Lab 4: Differentially expressed genes

March 7th, 2013

We have seen how to preprocess microarrays from raw data to gene expression levels – now we have a data object from which we can retrieve a matrix with rows representing genes and columns representing samples.

1 Limma

Note: if you do not have data package "estrogen" installed, use

```
source("http://bioconductor.org/biocLite.R")
biocLite("estrogen")
```

```
> library(limma)
> library(estrogen)
```

After the data package is installed we also have some phenotypic data files downloaded. The following functions show what files are installed:

```
> datadir = file.path(.find.package("estrogen"),"extdata")
> #file.path: Construct the path to a file from components in a
> #platform-independent way.
> #.find.package: Find the paths to one or more packages.
> dir(datadir)
```

```
 [1] "bad.cel"       "estrogen.txt"  "high10-1.cel"  "high10-2.cel"
 [5] "high48-1.cel"  "high48-2.cel"  "low10-1.cel"   "low10-2.cel"
 [9] "low48-1.cel"   "low48-2.cel"   "phenoData.txt" "targLimma.txt"
```

Now let's read in the data and process it into expression values.

```
> targets =  readTargets("phenoData.txt",path=datadir,sep="",row.names="filename
> #row.names: character string giving the name of a column from which to
> # obtain row names
> #readTargets: Read targets file for a microarray experiment into a
```

```
> #dataframe.
> ab = ReadAffy(filenames=targets$filename,celfile.path=datadir,
+               phenoData=targets)
> class(ab)

[1] "AffyBatch"
attr(,"package")
[1] "affy"

> eset = rma(ab)

Background correcting
Normalizing
Calculating Expression

> class(eset)

[1] "ExpressionSet"
attr(,"package")
[1] "Biobase"
```

The variable "target" is a data.frame containing information about the samples, and we put in into the *phenoData* slot by providing it in the *ReadAffy* argument. How many groups(types) of samples are there?

Here we will show the example of using *limma*. The main function is "lmFit". The *model.matrix* function creates the design matrix automatically once we pass the covariate vector. Here we fit a two-way ANOVA model considering both the time effect and the estrogen effect, but not the interaction. So there are $2 \times 2 = 4$ groups. We construct the design matrix without an intercept here.

```
> ## design matrix 1
> TS = paste(targets[,2],targets[,3],sep="")
> TS = factor(TS)
> design = model.matrix(~TS-1)
> design

  TSabsent10 TSabsent48 TSpresent10 TSpresent48
1          1          0           0           0
2          1          0           0           0
3          0          0           1           0
4          0          0           1           0
5          0          1           0           0
6          0          1           0           0
7          0          0           0           1
```

```
8          0          0          0          1
attr(,"assign")
[1] 1 1 1 1
attr(,"contrasts")
attr(,"contrasts")$TS
[1] "contr.treatment"
```

```
> colnames(design) = levels(TS)
> fit1=lmFit(eset,design)
```

You may pass the gene expression matrix to function

```
fit1=lmFit(exprs(eset),design)
```

gives you the same result), but it will recognize objects of the class "*ExpressionSet*" as well.

The lmFit function basically repeats the linear model for all genes. But instead of simply repeats, the package has a function *eBayes* that borrows information across genes in estimating the variance using empirical Bayes approach.

Now we need to consider what comparison(s) we are interested in. Say we are interested in the Time effect in absence of estrogen and the estrogen effect at hour 48.

```
> cont.matrix = makeContrasts(TIME=absent48-absent10,E48=present48-absent48,
+                             levels=design)
> fit2 = contrasts.fit(fit1,cont.matrix)
> fit2 = eBayes(fit2)
```

The "fit2" object now contains the comparisons for each gene. Check out the components of fit 2 with "*names(fit2)*". To see the estimated contrasts you can use "*fit2$coef*". Try retrieve the estimated contrasts for the first five genes. Which gene's expression changes the most by time in the absence of estrogen?

```
> names(fit2)
```

```
 [1] "coefficients"     "rank"             "assign"           "qr"
 [5] "df.residual"      "sigma"            "cov.coefficients" "stdev.unscaled"
 [9] "genes"            "Amean"            "method"           "design"
[13] "contrasts"        "df.prior"         "s2.prior"         "var.prior"
[17] "proportion"       "s2.post"          "t"                "df.total"
[21] "p.value"          "lods"             "F"                "F.p.value"
```

```
> head(fit2$coef)
```

```
          Contrasts
                 TIME          E48
  100_g_at  -0.1110524 -0.077917904
  1000_at   -0.1220724 -0.099999757
  1001_at    0.1914043  0.142421367
  1002_f_at -0.2149139  0.126817497
  1003_s_at  0.1367273  0.081459079
  1004_at    0.1142501  0.004993694
```

