Using metaMA for differential gene expression analysis from multiple studies

Guillemette Marot and Rémi Bruyère

Modified: January 28, 2015. Compiled: September 13, 2024

Abstract

This vignette illustrates the use of the *metaMA* package to combine data from multiple microarray experiments. Based on real publicly available data, it also explains the way to format them and draw conclusions thanks to annotation data. Since the use of GEOquery might take time, the command lines are not anymore evaluated during the building of the package. Please do not hesitate to contact the maintainer of the package (guillemette.marot@inria.fr) if you see a mistake.

Contents

1 Introduction

Microarray experiments have been widely used to study differential gene expression between normal and tumoral tissues or other experimental conditions. Very often, the number of repli-

cates is small due either to the cost of the experiment or the lack of biological replicates available. Meta-analyses have increased sensitivity by combining different studies but it is expected that even more sensitivity can be obtained using shrinkage approaches when the number of samples is small in each individual study. *metaMA* proposes a method to calculate moderated effect sizes or p-values from standard and moderated t-tests. The p-values are obtained from *limma*, *SMVar* or *t-test* procedures. *metaMA*'s methods can be applied on different datasets such as those found in public microarray databases.
`pvalcombination(esets, classes)` method combines p-values from different datasets (`esets`) to extract differentially expressed genes between two conditions (e.g. healthy, ill) coded by 0 or 1 for each biological sample in (`classes`). `esets` a is list of matrices which contains genes expression, with one matrix for each study. Sections 2 and 4 concern its building. `classes` is a list of vectors. Each vector codes by 0 samples in the first condition, and by 1 samples in the second condition. Section 3 gives an example.

2 Download public data with GEOquery

metaMA requires normalized micro-array data. In this section, *GEOquery* [1] is used to get some public data from head and neck cancers (might take time).

```
> library(GEOquery)
```

```
> data1 = getGEO('GSE9844')
> data2 = getGEO('GSE3524')
> data3 = getGEO('GSE13601')
```

Note: *Before using GEOquery on a Linux machine, make sure XML and Curl are up-to-date. Use the following command before starting R :* `sudo apt-get install r-base-core libxml2-dev libcurl4-openssl-dev curl`
GEOquery gives `ExpressionSet` which might be already normalized. Boxplots show their distributions.

```
> par(mfrow=c(2,2))
> boxplot(data.frame(exprs(data1[[1]])),main="data1",outline=FALSE)
> boxplot(data.frame(exprs(data2[[1]])),main="data2",outline=FALSE)
> boxplot(data.frame(exprs(data3[[1]])),main="data3",outline=FALSE)
```

While data1 and data2 look like normalized data, data3 presents large ranges of values, which might not be normalized. Since the present vignette does not aim at discussing the choice of the normalization method, we simply log2-transform raw intensities values. The reader is however invited to take special care of normalization on his own data and if possible, to use the same normalization strategy for each study.

```
> exprs(data3[[1]])<-log2(exprs(data3[[1]]))
> boxplot(data.frame(exprs(data3[[1]])),main="data3",outline=FALSE)
```

3 Conditions

metaMA compares genes expressions between 2 conditions (e.g. healthy vs ill). This step explains how to build the `classes` list to determine which replicate is in each condition (here tumor is coded by 1, control by 0).

```
> c1=as.numeric(pData(data1[[1]])["source_name_ch1"]==
+                  "Oral Tongue Squamous Cell Carcinoma")
> c2=as.numeric(apply(pData(data2[[1]])["description"],
+                      1,toupper)=="SERIES OF 16 TUMORS")
> c3=as.numeric(pData(data3[[1]])["source_name_ch1"]=="Tumor")
> classes=list(c1,c2,c3)
```

In public data, like in our example, information about diseases can be found in `pData`. Otherwise, theses classes can be built manually. For example, with 2 arbitrary studies :

```
> c1=c(1,0,1,1,0,1,0,0,0,1)
> c2=c(1,1,1,0,0,0)
> classes2=list(c1,c2)
```

