased estimates of the effects divided by their standard deviation
- *Qvals* are the Q statistics and *df* is the number of combined experiments minus one
- *MUvals* and *MUsds* are the overall mean effect sizes and their standard deviations
- *zSco* are the z scores
- *Qpvalues* is for each gene the probability that a chi-square distribution with *df* degree of freedom has a higher value than its *Q* statistic
- *Chisq* is the probability that a chi-square distribution with 1 degree of freedom has a higher value than *zSco2*

and the second one (*ScoresFDR*) is a list with three slots: `pos`, `neg` and `two.sided`. The first slot stores the results of the calculation, if the FDR is computed for the positive Z-scores, the second for the negative Z-scores and the last one for the two sided situation. Each slots contains a matrix with similar structure as `theScores` with additional FDR for each data set and combination.

Chapter 4

Similarity of Ordered Gene Lists (SOGL)

Introduction

Similarity of Ordered Gene Lists is another method for meta-analysis of microarray. It is call as ”comparison of comparisons” by its authors [14].

Briefly, it assigns a similarity score to a comparison of ranked (ordered) gene lists. The score is based on the number of overlapping genes in the top ranks. It computes the size of overlap for each rank. The final score is a weighted sum of these values, with more weight put on the top ranks.

Algorithm

1. Required data sets - data sets with same set of genes are required.
2. Ranking of genes - The genes are then ranked based on gene-wise test on difference of class mean. There is only one assumption about test result: a large positive test score corresponds to up-regulation and a large negative value to down-regulation.
3. Computing the overlap - for each rank (from 1 to number of genes) we count the number of genes that appear in both ordered lists up to that position. It is denoted as $O_n(G_A, G_B)$, where G_A and G_B refer to ordered gene lists.
4. Similarity score - First we compute a total overlap A_n at position n given as $O_n(G_A, G_B) + O_n(f(G_A), f(G_B))$, where $f()$ means flipped list (down-regulated genes on top). Later we add weights to it and we sum it up to preliminary score. Weights $w_\alpha = e^{-\alpha.n}$ are used as default and tunning of parameter α is needed. The weights are used to put more importance on top genes.

The algorithm above is valid for meta-analysis in which expression data are not available. However, in this situation we can not use same approach for tunning α.

Usage

Function `performSOGL` calculates the similarity score, its statistical significance and the genes responsible for the score.

```
> SOGL.res <- performSOGL(ColonData, varname = "MSIstatus",
+ test = "FCH", B = 100, which=c("score", "empirical"),
+ min.weight = 1e-05, two.sided = TRUE )
```

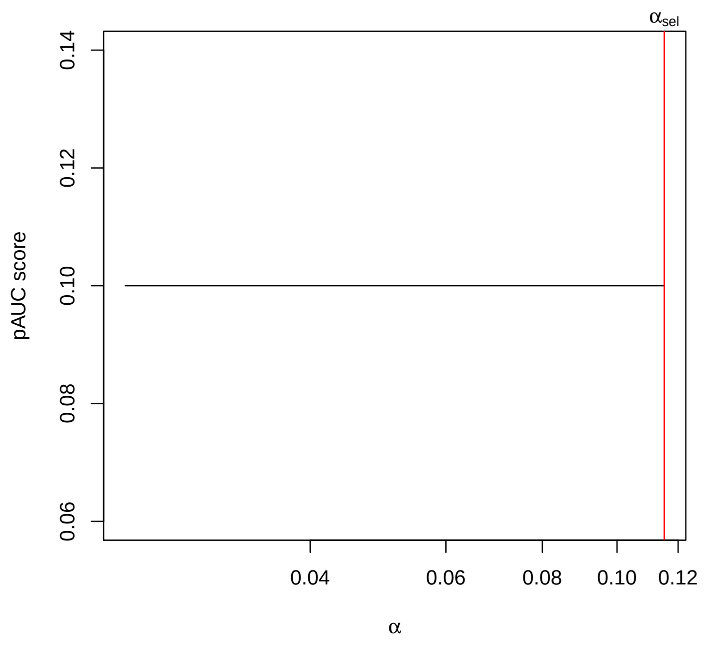
```
Processing data...Tuning alpha..Significance and genes...
```

`SOGL.res` is a list that contains:

- *ordering* ordered gene lists as a data.frame where columns refer to datasets
- *alpha.selected* selected value of alpha parameter
- *alpha.considered* vector of alpha considered for selection
- *pAUC* pAUC values related to all alphas considered
- *random* random scores (permutations of class labels)
- *subsample* scores after subsampling from each class and dataset
- *emp.ci* empirical confidence intervals for number of overlapping genes
- *common.genes* vector of number of overlapping genes
- *score* observed similarity score
- *significance* significance of the observed score in form of p-value
- *genes* genes that account for observed similarity score