To list the top differentially expressed genes, a convenient function "topTable" is handy.

```
?topTable
```

```
> #largest change by time in absence of estrogen
> which.max(fit2$coef[,"TIME"])

AFFX-M27830_5_at
           12607
```

For example, to list the top 15 genes showing differential expression when you compare the Estrogen vs Control at hour 48 (this is the second contrast you defined earlier in cont.matrix), you can use

```
> topTable(fit2,coef=2,number=15)
```

```
            ID     logFC   AveExpr         t      P.Value    adj.P.Val
12472   910_at  3.855061  9.660238  29.20918 8.266125e-10 1.043598e-05
1814  31798_at  3.597334 12.115778  21.04947 1.284430e-08 7.631722e-05
953    1854_at  3.340896  8.532099  20.19564 1.813478e-08 7.631722e-05
8195  38116_at  3.758891  9.513109  16.85669 8.116230e-08 2.511100e-04
8143  38065_at  2.993641  9.097183  16.20914 1.121213e-07 2.511100e-04
9848  39755_at  1.765249 12.131839  15.83434 1.359405e-07 2.511100e-04
642    1592_at  2.296484  8.311330  15.78841 1.392293e-07 2.511100e-04
11509 41400_at  2.243510 10.041553  15.28749 1.814762e-07 2.752126e-04
3766  33730_at -2.041390  8.573470 -15.14298 1.961911e-07 2.752126e-04
732    1651_at  2.968283 10.504276  14.78097 2.392480e-07 3.020507e-04
8495  38414_at  2.017022  9.461170  14.59062 2.660400e-07 3.053413e-04
1049   1943_at  2.190062  7.596579  14.00310 3.722630e-07 3.689505e-04
10214 40117_at  2.276228  9.676557  13.96826 3.799095e-07 3.689505e-04
10634 40533_at  1.640412  8.466000  13.52556 4.939428e-07 4.454305e-04
9735  39642_at  1.614445  7.876515  13.02636 6.705257e-07 5.180349e-04
              B
12472 11.606193
1814   9.890557
953    9.641399
```

```
8195   8.480197
8143   8.214175
9848   8.053134
642    8.033025
11509  7.808295
3766   7.741556
732    7.570470
8495   7.478222
1049   7.183041
10214  7.165023
10634  6.930891
9735   6.654811
```

```
> #Extract a table of the top-ranked genes from a linear model fit.
```

Can you find the top 10 differentially genes comparing the estrogen vs control at hour 10 instead? What do you have to modify in your above code?

```
> ## in order to get estrogen vs control at hour 10,
> ## we have to get result including this contrast
> cont.matrix3  = makeContrasts(E10=present10-absent10,levels=design)
> fit3 = contrasts.fit(fit1,cont.matrix3)
> fit3 = eBayes(fit3)
> #largest change by time in absence of estrogen
> topTable(fit3,coef=1,number=15)
```

```
             ID     logFC   AveExpr         t      P.Value    adj.P.Val
9735   39642_at  2.939428  7.876515  23.71715 4.741579e-09 3.128295e-05
12472    910_at  3.113733  9.660238  23.59225 4.955715e-09 3.128295e-05
1814   31798_at  2.800195 12.115778  16.38509 1.025747e-07 3.511070e-04
11509  41400_at  2.381040 10.041553  16.22463 1.112418e-07 3.511070e-04
10214  40117_at  2.555282  9.676557  15.68070 1.472942e-07 3.576234e-04
953     1854_at  2.507616  8.532099  15.15848 1.945518e-07 3.576234e-04
9848   39755_at  1.679331 12.131839  15.06365 2.048314e-07 3.576234e-04
922   1824_s_at  1.914637  9.238870  14.87915 2.266129e-07 3.576234e-04
140   1126_s_at  1.782825  6.879918  13.83040 4.119252e-07 5.778395e-04
580     1536_at  2.662258  5.937222  13.26247 5.795111e-07 7.316327e-04
12542    981_at  1.818408  7.781306  13.08578 6.462082e-07 7.416708e-04
3283   33252_at  1.740213  8.000356  12.58548 8.862851e-07 9.204706e-04
546     1505_at  2.395861  8.764756  12.48145 9.478113e-07 9.204706e-04
4405   34363_at -1.747957  5.553959 -12.19735 1.141480e-06 1.029370e-03
985   1884_s_at  2.799396  9.034796  12.05054 1.258522e-06 1.059256e-03
            B
9735  9.966810
12472 9.942522
```

```
1814  7.977290
11509 7.916921
10214 7.705093
953   7.490766
9848  7.450643
922   7.371475
140   6.892307
580   6.610486
12542 6.519352
3283  6.251908
546   6.194498
4405  6.034408
985   5.949749
```