4 Convert esets to an Unigene format

The probes of these 3 datasets are not the sames, but they have genes in common. The following functions link probes `Entrez Gene ID` and UNIGENE identifiers to intersect all common genes. The function `probe2unigene` returns all UNIGENE identifiers from probes available in microarrays. The function `unigene2probe` gives the correspondance back from a UNIGENE to `Entrez Gene ID`. To run this example, it is necessary to load the Bioconductor package *org.Hs.eg.db*. If not installed, the two following commands can be used to install it (delete the comment sign before):

```
> #source("http://bioconductor.org/biocLite.R")
> #biocLite("org.Hs.eg.db")

> require("org.Hs.eg.db")
> x <- org.Hs.egUNIGENE
> mapped_genes <- mappedkeys(x)
> link <- as.list(x[mapped_genes])

> probe2unigene<-function(expset)
+ {
+   #construction of the map probe->unigene
+   probes=rownames(exprs(expset))
+   gene_id=fData(expset)[probes,"ENTREZ_GENE_ID"]
```

```
+   unigene=link[gene_id]
+   names(unigene)<-probes
+   probe_unigene=unigene
+ }

> unigene2probe<-function(map)
+ {
+   suppressWarnings(x <- cbind(unlist(map), names(map)))
+   unigene_probe=split(x[,2], x[,1])
+ }
```

Note that probes which can not be linked to a UNIGENE identifier will not be taken into account in the meta-analysis. Because one unigene corresponds to many probes, it is necessary to choose a strategy to merge such probes from a single study. In the following, convert2metaMA chooses to summarize values of probes corresponding to the same UNIGENE by their mean, as attested by the default value given to the argument mergemeth.

```
> convert2metaMA<-function(listStudies,mergemeth=mean)
+ {
+   if (!(class(listStudies) %in% c("list"))) {
+     stop("listStudies must be a list")
+   }
+   conv_unigene=lapply(listStudies,
+                       FUN=function(x) unigene2probe(probe2unigene(x)))
+   id=lapply(conv_unigene,names)
+
+   inter=Reduce(intersect,id)
+   if(length(inter)<=0){stop("no common genes")}
+   print(paste(length(inter),"genes in common"))
+   esets=lapply(1:length(listStudies),FUN=function(i){
+     l=lapply(conv_unigene[[i]][inter],
+              FUN=function(x) exprs(listStudies[[i]])[x,,drop=TRUE])
+     esetsgr=t(sapply(l,FUN=function(ll) if(is.null(dim(ll))){ll}
+                      else{apply(ll,2,mergemeth)}))
+     esetsgr
+   })
+   return(list(esets=esets,conv.unigene=conv_unigene))
+ }
> conv=convert2metaMA(list(data1[[1]],data2[[1]],data3[[1]]))
> esets=conv$esets
> conv_unigene=conv$conv.unigene
```

The function convert2metaMA builds matrices for each study containing only rows corresponding to the identifiers common to all studies (based on the conversion via UNIGENE).

These matrices are stored in the sublist `esets` of the value returned by the function. In this example, the intersection of the 3 studies includes 4853 genes. `convert2metaMA` also returns `conv.unigene` to be able to go back to original probe names after meta-analysis.

5 metaMA Application

```
> library(metaMA)
> res=pvalcombination(esets=esets,classes=classes)
> length(res$Meta)
> Hs.Meta=rownames(esets[[1]])[res$Meta]
```

The p-value combination gives different indicators to evaluate the performance of the meta-analysis. DE corresponds to the number of differentially expressed genes. IDD (Integration Driven discoveries) returns the number of genes that are declared DE in the meta-analysis that were not identified in any of the individual studies alone, Loss the number of genes that are identified DE in individual studies but not in meta-analysis. The Integration-driven Discovery Rate (IDR) and Integration-driven Revision Rate (IRR) are the corresponding proportions of IDD and Loss.