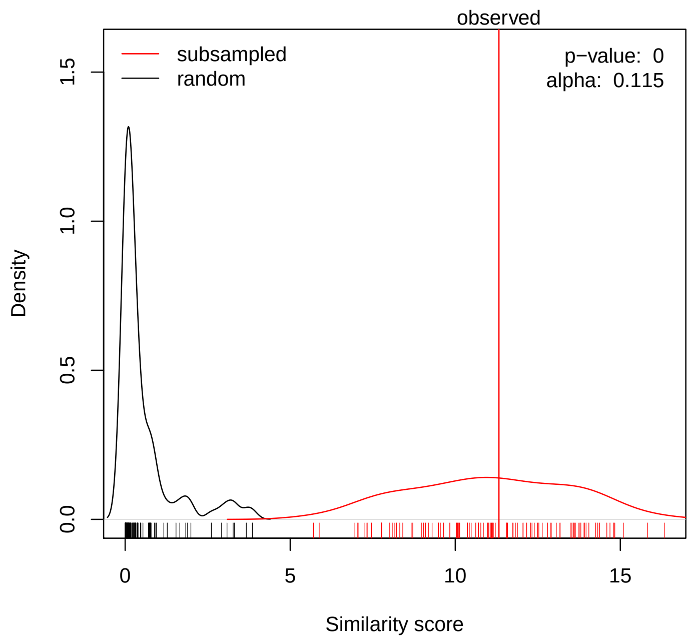
The `SOGL.res` is an object of class `SOGLresult` for which plot function exist.

```
> plot(SOGL.res, "alpha selection")
```



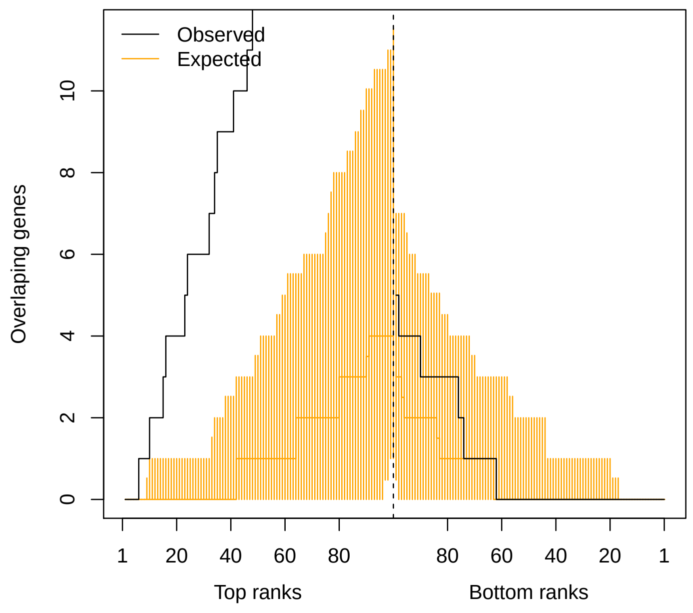
The graph shows pAUC of all α's considered. The selected α is singed by red vertical line.

```
> plot(SOGL.res, "density")
```



This plots the estimated densities of random (in black) and subsampled score (in red) for selected α. The observed similarity score is marked by vertical line. The bottom rugs are scores actually achieved in permutations.

```
> plot(SOGL.res, "empirical CI")
```



The graph displays the empirical confidence intervals for number of overlapping genes to each position. Obsereved overlaps are drawn as black step-function.

Chapter 5

RankProduct

Introduction

RankProduct is a non-parametric statistic that detects up-regulated and down-regulated genes under one condition against another condition. In our sample data set we look for difference in expression between MSI and MSS colon cancer.

It focuses on genes which are consistently highly ranked in a number of lists, for example genes that are regularly found among top up-regulated genes in many microarray studies. It assumes that under the null hypothesis the order of all items is random then the probability of finding a certain item among the top r of n items in a list is $p = r/n$. Rank product is defined by multiplying these probabilities $RP = \prod_i \frac{r_i}{n_i}$, where r_i is the rank of the item in the i-th list and n_i is the total number of the items on i-th list. The smaller the RP value the smaller the probability that the observation of the item at the top of the lists is due to chance. It is equivalent to calculating the geometric mean rank. A list of up- or down-regulated genes are selected based on the estimated percentage of false positive prediction (pfp) - known as false discovery rate (FDR), too.

Algorithm

Algorithm of the method has five steps:

1. Fold-change ratio is calculated in each data set.
2. Ranks are assigned (1 for the highest value) according to fold-change ratio. r_{gi} is rank of gene g in comparison i, where i is from 1 to K, where K is sum of products of number of slides in groups.
3. RankProduct for a gene (RP_g) is calculated as $\prod_i r_{gi}^{1/K}$
4. l permutations of expression values at each microarray slide is performed and all previous steps repeated. We obtain $RP_g^{(l)}$
5. Step 4 is repeated L times to estimate the distribution of $RP_g^{(l)}$. This distribution is used to calculate p-value and pfp for each gene.

Usage

Function `RankProduct` provides the tables of identified up- and down- regulated genes from a MetaArray-class object.

```
> rp <- RankProduct(ColonData, "MSIstatus", num.perm = 50,
+  gene.names = rownames(GEDM(ColonData)[[1]]) )

 The data is from  3 different origins

Rank Product analysis for two-class case

Starting  50 permutations...
Computing pfp...
Table1: Genes called significant under class1 < class2

Table2: Genes called significant under class1 > class2
```

Chapter 6

Z-statistic - posterior mean differential expression

Introduction

The main idea of this method is that one can use data from one study to construct a prior distribution of differential expression and thus utilize the posterior mean differential expression, weighted by variances, whose distribution is standard normal distribution due to classic Bayesian probability calculation.

