Analysis of Microarray Data

Keith Baggerly and Kevin Coombes
Section of Bioinformatics
Department of Biostatistics and Applied Mathematics
UT M. D. Anderson Cancer Center
kabagg@mdanderson.org
kcoombes@mdanderson.org

12 October 2006
Lecture 13: Reporting Differential Expression

- Borrowing and Ranks
- Tail Rank Revisited
- Differential Expression in BioC
- Annotation and Reporting of Data
Does borrowing help a great deal?

In our earlier discussion of borrowing, we noted that better characterization of the variability did improve things, but maybe not so much.

However, that assessment was predicated on our having a good idea of the underlying distribution to begin with. If the data are skewed or subject to frequent outliers, things can get worse.
Skewness, Outliers, and Bears?

This, of course, is why we often shift to rank tests which don’t depend on the particular shape of the distribution. But, as we saw last time, the discrete nature of the ranks may preclude a rank test from yielding a small p-value even when something extreme is going on.
Linking Borrowing and Ranks

Small p-values, however, can be obtained if we have more “effective samples” with which to characterize the underlying distribution, leading us to combine the idea of borrowing strength across genes with the idea of using ranks to remain less sensitive to the particular shape of the underlying distribution.
The Relative Rank Test

Oddly enough, we haven’t seen that much written about borrowing with ranks, but here goes.

Assume that we are interested in deciding if the levels of gene \( g \) are different between two groups \( A \) and \( B \), and that \( g \) is for the most part contained within a set of genes \( G \) having similar null distributions.

The standard procedure (Wilcoxon) is to rank the \( n_A + n_B \) values of \( g \) and sum the ranks of those in one of the groups (say \( A \)). The p-values are then computed by permutation and counting arguments.
The Relative Rank Test

For Wilcoxon, we note that we could just as easily have worked with the difference in average ranks for $A$ and $B$, respectively, as the total must stay fixed.
The Relative Rank Test

For Wilcoxon, we note that we could just as easily have worked with the difference in average ranks for $A$ and $B$, respectively, as the total must stay fixed.

Here, we rank all $G \times (n_A + n_B)$ expression values within the “relevant set” $G$, and focus on the difference between the average ranks for gene $g$ in groups $A$ and $B$. Here, the choice of just one sum (say the $A$ ranks) or the difference does matter because there are intervening values present.
What does this potentially buy us?
What does this potentially buy us?

The ability to get small p-values
What does this potentially buy us?

The ability to get small p-values

The ability to get large p-values
What does this potentially buy us?

The ability to get small p-values

The ability to get large p-values

The ability to differentiate between “extreme cases”
What does this potentially buy us?

The ability to get small p-values

The ability to get large p-values

The ability to differentiate between “extreme cases”

Some robustness against outliers (we lose some of this relative to the Wilcoxon test, however) or different distributions
What does this potentially cost us?
What does this potentially cost us?

accuracy, if the distributions are starkly different (eg, including high real variability genes with low real variability genes).

The traditional borrowing of strength focuses on a single number (such as the variance) and presumes that will be stable. Rank sharing makes stronger distributional assumptions.
Some Math

What can we say about the distribution of the difference \( \bar{r}_A - \bar{r}_B \)?

Well, if \( G \) is large, then we can effectively ignore the discrete nature of the rank distribution. To make things easier (on me), let’s divide the ranks by \( G \times (n_A + n_B) \) so that we have values ranging from 0 to 1.
Some Math

When nothing is going on, the expected difference in average ranks is 0. The variance of a single draw from a uniform distribution is 1/12, so the variance of the difference is

$$\frac{1}{12} \left( \frac{1}{n_A} + \frac{1}{n_B} \right).$$

Approximate normality kicks in fairly quickly, and for finite samples the shape involves the repeated convolution of uniforms (giving B-splines).
Some Outcomes

So, what do the results of using this test look like?