6 Venn Diagram

To compare visually the number of differentially expressed genes in individual studies or in meta-analysis, it is possible to draw a Venn diagram, for example with the *VennDiagram* package.

```
> library(VennDiagram)
> venn.plot<-venn.diagram(x = list(study1=res$study1,
+                                  study2=res$study2,
+                                  study3=res$study3,
+                                  meta=res$Meta),
+              filename = NULL, col = "black",
+              fill = c("blue", "red", "purple","green"),
+              margin=0.05, alpha = 0.6)
> jpeg("venn_jpeg.jpg")
> grid.draw(venn.plot)
> dev.off()
```

7 Annotations

To obtain annotations related to initial probes, it is necessary to know the chip types used in each study. It is preferable to know them in advance (to save time) but it is also possible to get it with the following `getanndb` function.

```
> getanndb<-function(expset)
+ {
+   gpl_name=annotation(expset)
+   gpl=getGEO(gpl_name)
+   title=Meta(gpl)$title
+   db=paste(gsub("[-|_]","",
+                 tolower(strsplit(title,"[[]|[]]")
+                         [[1]][[2]])),"db",sep=".")
+   return(db)
+ }
> db1=getanndb(data1[[1]])
> db1
> db2=getanndb(data2[[1]])
> db2
> db3=getanndb(data3[[1]])
> db3
```

It is then necessary to download annotation packages corresponding to these chip types (use biocLite to install them if necessary).

```
> #source("http://bioconductor.org/biocLite.R")
> #biocLite(db1)
> #biocLite(db2)
> #biocLite(db3)
> library(db1,character.only=TRUE)
> library(db2,character.only=TRUE)
> library(db3,character.only=TRUE)
```

In the following code, `origId.Meta` gives back for each study the original ids corresponding to the common `Hs.Meta` ids of the differentially expressed genes in the meta-analysis. Since several original probes matched the same UNIGENE identifiers, there might be many more ids in `origId.Meta` than in `Hs.Meta`. Then, functions `aafTableAnn` and `saveHTML` from the package `annaffy` are used to save in the current directory an HTML file "annotation.html", which provides biological information about identified probes.

```
> origId.Meta=lapply(conv_unigene,
+                    FUN=function(vec) as.vector(unlist(vec[Hs.Meta])))
> library(annaffy)
> annlist=lapply(1:length(origId.Meta),
+                FUN=function(i) aafTableAnn(origId.Meta[[i]],chip=get(pa
> annot=do.call(rbind,annlist)
> saveHTML(annot,file="annotation.html",title="Responder genes")
```

8 Session Info

```
> sessionInfo()

R version 4.4.1 (2024-06-14)
Platform: x86_64-pc-linux-gnu
Running under: Ubuntu 24.04.1 LTS

Matrix products: default
BLAS:   /usr/lib/x86_64-linux-gnu/openblas-pthread/libblas.so.3
LAPACK: /usr/lib/x86_64-linux-gnu/openblas-pthread/libopenblasp-r0.3.26.so

locale:
 [1] LC_CTYPE=en_US.UTF-8       LC_NUMERIC=C
 [3] LC_TIME=en_US.UTF-8        LC_COLLATE=C
 [5] LC_MONETARY=en_US.UTF-8    LC_MESSAGES=en_US.UTF-8
 [7] LC_PAPER=en_US.UTF-8       LC_NAME=C
 [9] LC_ADDRESS=C               LC_TELEPHONE=C
[11] LC_MEASUREMENT=en_US.UTF-8 LC_IDENTIFICATION=C

time zone: Etc/UTC
tzcode source: system (glibc)

attached base packages:
[1] stats     graphics  grDevices utils     datasets
[6] methods   base

loaded via a namespace (and not attached):
[1] compiler_4.4.1   tools_4.4.1      maketools_1.3.0
[4] buildtools_1.0.0 knitr_1.48       xfun_0.47
[7] sys_3.4.2
```

References

[1] S. Davis and P. Meltzer. Geoquery: a bridge between the gene expression omnibus (geo) and bioconductor. *Bioinformatics*, 14:1846–1847, 2007.