It is based on assumption that gene expression is normally distributed with mean μ_g and SD σ_g^2 and that we can estimate σ_g^2 by pooling together all genes with similar levels of mean intensity. The difference in gene expression is tested by

$$Z = \frac{D}{\sigma_D} = \frac{\overline{X}_1 - \overline{X}_2}{\sqrt{\frac{\sigma_1^2}{n_1} + \frac{\sigma_2^2}{n_2}}} \sim N(0,1),$$

where $\overline{X}_1$ and $\overline{X}_2$ denotes mean gene expression values in classes, σ_1^2 and σ_2^2 denotes the estimated SD in classes and n_1 and n_2 denotes the number of samples in classes.

Usage

Because the same number of samples in each class and study is used in primary publication of the method [16], we will first look at number of samples in our data.

```
> ColonData

Dataset Denmark  containing 500 probes and  77 samples.
Sumarization of samples:
MSIstatus
---------
MSI MSS
 39  38
age
```

```
---
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  36.00   45.00   54.00   54.92   62.00   80.00

Dataset Australia  containing 500 probes and  36 samples.
Sumarization of samples:
MSIstatus
---------
MSI MSS
  5  31
age
---
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  50.00   57.75   66.00   66.36   73.75   85.00

Dataset Japan  containing 500 probes and  41 samples.
Sumarization of samples:
location
--------
  distal proximal  unknown
      28       11        2
MSIstatus
---------
MSI MSS
 16  25
age
---
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  40.00   47.00   53.00   53.61   60.00   65.00
```

The smallest value in the tables above is 5, therefore we will randomly choose 5 samples in each class and data set. Function `posterior.mean` performs such data reduction and Z-statistic calculation. It has three required arguments: a data set as MetaArray object (`data`), name of clinical parameter with class labels (`varname`) and number of samples to be selected (`nsamp`).

Detecting differentially expressed genes

We apply this method by

```
> z.stat<-posterior.mean(ColonData, "MSIstatus", nsamp=5)
```

```
0 marked as 0
1 marked as 1
Contrast will be 1 - 0
```

Results

```
> head(round(z.stat,3))
```

```
             Zscore Pvalue
1552281_at    5.380  0.000
1552365_at   -6.552  0.000
1552485_at   -3.409  0.001
1552502_s_at -1.056  0.291
1552546_a_at  0.544  0.586
1552553_a_at -1.075  0.282
```

Only values of Z-statistic (`Zscore`) and their p-values (`Pvalue`) are the outputs from this method.

Notes and discussion

This implementation expects either same microarray platform or same scale of expression values (like after POE transformation [17]) in all data sets.

Chapter 7

VennMapping

Introduction

VennMapping [20] is a method based on Venn diagrams and contingency tables. It looks for number of common genes in pairs of gene lists, statistical significance of observed match and returns also names of the common genes.

Algorithm

Algorithm of this method consists of three steps:

1. Calculation of fold-change in each data set.
2. Selection of significant (interesting) genes.
3. Comparison of gene lists pairs.

Usage

Function `VennMapper` selects genes in each dataset according to the fold-change cutoff and compares the lists of the selected genes.

```
> vm <- VennMapper(ColonData, varname="MSIstatus", cutoff=1)
> vm

$conting.tab
          Denmark Australia Japan
Denmark        NA        12    16
Australia      12        NA     7
Japan          16         7    NA

$z.score
           Denmark Australia     Japan
Denmark         NA 14.964793 17.550323
Australia 14.93428        NA  8.098665
Japan     17.56230  8.120747        NA
```

```
$genes
          Denmark
Denmark   NA
Australia "205009_at;206239_s_at;37145_at;205044_at;213385_at;228030_at;205242_at;204818_a
Japan     "202803_s_at;230964_at;213915_at;206239_s_at;1556055_at;1552281_at;37145_at;2093
          Australia
Denmark   "205009_at;206239_s_at;37145_at;205044_at;213385_at;228030_at;205242_at;204818_a
Australia NA
Japan     "206239_s_at;209583_s_at;37145_at;228030_at;206442_at;210143_at;230793_at"
          Japan
Denmark   "202803_s_at;230964_at;213915_at;206239_s_at;1556055_at;1552281_at;37145_at;2093
Australia "206239_s_at;209583_s_at;37145_at;228030_at;206442_at;210143_at;230793_at"
Japan     NA

attr(,"class")
[1] "VennMapper.res"
```

vm$conting.tab is the contingency table of the numbers of overlapping genes. *vm$z.score* is the table of z-scores corresponding to the overlaps and *vm$genes* is the table of the overlapping genes.

Chapter 8

MAP-Matches

Introduction

Meta-Analysis Pattern Matches (MAP-Matches) [21] is a method that extends VennMapping [20] and meta-profiling [22]. It is designed to analyze more distinct microarray data (search for common molecular mechanism in all types of cancer). It assumes same gene set in all data sets.

Algorithm

Algorithm of this method has five steps:

1. Calculation of T-statistic for each two classes in each data set.
2. Building matrix of T-statistics (T-matrix) with rows referring to genes and columns to pairs of classes and data set.
3. Selection of threshold for T-statistic.
4. Transformation of T-matrix into a binary matrix: 1 for T-statistics above threshold, 0 for T-statistics below threshold.
5. Statistical analysis of transformed T-matrix (more details in Usage section).