Looking at the prostate cancer data, the values returned by the relative rank test look intermediate between those of Wilcoxon and t-tests. By using multiple genes to more finely partition the ranks, we recapture some of the parametric sensitivity of the t-test. Here, the data were approximately normal to begin with.
Relative Tails?

Can the relative rank approach be used to help with the tail rank test for biomarkers?

Well, in the description of the tail rank test given earlier, it was stated that we needed to specify two things before using the test:

\[ \psi \], the desired specificity of the biomarker, and

\[ \gamma \], the bound on the FWER.

The way that the relative rank approach can help is hidden in the way the value of \( \psi \) is used.
Defining Quantiles

Specifically, in order to use the tail rank statistic we need to estimate, for each gene $g$, a threshold value $\tau_g$ such that $P(X_g < \tau_g) = \psi$

The difficulty is that $\tau_g$ represents a tail quantile of a distribution, because we want $\psi$ to be close to 1. Tail quantiles are hard to estimate well unless (a) you have lots of samples (which we typically won’t) or (b) you have some knowledge of the parametric form of the distribution of $X_g$. 
Tradeoffs

The question then becomes one of which assumption is more plausible:

that we know a parametric form well enough to characterize tails,

or

that the distribution of expression values in a given intensity range when nothing is going on may be similar enough across genes for them to be productively combined.
The Upside

If we collect the ranks for $G$ genes as above, and focus on the results in the control samples, then our “effective sample size” will increase, typically to the point where we can get point estimates of some extreme quantiles (such as 99%).
Further Extensions

What other ways can we use the relative rank approach?
Further Extensions

What other ways can we use the relative rank approach?

Kruskal-Wallis can be revisited.
Further Extensions

What other ways can we use the relative rank approach?

Kruskal-Wallis can be revisited.

Is there some way to build sensitivity into the tail rank procedure?
Further Extensions

What other ways can we use the relative rank approach?

Kruskal-Wallis can be revisited.

Is there some way to build sensitivity into the tail rank procedure? Probably not, since we’re assuming that the behavior of the biomarker is “atypical” for the subset that it flags as interesting.
Sensitivity and Biomarkers

There is an asymmetry here, which reflects the asymmetry in the question we’re asking.

For good biomarkers, we want the specificity to be high, but we’re willing to live with low sensitivity.

The rationale for this is that the heterogeneity of the disease suggests that if markers are to be productively used, this should be as part of a panel.

We don’t yet know how to assemble a good and parsimonious panel.

We may be able to assemble a good panel.
Putting Some Pieces Together

Let’s examine some aspects of differential expression in R, using some of the datasets from BioConductor.

```r
> library("ALL");
> data("ALL");
```

This is an ExprSet derived from 128 U95Av2 arrays, quantified using RMA. The phenoData has 21 variables, including “mol.biol”. This specifies cytogenetic abnormalities, such as “BCR/ABL” or “NEG”.
Subsetting the Group

```r
> mySubset <- ALL$mol.biol %in% c("BCR/ABL", "NEG");
> ALLs <- ALL[, mySubset];
```

There are 37 samples with the BCR/ABL fusion, and 74 samples that are negative for this.

Let’s contrast these 2 groups.
Looking for Differences

> library("genefilter");
> g <- ALLs$mol.biol; # choose a factor
> ALLs.t <- rowttests(ALLs, g);

The rowttests function is written in C and is pretty fast. For each row, it returns
"statistic" "dm" "df" "p.value"
(dm is the difference in means.)

Unfortunately...
Looking for Differences

> ALLs.t <- rowttests(ALLs, g);
Error in rowttests(ALLs, g) : Number of groups must be <= 2 for 'rowttests'.
> levels(g)
[1] "ALL1/AF4" "BCR/ABL" "E2A/PBX1"
[4] "NEG" "NUP-98" "p15/p16"

Subsetted factors remember where they came from...
Looking for Differences

> ALLs.t <- rowttests(ALLs, g);
Error in rowttests(ALLs, g) : Number of groups must be <= 2 for 'rowttests'.
> levels(g)
[1] "ALL1/AF4"  "BCR/ABL"  "E2A/PBX1"
[4] "NEG"       "NUP-98"    "p15/p16"

Subsetted factors remember where they came from...

