Affymetrix Microarrays

Cavan Reilly

October 21, 2019

Table of contents

Overview

Affymetrix microarrays have seen extensive use: many publically available datasets.

They have unique features and there has been extensive development of R based tools for their analysis

In contrast to the other major manufacturer, Illumina, one has greater access to the “raw data”.

This data is stored as text files so manipulation in an environment like R is straightforward.

The “raw data”

The result from scanning the microarray image is available: these files have the extension `.CEL` and as text files can be read into R.

These files hold the intensities for all probes (mismatch and perfect match), so one can test out different algorithms for estimating gene expression or conduct quality assessment at the probe level.

While the former is not a typical part of a contemporary analysis of gene expression estimation, the latter is recommended.

The ALL data set

We will examine a dataset with samples obtained from a group of chronic lymphocytic leukemia (CLL).

```
> library(affy)
> library(CLL)
> data("CLLbatch")
> CLLbatch
AffyBatch object
size of arrays=640x640 features (99224 kb)
cdf=HG_U95Av2 (12625 affyids)
number of samples=24
number of genes=12625
annotation=hgu95av2
notes=
```

The ALL dataset

We will now read in some data about the disease state of the patients from whom these samples were obtained.

Then we will fix up the sample names so that they match the names in disease.

```
> data(disease)
> rownames(disease)=disease$SampleID
> sampleNames(CLLbatch)=sub("\\.CEL$","",
+  sampleNames(CLLbatch))
> mt=match(rownames(disease),sampleNames(CLLbatch))
```

The ALL dataset

Then we provide better phenotype descriptions via a annotated data frame and get rid of a sample with no data on disease.

```
> vmd=data.frame(labelDescription=c("Sample ID",
+  "Disease status: progressive or stable"))
> phenoData(CLLbatch)=new("AnnotatedDataFrame",
+  data=disease[mt,], varMetadata=vmd)
> CLLbatch=CLLbatch[,!is.na(CLLbatch$Disease)]
```

QA/QC

One should graphically examine the quality of microarrays and if there are signs of overall poor quality, remove the array from the analysis.

The package `affQCReport` will produce a collection of QA plots and a .tex file which will be used to construct a report.

This program will create a subdirectory called `affyQA` inside the directory from which you are running R (in Linux) or under Documents on windows.

QA/QC

Then inside this directory will be a subdirectory with the name of the ExpressionSet object for which you are generating a quality report.

Then inside this directory will be a collection of pdf images, a .tex file and a pdf of the report if you have latex installed.

This will produce all of the plots in Section 3.2 of the text.

QA/QC

The data for the array with name CLL1 looks like it might have some problems: we can get rid of it with the following syntax.

```
> badArray=match("CLL1", sampleNames(CLLbatch))
> CLLB=CLLbatch[,-badArray]
```

Preprocessing

There are 3 tasks that are conducted on Affymetrix data sets:

1. background correction
2. between array normalization
3. probe set summarization

While many approaches have been proposed the most commonly used approach to do all of this is to use RMA.

This can be done easily using the `rma` function in the `affy` package: it converts an AffyBatch object into an ExpressionSet object.

Preprocessing

The syntax is quite simple

```
> CLLrma=rma(CLLB)
Background correcting
Normalizing
Calculating Expression
> CLLrma
ExpressionSet (storageMode: lockedEnvironment)
assayData: 12625 features, 22 samples
  element names: exprs
protocolData: none
phenoData
  sampleNames: CLL11 CLL12 ... CLL9 (22 total)
  varLabels: SampleID Disease
  varMetadata: labelDescription
featureData: none
experimentData: use 'experimentData(object)'
Annotation: hgu95av2
```

Non-specific filtering

As discussed previously, probe sets are frequently filtered due to insuffcient variability.

Here we also filter based on probe sets that map to the same Entrez Gene ID (retain the one with the largest variance).

Can also filter out genes that don't have annotation and can dump the Affymertix controls.

```
> CLLf=nsFilter(CLLrma, remove.dupEntrez=F,
+   var.cutof=0.5)$eset
```

Testing for Differential Expression

A common question of interest is if there are genes that have different expression levels across 2 groups.

We can use a 2 sample *t*-test as we've seen before, but there are ways to automate this process.

```
> CLLtt=rowttests(CLLf, "Disease")
```

Testing for Differential Expression

```
> CLLtt[1,]
         statistic          dm  p.value
1000_at -0.5968917 -0.06953843 0.557277
> t.test(exprs(CLLf)[1,pData(CLLf)$Disease=="progres."],
+  exprs(CLLf)[1,pData(CLLf)$Disease=="stable"],var.equal=T)

Two Sample t-test

t = -0.59689, df = 20, p-value = 0.5573
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 -0.3125551  0.1734782
sample estimates:
mean of x mean of y
 8.107587  8.177125
> 8.107587-8.177125
[1] -0.069538
```

Moderated Tests

There are more powerful approaches to testing for a difference between 2 groups than provided by a 2 sample *t*-tests.

These approaches are based on hierarchical Bayesian models that combine information across probe sets to estimate standard deviations.

```
> design=model.matrix(~CLLf$Disease)
> CLLlim=lmFit(CLLf, design)
> CLLeb=eBayes(CLLlim)
```

Moderated Tests

As eBayes is an S3 type object we can examine its details as we've seen many times.