Usage

Function `MAP.Matches` provides all steps of MAP-Matches method in one-line command. In

```
> map <- MAP.Matches(ColonData, "MSIstatus", t.cutoff = "95.00%",
+ nperm = 100, sig.col = "p.col.strong")
```

```
Examinig the data...
Statistical analysis...
Permutation:
50
```

```
100
Permuting class labels in dataset:
Denmark
Australia
Japan
Permutation:
50
100
```

the first argument is the dataset (MetaArray object), the second one is a column name of the class labels, `nperm` denotes the number of permutations and the last one `sig.col` is column name for selection of significant patterns - `"p.col.strong"` refer to the p-value of the strong matches when permutation of columns are used.

It returns a list of seven slots:

- *tests* is a vector of test statistics
- *bin.matrix* is a matrix of genes selected in each of the datasets
- *sumarization* is a sumarization of *bin.matrix*: the number of selected genes, the observed patterns and their empirical probabilities
- *MAP* is data frame with the numbers of observed strong or soft matches for each gene pattern
- *stat.analysis* extends *summarization* with the p-values of the permutation-based tests of the statistical significance of the observed number of gene pattern matches
- *genes* is a list of gene names, where each slot refers to one gene pattern and contains the names of the genes for which the pattern was observed.
- *all.genes* is a vector of the names of the all genes in the analysis

Chapter 9

METRADISC

Introduction

METRADISC [23] is unique among rank-based methods because it provides an estimate of heterogeneity as one of its outputs. Additionally the method can deal with genes which are being measured in only some of the studies. The implementation available in MAMA package is restricted to genes common in all microarray studies analyzed.

Algorithm

1. Gene Ranking - In microarray analysis we usually test samples for a large number of genes. The results provide for each gene a test statistic and its statistical significance (p-value). Therefore we can rank the tested genes in each study based on direction in expression change and statistical significance. If there are n genes being tested, the highest rank n is given to the gene that shows the lowest p-value and it is up-regulated in diseased samples. Then follow all other up-regulated genes ranked according to increasing p-value. These are followed by down-regulated genes and the lowest rank (1) is given to gene that shows the lowest p-value and is down-regulated in diseased samples. Genes with equal p-values are assigned tied ranks.

2. The Average Rank and Heterogeneity metrics - In this step we compute a average rank and heterogeneity metrics. The average rank R^* is defined as $R^* = \frac{\sum_{i=1}^{s} R_i}{s}$, where R_i is the rank of the gene in study i and s is total number of studies ($i = 1, 2, ..., s$). The heterogeneity metrics Q^* is given by formula $Q^* = \sum_{i=1}^{s} (R_i - R^*)^2$, it is actually generalization of Cochran's Q statistic.

3. Monte Carlo permutation test - To obtain statistical significance for average rank and heterogeneity metrics we randomly permute the ranks of each study and the stimulated metrics are calculated. Then we repeat the procedure to generate null distribution for the metrics. Each variable is then tested against the corresponding null distribution. We are interested

genuinely in four statistical significances: for high average rank, for low average rank, for high heterogeneity and for low heterogeneity. Distinction between high and low average rank is important as we want to keep the direction of effect in mind. Ignoring it can lead to spurious results that a gene is consistently significant even if it is up-regulated in one study and down-regulated in second one. On the other hand, statistically low heterogeneity may suggest consistent results among different studies. The statistical significance for high average rank (R^*) is defined as the percentage of simulated metrics that exceed or are equal to the observed (R^*). The statistical significance for low average rank (R^*) is defined as the percentage of simulated metrics that are below or equal to the observed (R^*). Significance of heterogeneity is defined analogously.

Usage

Function `METRADISC` performs all steps of this method in one command.

```
> metra<-METRADISC(ColonData, "MSIstatus", nperm = 1000)
```

```
100    200    300    400    500    600    700    800    900    1000
```

```
> str(metra)
```

```
List of 3
 $ ranks :'data.frame':        500 obs. of  3 variables:
  ..$ Denmark  : num [1:500] 105 326 348 390 473 484 83 126 498 482 ...
  ..$ Australia: num [1:500] 368 257 488 397 170 424 179 187 490 495 ...
  ..$ Japan    : num [1:500] 156 153 380 442 445 419 88 187 390 469 ...
 $ RQ    : num [1:500, 1:2] 210 245 405 410 363 ...
  ..- attr(*, "dimnames")=List of 2
  .. ..$ : chr [1:500] "217562_at" "203766_s_at" "1554394_at" "212662_at" ...
  .. ..$ : chr [1:2] "r.star" "q.star"
 $ MCtest: num [1:500, 1:4] 0.698 0.547 0.029 0.028 0.094 0.008 0.953 0.826 0.002 0.001 ..
  ..- attr(*, "dimnames")=List of 2
  .. ..$ : chr [1:500] "217562_at" "203766_s_at" "1554394_at" "212662_at" ...
  .. ..$ : chr [1:4] "R.high" "R.low" "Q.high" "Q.low"
 - attr(*, "class")= chr "METRADISC.res"
```

It returns a list with ranks for each gene and dataset $r$anks, average rank and heterogeneity for each gene RQ and results of the Monte Carlo test MCtest.

Chapter 10

Results combination

In this chapter we are going to compare and combine outputs from all methods so we can look and changes in gene expression in various ways.