> ALLs$mol.biol <- factor(ALLs$mol.biol);
> g <- ALLs$mol.biol;
> ALLs.t <- rowttests(ALLs, g); # works
Are There Differences?

> hist(ALLs.t$p.value, breaks=100);
Ok, Can We See Them?

> heatmap(exprs(ALLs)); # BAD.

Why?
Ok, Can We See Them?

> heatmap(exprs(ALLs)); # BAD.

Why?

We’re considering too many genes at present. (Quick quiz: how many?) Clustering will hang your computer.

We need to filter our list down.
Some Filtering

> meanThresh <- 100;
> filt1 <- rowMeans(exprs(ALLs) [, g ==
+ levels(g)[1]]) > meanThresh;
> filt2 <- rowMeans(exprs(ALLs) [, g ==
+ levels(g)[2]]) > meanThresh;
> selProbes <- (filt1 | filt2);
> ALLfilt <- ALLs[selProbes, ];
> dim(exprs(ALLfilt)); # 3660 by 111, a bit big

> meanThresh <- 200;
...
> dim(exprs(ALLfilt)); # 1771 by 111, better
Focus on the Interesting Ones

```r
> filt3 <- ALLs.t$p.value < - 0.0001;
> selProbes <- (filt1 | filt2) & filt3;
> ALLfilt <- ALLs[selProbes, ];
> dim(exprs(ALLfilt)); # 104 by 111, ok

Try picturing this...

```r
> spcol <- ifelse(ALLfilt$mol.biol == "NEG", "goldenrod", "skyblue")
> heatmap(exprs(ALLfilt), ColSideColors=spcol);
```
Huzzah! (Right?)
That Was Odd...

```r
> plot(exprs(ALLfilt)[2,order(ALLfilt$mol.biol)],
    xaxt='n', xlab='Group');
```
That Was Odd... Right?

> lines(10 - 3.5*(ALLfilt$mol.biologic[order(ALLfilt$mol.biologic)] == 'NEG'), col='red');
> title(main = 'BCR/ABL, followed by NEG');
What’s Going On?

> names(pData(ALLfilt))

[1] "cod"             "diagnosis" "sex"          "age"
[5] "BT"             "remission" "CR"           "date.cr"
[9] "t(4;11)"        "t(9;22)"   "cyto.norm" "citog"
[13] "mol.biol"      "fus prot"  "mdr"          "kinet"
[17] "ccr"           "relapse"   "transplant" "f.u"
[21] "date last seen"

Are there other variables that may dominate the one I chose?
What Cells?

```r
> ALLfilt$BT
[41] B2 B2 B3 B3 B3 B3 B4 B3 B3 B1
[51] B1 B3 B3 B3 B3 B3 B3 B3 B3 B3
[81] T2 T2 T3 T2 T T4 T2 T3 T3 T
[91] T2 T3 T2 T2 T2 T1 T4 T T2 T3
[101] T2 T2 T2 T2 T3 T3 T3 T2 T3 T2
[111] T
```

Levels: B B1 B2 B3 B4 T T1 T2 T3 T4
Matching Patterns

We want entries that begin with B. This is a “regular expression”, and one of the tools for extracting these is “grep”.

> BT <- as.character(ALLfilt$BT);
> grep("B", BT); # returns 1..79
> grep("^B", BT); # same
> grep("^T", BT); # 80..111
> grep("B*", BT); # 1..111
> grep("B.*", BT); # 1..79
> grep("B$", BT); # 14,26,27,79
> grep("^B$", BT); # same
> grep("^b", BT); # null
> grep("^b", BT, ignore.case=TRUE);
Once More Unto the Breach!

> plot(exprs(ALLfilt)[2,, xlab='Sample']);
> y1 <- rep(0,111);
> y1[grep("^T", zed2)] = 1;
> lines(10 - 3.5*y1, col='red')
> title(main="B Cells, then T Cells");
Finally!

![Graph showing expression levels of B Cells and T Cells over sample numbers](image-url)
Analysis Redux 1

> mySubset1 <- grep("^B", ALL$BT);
> ALLs1 <- ALL[,mySubset1];
> dim(exprs(ALLs1))
[1] 12625 0

erm, oops?
Analysis Redux 1

> mySubset1 <- grep("^B", ALL$BT);
> ALLs1 <- ALL[,mySubset1];
> dim(exprs(ALLs1))
[1] 12625 0

erm, oops?

> mySubset1 <- grep("^B", as.character(ALL$BT));
> ALLs1 <- ALL[,mySubset1];
> dim(exprs(ALLs1))
[1] 12625 95
> mySubset2 <- ALLs1$mol.biol %in% c("BCR/ABL", "NEG")
> ALLs2 <- ALLs1[,mySubset2];
> dim(exprs(ALLs2))
[1] 12625 79
> ALLs2$mol.biol <- factor(ALLs2$mol.biol);
> g <- ALLs2$mol.biol;
> ALLs2.t <- rowttests(exprs(ALLs2), g);
> hist(ALLs2.t$p.value, breaks=100);
Analysis Redux 3