The default setting in function eBayes used Benjamini-Hochberg adjustment to adjust p-values and the genes are ranked by posterior log odds of differential expression. If you want to perform other multiple testing adjustment, or rank the genes by other parameters, see the help file for options.

Can you find the top 20 differentially expressed genes based on posterior log odds, but sort these 20 genes by the magnitude of log fold change?

```
> topTable(fit3,coef=1,number=20,resort.by="logFC")
```

	ID	logFC	AveExpr	t	P.Value	adj.P.Val
12472	910_at	3.113733	9.660238	23.59225	4.955715e-09	3.128295e-05
9735	39642_at	2.939428	7.876515	23.71715	4.741579e-09	3.128295e-05
1814	31798_at	2.800195	12.115778	16.38509	1.025747e-07	3.511070e-04
985	1884_s_at	2.799396	9.034796	12.05054	1.258522e-06	1.059256e-03
580	1536_at	2.662258	5.937222	13.26247	5.795111e-07	7.316327e-04
10214	40117_at	2.555282	9.676557	15.68070	1.472942e-07	3.576234e-04
953	1854_at	2.507616	8.532099	15.15848	1.945518e-07	3.576234e-04
6194	36134_at	2.491536	8.275734	11.78585	1.504819e-06	1.187396e-03
546	1505_at	2.395861	8.764756	12.48145	9.478113e-07	9.204706e-04
11509	41400_at	2.381040	10.041553	16.22463	1.112418e-07	3.511070e-04
8195	38116_at	2.318493	9.513109	10.39725	4.092989e-06	2.658504e-03
922	1824_s_at	1.914637	9.238870	14.87915	2.266129e-07	3.576234e-04
12542	981_at	1.818408	7.781306	13.08578	6.462082e-07	7.416708e-04
140	1126_s_at	1.782825	6.879918	13.83040	4.119252e-07	5.778395e-04
3283	33252_at	1.740213	8.000356	12.58548	8.862851e-07	9.204706e-04
9848	39755_at	1.679331	12.131839	15.06365	2.048314e-07	3.576234e-04
7557	37485_at	1.607306	6.672714	11.38504	1.986316e-06	1.475132e-03
1244	239_at	1.569568	11.249179	10.40391	4.072260e-06	2.658504e-03
10634	40533_at	1.256466	8.466000	10.35984	4.211491e-06	2.658504e-03
4405	34363_at	-1.747957	5.553959	-12.19735	1.141480e-06	1.029370e-03

```
            B
12472 9.942522
9735  9.966810
1814  7.977290
985   5.949749
580   6.610486
10214 7.705093
953   7.490766
6194  5.793668
546   6.194498
11509 7.916921
8195  4.896058
922   7.371475
12542 6.519352
140   6.892307
3283  6.251908
9848  7.450643
7557  5.548589
1244  4.900709
10634 4.869899
4405  6.034408
```

```
> #resort.by: character string specifying statistic to sort the selected
> # genes by in the output data.frame.
```

For more details of the limma package, consult Chapter 23 of your textbook. The user guides for siggenes and limma can be found

http://bioconductor.org/packages/2.5/bioc/vignettes/limma/inst/doc/limma.pdf
http://bioconductor.org/packages/2.5/bioc/vignettes/limma/inst/doc/usersguide.pdf
http://bioconductor.org/packages/2.5/bioc/vignettes/siggenes/inst/doc/siggenes.pdf