We are going to start with lists of differentially expressed genes, because this is the only one output common for all methods mentioned in this vignette. We will merge all lists into one variable via function `join.DEG`.

```
> lists<-join.DEG(pval, es, es2, SOGL.res, rp, z.stat,
+   map,  cutoff=0.05)
> names(lists)<-c("PvalCom", "ESCom","ESCom2","OrderedList",
+ "RankProduct", "Z-stat","MAP")
> summary(lists)
```

```
            Length Class  Mode
PvalCom     212    -none- character
ESCom       166    -none- character
ESCom2      189    -none- character
OrderedList  11    -none- character
RankProduct 179    -none- character
Z-stat      184    -none- character
MAP          18    -none- character
```

Now, we will transform this list to a binary matrix where rows refer to genes and columns to method and 1 means that the gene was identified as a differentially expressed gene in the method. Function `make.matrix` provides such transformation.

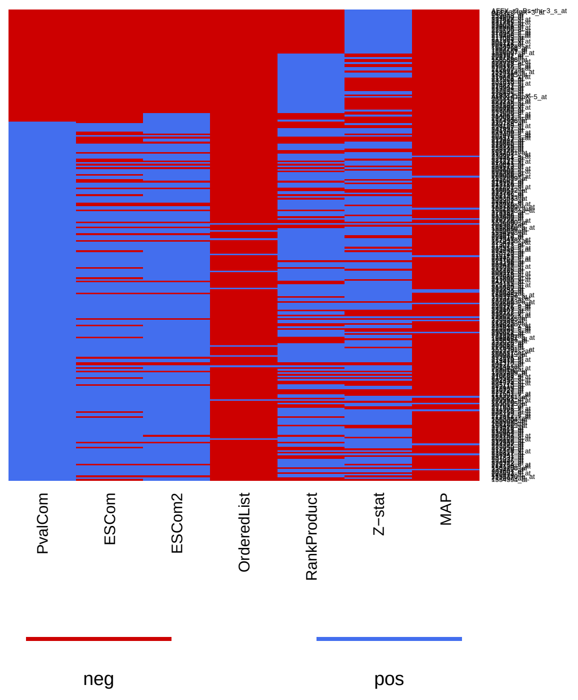
```
> MAT<-make.matrix(lists)
> MAT[1:5,1:5]
```

```
             PvalCom ESCom ESCom2 OrderedList RankProduct
1554394_at         1     0      1           0           0
212662_at          1     1      1           0           0
1555370_a_at       1     0      0           0           1
240574_at          1     1      1           0           1
203553_s_at        1     1      1           0           0
```

It is very popular to visualize results of microarray analysis as a heatmap. A heatmap is a graphical representation for a numeric matrix where values are presented as colors. Gene expression values are usually used in microarray analysis. In these pictures colors go continuously from green (for down-regulation) through black (for no change in gene expression) to red (for up-regulation). There are several R-packages which implement plotting heatmaps in slightly different way. Functions `metaheat` and `metaheat2` are modification of two of them, so a discrete set of colors (only two in `metaheat` but even several in `metaheat2`) can be used with an appropriate legend.

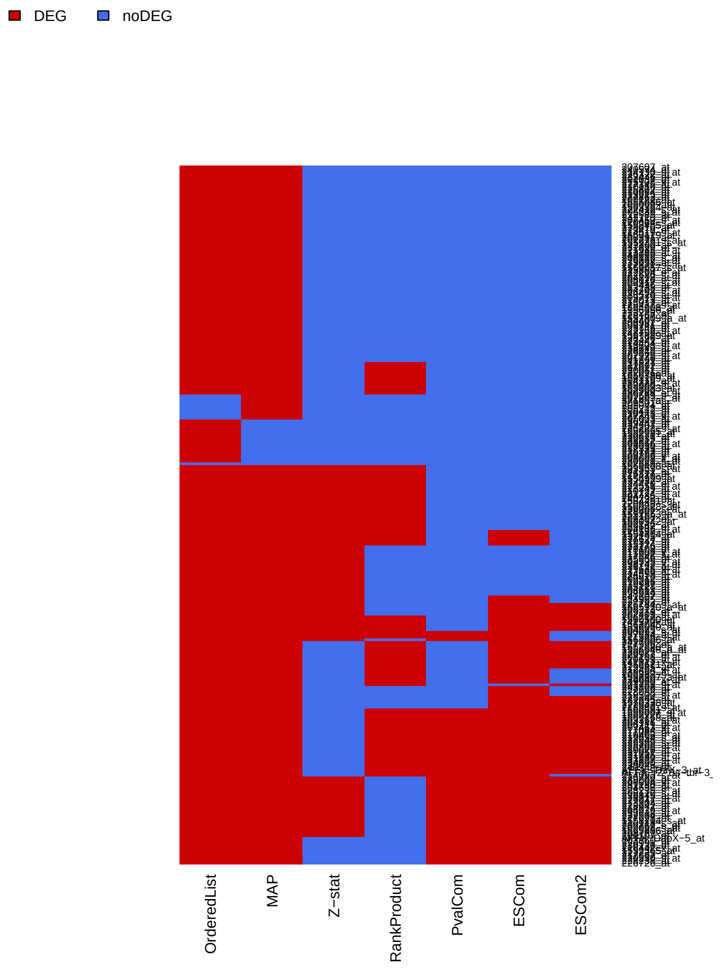
Function `metaheat` has three arguments: a data matrix (`MAT`), a number defining position of legend (`legend=1` is legend drawn below the picture) and vector of colors (`col`).

```
> metaheat(MAT, legend=1, col=c("red3", "royalblue2"))
```



Function `metaheat2` has as many arguments as `heatmap.2` form gplots package and two more. Argument `legend.names` is a character vector with labels to be used in legend. Setting `discret=TRUE` will indicate that legend for discrete values should be drawn.

```
> metaheat2(MAT, col=c("red3", "royalblue2"), legend.names=c("DEG",
+ "noDEG"), discrete=TRUE, trace="none",
+ dendrogram="none", cexCol=0.9)
```



The user can perform cluster analysis on `MAT` to search for similarities between methods or genes.