```r
> meanThresh <- 200;
> filt1 <- rowMeans(exprs(ALLs2)[, g == levels(g)[1]]) > meanThresh;
> filt2 <- rowMeans(exprs(ALLs2)[, g == levels(g)[2]]) > meanThresh;
> filt3 <- ALLs2$t$p.value < 0.0001;
> selProbes <- (filt1 | filt2) & filt3;
> ALLs2Filt <- ALLs2[selProbes,];
> dim(exprs(ALLs2Filt))
[1] 0 79
```

Hmm.
Analysis Redux 4

```r
> meanThresh <- 100;
> filt1 <- rowMeans(exprs(ALLs2)[, g == levels(g)[1]]) > meanThresh;
> filt2 <- rowMeans(exprs(ALLs2)[, g == levels(g)[2]]) > meanThresh;
> filt3 <- ALLs2.t$p.value < 0.001;
> selProbes <- (filt1 | filt2) & filt3;
> ALLs2Filt <- ALLs2[selProbes,];
> dim(exprs(ALLs2Filt))
[1] 0 79
```

ok, maybe that wasn’t it...
Analysis Redux 5

```r
> filt1[1:5]
 1000_at 1001_at 1002_f_at 1003_s_at 1004_at
 FALSE FALSE FALSE FALSE FALSE
> plot(filt1)  # all FALSE
> levels(g)
[1] "BCR/ABL" "NEG"
> plot(rowMeans(exprs(ALLs2)))
```
Analysis Redux 5 – AHA!
> meanThresh <- log2(100);
> filt1 <- rowMeans(exprs(ALLs2)[, g == levels(g)[1]]) > meanThresh;
> filt2 <- rowMeans(exprs(ALLs2)[, g == levels(g)[2]]) > meanThresh;
> filt3 <- ALLs2.t$p.value < 0.0001;
> selProbes <- (filt1 | filt2) & filt3;
> ALLs2Filt <- ALLs2[selProbes,];
> dim(exprs(ALLs2Filt)) # better
[1] 36 79
A Better Figure

```r
> spcol <- ifelse(ALLs2Filt$mol.biol == "NEG", "goldenrod", "skyblue");
> heatmap(exprs(ALLs2Filt), ColSideColors=spcol);
```
So, What About These Genes?

Through all this processing, the gene identities have been preserved, so we can access them easily.