Just as was the case when we looked at large numbers of genetic variants, we need to be aware of multiple hypothesis testing.

```
> table(CLLtt[,3]<.05,CLLeb$p.value[,2]<.05)

        FALSE TRUE
  FALSE  4901   72
  TRUE     34  720
> table(p.adjust(CLLtt[,3],method="BH")<.05,
+  p.adjust(CLLeb$p.value[,2],method="BH")<.05)

        FALSE TRUE
  FALSE  5718    2
  TRUE      0    7
```

Moderated Tests

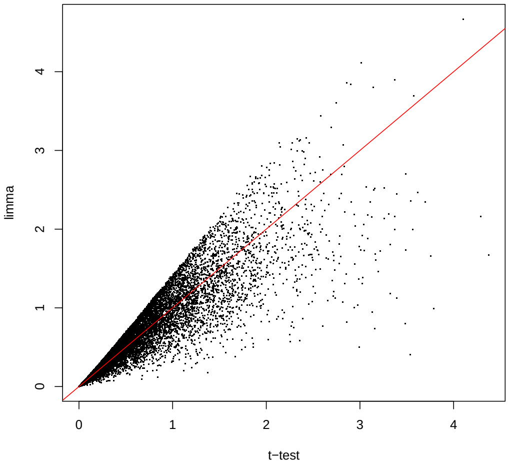
As another example, let's look at the ALL data again except now include only 3 samples per group.

```
> subs=c(35,65,75,1,69,71)
> ALLset2=ALL_bcrneg[,subs]

> cl=as.numeric(ALL_bcrneg$mol.biol=="BCR/ABL")
> design=cbind(mean=1,diff=cl)
> tt2=rowttests(ALLset2, "mol.biol")
> fit2=eBayes(lmFit(exprs(ALLset2),design=design[subs,]))
```

Moderated Tests

If we plot the *p*-values we note that there is substantial disagreement between the methods in this case.



Volcano Plots

It is common to examine unadjusted *p*-values and log fold changes graphically.

By the *fold change* we mean the ratio of the mean level of gene expression in one group to the mean in the other group.

If there is pairing of samples it makes more sense to compute the ratio for each pair, then take the average, as that would be less variable.

```
> plot(CLLtt$dm, -log10(CLLtt$p.value),
+   pch=".",xlab="log-ratio",ylab=expression(-log[10]~p))
```

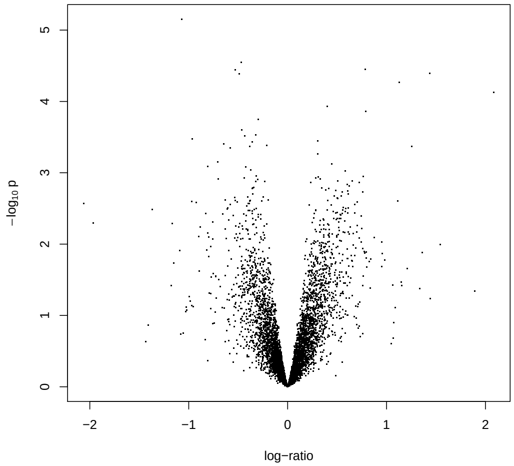


Figure: A volcano plot for the CLL data set.

Annotation

R has multiple ways to keep track of information about various microarrays.

This is useful as it allows one to determine the identity of genes that are represented by probe sets (or probes for non-Affymetrix arrays).

There are multiple ways that this has been implemented in R.

Annotation via environments

We start by loading the annotate package, determine the annotation for our microarray, then load the library specific package for our microarray.

We'll also grab the feature names for those genes that differed.

```
> library(annotate)
Loading required package: XML
> annotation(CLLf)
[1] "hgu95av2"
> library(hgu95av2.db)
> genenames=featureNames(CLLf)[p.adjust(
+  CLLeb$p.value[,2],method="BH")<.05]
```

Annotation via environments

We can use the get function to retreive information about these genes with the following syntax:

```
> ll=getEG(genenames,"hgu95av2")
> sym=getSYMBOL(genenames,"hgu95av2")
> ll
 1303_at 33791_at 36129_at 36131_at 36939_at 37636_at
  "6452"  "10301"   "9905"   "1192"   "2823"   "9767"
  39400_at 41776_at 551_at
   "23102"    "475" "2033"
> sym
  1303_at  33791_at  36129_at  36131_at  36939_at
 "SH3BP2"   "DLEU1"   "SGSM2"   "CLIC1"   "GPM6A"
  37636_at  39400_at  41776_at  551_at
   "JADE3" "TBC1D2B"   "ATOX1" "EP300"
```

HTML Reports

We can also make HTML reports of our analysis using syntax like the following.

```
> tab=topTable(CLLeb, coef=2, adjust.method="BH", n=9)
> tab=data.frame(sym,signif(tab[,-1],3))
> htmlpage(list(ll),othernames=tab,filename=
+  "GeneList1.html",title="HTML Report", table.center=T,
+   table.head=c("Entrez ID",colnames(tab)))
```

That will generate the file `GeneList1.html` in your working directory that can be opened with a browser.

HTML Reports

There are packages that can create more detailed annotation reports for Affymetrix microarrays.

```
> library("annaffy")
> library("KEGG.db")

> atab=aafTableAnn(genenames, "hgu95av2.db",
+  aaf.handler())
> saveHTML(atab, file="GeneList2.html")
```

Again you can inspect this with a browser (note all of the links).