We can look at number of genes found by number of methods by

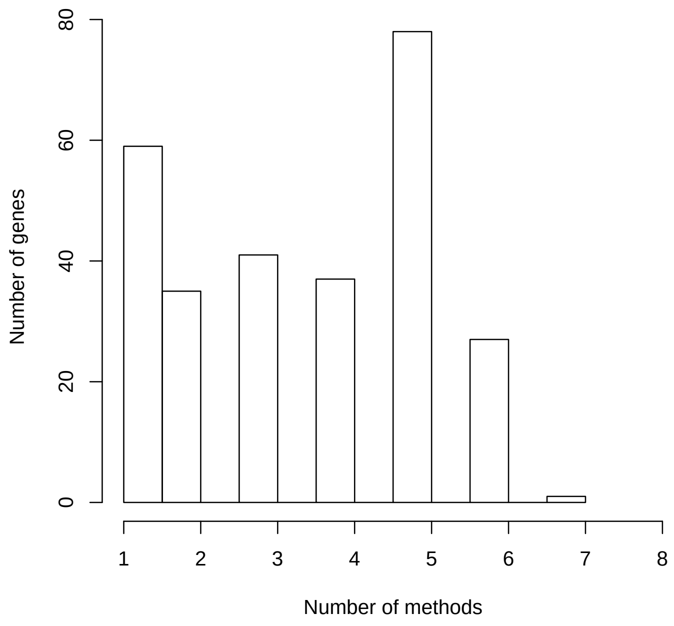
```
> dim(MAT)
```

```
[1] 278   7
```

According to the outsprint above, eight different methods have found more than 270 differentially expressed genes.

The histogram below shows that the most of the genes have been selected in only one method.

```
> n.met<-apply(MAT,1,sum)
> hist(n.met, main="", xlab="Number of methods",
+ ylab="Number of genes", xlim=c(1,8))
```



`n.met` is a numeric vector of number of methods that identified the gene as differentially expressed.

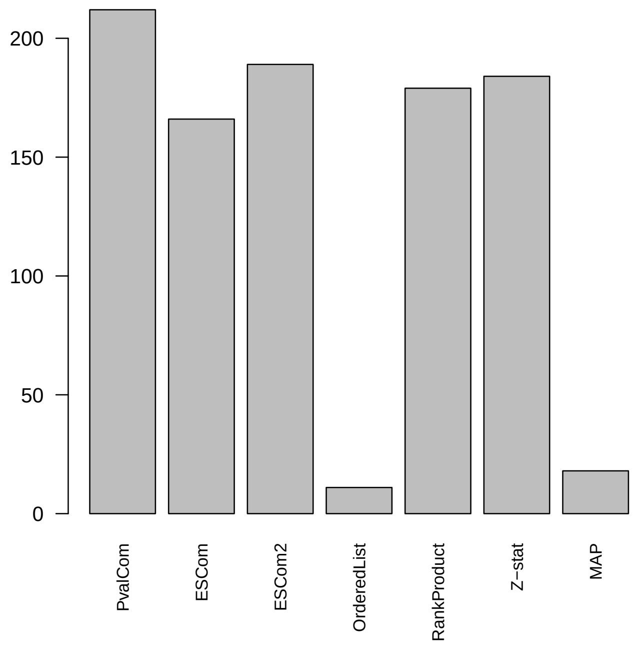
Next, we can look for example how many genes have been found as differentially expressed in at least 6 methods.

```
> dim(MAT[n.met>5,])
```

```
[1] 28  7
```

On the other hand, we can find out how many genes have been found by a method.

```
> n.gen<-apply(MAT,2,sum)
> barplot(n.gen, cex.names=0.8, las=2)
```



Function `contig.tab` provides a number of genes common in two gene lists. It can be applied to `lists`, too.

```
> TAB<-conting.tab(lists)
> TAB[1:5,1:5]
```

```
            PvalCom ESCom ESCom2 OrderedList RankProduct
PvalCom          NA   166    184          11         143
ESCom           166    NA    165          11         126
ESCom2          184   165     NA          11         134
OrderedList      11    11     11          NA          11
RankProduct     143   126    134          11          NA
```

Expression of one gene

In this section we are going to focus on one gene and to look at its expression change from different points of view. The different points of view are represented by different approaches used in the methods.

First we will join all the available results to one list and then select only rows for one gene.