```r
> geneNames(ALLs2Filt)[1:3]
[1] "106_at" "1134_at" "1635_at"

> ALLs2Filt.t <- rowttests(exprs(ALLs2Filt), g);
> plot(ALLs2Filt.t$statistic)
> index <- order(abs(ALLs2Filt.t$statistic), decreasing = TRUE);
> probeids <- geneNames(ALLs2Filt)[index]
> probeids[1:3]
[1] "1636_g_at" "39730_at" "1635_at"
```
Let's Add to the Report

Grab some of the annotation from the environment

```r
> library("annaffy")
Loading required package: GO
Loading required package: KEGG
> library("hgu95av2")
> syms <- unlist(mget(probeids, hgu95av2SYMBOL))
> locuslinks <- unlist(mget(probeids, hgu95av2LOCUSID))
> library("annotate")
```
Make a Web Page

```r
> geneList <- list(probeids);
> repository <- list("affy");
> otherNames <- list(syms, locuslinks);
> head <- c("Probe ID", "Symbol", "LocusLink");
> fileName <- "out1.html";
> htmlpage(genelist = geneList, filename = fileName, title="ALL Interesting",
othernames = otherNames,
table.head = head,
repository = repository)
```

Note: the elements in geneList will be live links!
The Output

Differentially Expressed Genes

ALL Interesting

| 1636 g at | ABL1 | 25 |
| 39730 at | ABL1 | 25 |
| 1635 at  | ABL1 | 25 |
| 40202 at | KLF9  | 687 |
| 37027 at | AHNAK | 79026 |
| 39837 s at | ZNF467 | 168544 |
| 40480 s at | FYN  | 2534 |
| 33774 at | CASP8 | 841 |
| 36591 at | TUBA1 | 7277 |
| 37014 at | MX1   | 4599 |
| 39338    | LSTN1 | 187 |
More Details

The Affy links take you to NetAffx; you must log in there to see the details.

For htmlpage, values to be linked must come first.

Calling this function without a list of values to be linked (ie, all columns are “othernames”) will break.

More than one set of links can be included in a single table; every such set requires us to specify a repository.

The documentation for htmlpage doesn’t directly state what the set of repositories is...
Checking Code

> htmlpage
... 
  for (i in seq(along = repository)) {
    rows <- paste(rows,
                 getTDRows(genelist[[i]],
                            repository[[i]]))
  }
...

checking “getTDRows”:

repository: A character string for the name of a public repository. Valid values include "ll", "ug", "gb", "sp", "omim", "affy", "en", and "fb".
What are These?

• LL: LocusLink
• UG: UniGene
• GB: GenBank
• SP: SwissProt
• OMIM: Online Mendelian Inheritance in Man
• AFFY: NetAffx
• EN: EntrezGene (replaces LocusLink)
• FB: FlyBase
Getting More Sophisticated

The annaffy package contains quite a few more accessors for different types of databases and information. These begin with “aaf” (Annotation for AFFy), and return lists of information.

The list (all prefixed by aaf): ChromLoc, Chromosome, Cytoband, Description, Function, GenBank, GO, LocusLink, Pathway, Probe, PubMed, Symbol, UniGene.
Mapping

Most queries share a common syntax:

> ALLbands <- aafCytoband(probeids, "hgu95av2");
> ALLbandLinks <- getURL(ALLbands);
> ALLbandLinks[[1]]


> ALLbands[[1]]

An object of class "aafCytoband"
Slot "band":
[1] "9q34.1"
Slot "genbank":
[1] "U07563"
Assembling a Table, Take 2