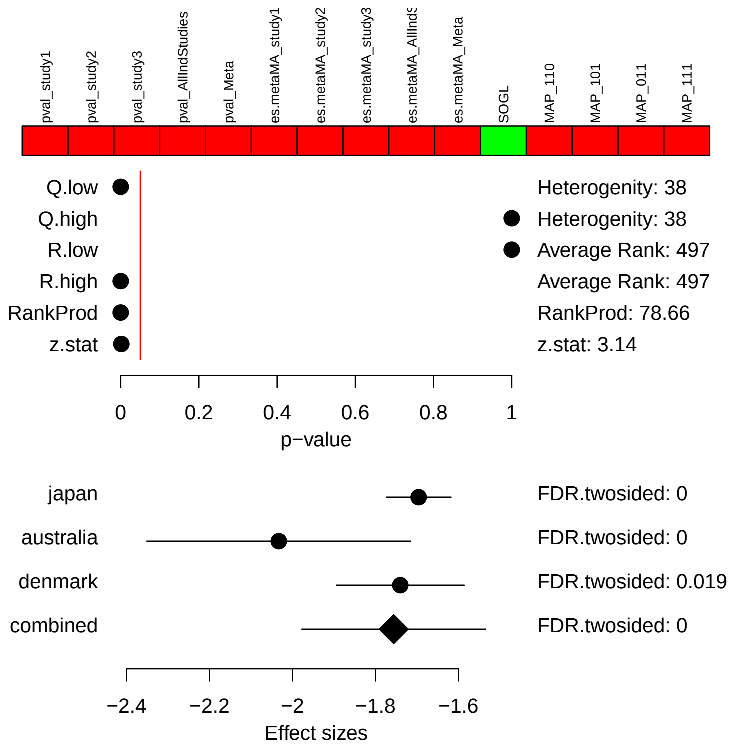
```
> results<-join.results(pval, es, es2, SOGL.res, rp, z.stat, map, metra)
> gene<-metagene("203008_x_at",results)
> names(gene)<-c("pval","es.metaMA", "es.GeneMeta", "SOGL", "RP",
+  "z.stat", "MAP", "METRADISC")
> gene
```

```
$pval
       study1        study2        study3 AllIndStudies
     1.000000      1.000000      1.000000      1.000000
         Meta TestStatistic
     1.000000      8.732214
attr(,"class")
[1] "metaMA"      "metaMA.gene"

$es.metaMA
       study1        study2        study3 AllIndStudies
     1.000000      1.000000      1.000000      1.000000
         Meta TestStatistic
     1.000000     -8.492688
attr(,"class")
[1] "metaMA"      "metaMA.gene"

$es.GeneMeta
     zSco_Ex_1      zSco_Ex_2      zSco_Ex_3           zSco
   -6.44329249    -3.76525489    -4.69659562    -8.80275939
        MUvals          MUsds          Qvals             df
   -1.77309287     0.20142467     0.26260012     2.00000000
      Qpvalues          Chisq    Effect_Ex_1    Effect_Ex_2
    0.87695460     0.00000000    -1.71847860    -2.02486588
   Effect_Ex_3 EffectVar_Ex_1 EffectVar_Ex_2 EffectVar_Ex_3
   -1.75867849     0.07113324     0.28920365     0.14021890
     zSco_Ex_1       FDR_Ex_1      zSco_Ex_2       FDR_Ex_2
   -6.44329249     0.00000000    -3.76525489     0.01875000
     zSco_Ex_3       FDR_Ex_3           zSco            FDR
   -4.69659562     0.00000000    -8.80275939     0.00000000
        MUvals          MUsds          Qvals             df
   -1.77309287     0.20142467     0.26260012     2.00000000
      Qpvalues          Chisq
    0.87695460     0.00000000
attr(,"class")
[1] "ES.GeneMeta"      "ES.GeneMeta.gene"

$SOGL
NULL
<0 rows> (or 0-length row.names)

$RP
        gene.index           RP/Rsum FC:(class1/class2)
          415.0000           78.6607             0.5803
               pfp           P.value
            0.0000            0.0000
attr(,"class")
[1] "RankProduct"      "RankProduct.gene"

$z.stat
              Zscore      Pvalue
```

```
203008_x_at 3.140616 0.001685928

$MAP
 110  101  011  111
TRUE TRUE TRUE TRUE
attr(,"class")
[1] "MAP.Matches"      "MAP.Matches.gene"

$METRADISC
r.star q.star R.high  R.low Q.high  Q.low
   497     38      0      1      1      0
attr(,"class")
[1] "METRADISC"      "METRADISC.gene"

attr(,"class")
[1] "metagene"
```

This output provides much of the information available on the gene through all the described methods. It is a rather complicated structure, so we will try to represent it graphically in comprehensible form.

```
> plotgene(gene, datalabels=c("denmark", "australia", "japan", "combined"))
```



The picture above shows in top part occurrence of gene in lists of selected genes in some methods. The dark box means that the gene is present in the list. Values from objects: `pvalt`, `ESt`, `x.z` and `probs` are used in here.

The middle part is dedicated to p-values available in meta-analysis. Specific values of the statistics can be found on the right side of the chart. The vertical dashed line denotes the signifficance threshold 5%. P-values from `MC`, `RankRes` and `z.stat` are drawn in here.

Combination of effect size is plotted in the bottom graph. The point marks the effect size. Horizontal lines denote the variance of effect size. Statistical significance of the difference in gene expression (FDR adjusted) can be found on the right side of the chart. This graph uses values from `theScores` and `ScoresFDR`.

Bibliography

[1] Jorissen, R. N., Lipton, L., Gibbs, P., Chapman, M. et al. 2008, *DNA copy-number alterations underlie gene expression differences between microsatellite stable and unstable colorectal cancers*, Clinical Cancer Research, Vol. 14, pp. 8061-8069

[2] Watanabe, T., Kobunai, T., Toda, E., Yamamoto, Y. et al. 2006, *Distal colorectal cancers with microsatellite instability (MSI) display distinct gene expression profiles that are different from proximal MSI cancers* Cancer Research, Vol.66, no. 20, pp. 9804-9808

[3] Falcon, S., Morgan, M. and Gentleman, R. 2007, *An introduction to Biocinductor's ExpressionSet class*, available at: http://www.bioconductor.org/packages/2.2/bioc/vignettes/Biobase/inst /doc/ExpressionSetIntroduction.pdf

[4] Marot, G., Foulley, J.L., Mayer, C.D.,Jaffrezic, F. 2009, *Moderated effect size and P-value combinations for microarray meta-analyses*, Bioinformatrics, Vol. 25 no. 20 2009, pp. 2692-2699

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

[6] Fisher, R.A. 1925, *Statistical methods for research*, Oliver and Boyd, Edinburgh

[7] Smyth, G. K. 2004, *Linear models and empirical Bayes methods for assessing differential expression in microarray experiments*, Statistical Applications in Genetics and Molecular Biology 3, No. 1, Article 3

[8] Jaffrezic, F., Marot, G., Degrelle, S., Hue, I., Foulley, J.L. 2007, *A structural mixed model for variances in differential gene expression studies*, Genetical Research, Vol. 89, pp. 19-25.

[9] Choi, J.K., Yu, U., Kim, S. a Yoo, O.J. 2003, *Combining multiple microarray studies and modeling interstudy variation*, Bioinformatics, Vol. 19, Suppl. 1 2003, pp. i84-i90

[10] Gentleman, R., Rauschhaupt, M., Huber, W., a Lusa L. 2008, *Meta-analysis for Microarray Experiments*, available at: http://www.bioconductor.org/packages/2.3/bioc/vignettes/GeneMeta /inst/doc/GeneMeta.pdf

[11] Hedged, V. L. a Olkin, I. 1985, *Statistical Methods for Metaanalysis*, Academic Press, Orlando

[12] Cochran, B.G. 1954, *The combination of estimates from different experiments*, Biometrics, Vol. 10, pp. 101-129

[13] DerSimonian, R., a Laird, N. M. 1986, *Meta-analysis in clinical trials*, Controlled Clinical Trials, Vol. 7, pp. 177-188

[14] Scheid, S., Lottaz, C., Yang, X. a Spang, R. 2006, *Similarities of Ordered Gene Lists User's Guide to the Bioconductor Package OrderedList 1.11.3*, availale at: http://www.bioconductor.org/packages/2.5/bioc/vignettes/OrderedList/inst/doc/tr_2006_01.pdf

[15] Hong, F., Breitling, R., McEntee,C. W., Wittner, B. S., Nemhauser, J. L. a Chory, J. 2006, *RankProd: a bioconductor package for detecting differentially expressed genes in meta-analysis*, Bioinformatics, Vol. 22, no. 22 2006, pp. 2825-2827

[16] Wang, J., Coombes, K. R., Highsmith, W. E., Keating, M. J. a Abruzzo, L. V. 2004, *Differences in gene expression between B-cell chronic lymphocytic leukemia and normal B cells: a meta-analysis of three microarray studies*, Bioinformatics, Vol. 20, no. 17 2004, pp. 3166-3178

[17] Ghosh, D. a Choi, H. 2009, *metaArray package for meta-analysis of microarray data*, available at: http://bioconductor.org/packages/2.5/bioc/vignettes/metaArray/inst/doc/metaArray.pdf

[18] Geman, D., d'Avignon, Ch., Naiman, D. Q. a Winslow, R.L. 2004, *Classifying Gene Expression Profiles from Pairwise mRNA Comparisons*, Statistical Applications in Genetics and Molecular Biology 2004, Vol. 3, Issue 1, Article 19

[19] A.C. Tan, D.Q. Naiman, L. Xu, R.L. Winslow, D. Geman, *Simple decision rules for classifying human cancers from gene expression profiles*, Bioinformatics, 21: 3896-3904, 2005.

[20] Smid, M., Dorssers, L. C. J. a Jenster, G. 2003, *Venn Mapping: clustering of heterologous microarray data based on the number of co-occurring differentially expressed genes*, Bioinformatics, Vol. 19, no. 16 2003, pp. 2065-2071

[21] Yang, X., Bentink, S. a Spang, R. 2005, *Detecting Common Gene Expression Patterns in Multiple Cancer Outcome Entities*, Biomedical Microdevices, Vol.7:3, pp. 247-251

[22] Rhodes, D. R., Yu, J., Shanker, K., Deshpande, N., Varambally, R., Ghosh, D., Barrette, T., Pandey, A. a Chinnaiyan, A. M. 2004, *Large-scale meta-analysis of cancer microarray data identifies common transcriptional profiles of neoplastic transformation and progression*, PNAS, Vol. 101, no. 25, pp. 9309-9314

[23] Zintzaraz, E a Ioannidis, J.P.A. 2008, *Meta-analysis for ranked discovery datasets: Theoretical framework and empirical demonstration for microarrays*, Computational Biology and Chemistry 32, pp. 39-47