```r
> aaf.handler()
[1] "Probe"   "Symbol"   "Description"
[4] "Function" "Chromosome" "Chromosome Location"
[7] "GenBank"  "LocusLink"  "Cytoband"
[10] "UniGene"  "PubMed"    "Gene Ontology"
[13] "Pathway"
>
> ALLTable <- aafTableAnn(probeids, "hgu95av2");
```

the argument “colnames” can be used to produce subsets of this output, or to rearrange the order.
Assembling a Table, Take 2

> ALLCols <- aaf.handler()[c(1:3, 9, 5:6)];
> ALLTable2 <- aafTableAnn(probeids, "hgu95av2", colnames=ALLCols);

These approaches produce tables which are lists of lists:

> ALLTable[[1]][[1]]
[1] "1636_g_at"
attr("class")
[1] "aafProbe"
> ALLTable[[2]][[1]]
[1] "ABL1"
attr("class")
[1] "aafSymbol"
Making a Page, Take 2

> saveHTML(ALLTable, "out2.html")
> saveHTML(ALLTable2, "out3.html")

Why do it twice?
# INTRODUCTION TO MICROARRAYS

## Bioconductor Affymetrix Probe Listing

<table>
<thead>
<tr>
<th>Probe</th>
<th>Symbol</th>
<th>Description</th>
<th>Function</th>
<th>Chromosome</th>
<th>Chromosome Location</th>
<th>GenBank</th>
<th>LocusLink</th>
<th>Cytoband</th>
</tr>
</thead>
<tbody>
<tr>
<td>1636 g at</td>
<td>ABL1</td>
<td>v-abl Abelson murine leukemia viral oncogene homolog 1</td>
<td>9</td>
<td>130740384, 130618821</td>
<td></td>
<td>U07563</td>
<td>25</td>
<td>9q34.1</td>
</tr>
<tr>
<td>Probe</td>
<td>Symbol</td>
<td>Description</td>
<td>Cytoband</td>
<td>Chromosome</td>
<td>Chromosome Location</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>--------</td>
<td>--------</td>
<td>-------------------------------------------------------</td>
<td>----------</td>
<td>------------</td>
<td>---------------------------</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1636 g at</td>
<td>ABL1</td>
<td>v-abl Abelson murine leukemia viral oncogene homolog 1</td>
<td>9q34.1</td>
<td>9</td>
<td>130740384, 130618821</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>39730 at</td>
<td>ABL1</td>
<td>v-abl Abelson murine leukemia viral oncogene homolog 1</td>
<td>9q34.1</td>
<td>9</td>
<td>130740384, 130618821</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1635 at</td>
<td>ABL1</td>
<td>v-abl Abelson murine leukemia viral oncogene homolog 1</td>
<td>9q34.1</td>
<td>9</td>
<td>130740384, 130618821</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>40202 at</td>
<td>KLF9</td>
<td>Kruppel-like factor 9</td>
<td>9q13</td>
<td>9</td>
<td>-70229068</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>37027 at</td>
<td>AHNAK</td>
<td>AHNAK nucleoprotein (desmoyokin)</td>
<td>11q12.2</td>
<td>11</td>
<td>-62039950, -61957591</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>39837 s at</td>
<td>ZNF467</td>
<td>zinc finger protein 467</td>
<td>7q36.1</td>
<td>7</td>
<td>-148899099</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>40480 s at</td>
<td>FYN</td>
<td>FYN oncogene related to SRC, FGR, YES</td>
<td>6q21</td>
<td>6</td>
<td>-112089179, -112089186</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>33774 at</td>
<td>CASP8</td>
<td>caspase 8, apoptosis-related cysteine peptidase</td>
<td>2q33-q34</td>
<td>2</td>
<td>201923686, 201948284, 201950747, 201923693</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>36591 at</td>
<td>TUBA1</td>
<td>tubulin, alpha 1 (testis specific)</td>
<td>2q35</td>
<td>2</td>
<td>-219940505</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>37014 at</td>
<td>MX1</td>
<td>myxovirus (influenza virus) resistance 1, interferon-inducible protein p78 (mouse)</td>
<td>21q22.3</td>
<td>21</td>
<td>41720023</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>39329 at</td>
<td>ACTN1</td>
<td>actinin, alpha 1</td>
<td>14q24.1-q24.2</td>
<td>14q24.1</td>
<td>14q22-q24</td>
<td>14</td>
<td>-68410792</td>
<td></td>
</tr>
<tr>
<td>32542 at</td>
<td>FHL1</td>
<td>four and a half LIM domains 1</td>
<td>Xq25</td>
<td>X</td>
<td>134955199</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>40051 at</td>
<td>TRAM2</td>
<td>translocation associated membrane protein 2</td>
<td>6p21.1-p12</td>
<td>6</td>
<td>-52470160</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>