Analysis of Microarray Data

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

10 November 2004
Web site

http://bioinformatics.mdanderson.org

- /MicroarrayCourse
- /Workshop
- /software.html
- /tutorial.html
Themes

- There are lots of good methods available to analyze microarrays.
- Simulations help assess method characteristics.
- Permutations help assess the significance or believability of results.
- Take nothing for granted.
Analysis of Microarray Data

- Statistical Tests of Differential Expression
- Rank-Based Tests of Differential Expression
- Clustering Techniques
- Clustering and Project Normal
Clustering Techniques

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

10 November 2004
Clustering Techniques

- Measuring distances
- Hierarchical clustering
- When is a cluster valid?
- Clustering with fewer genes
- Simulating something
Measuring distances

Ideally, clustering methods tell us that some samples form a more coherent set than the data as a whole, where “more coherent” is generally taken to mean that the samples are closer together.

So, how do we define “closer”?

This requires the specification of a distance or “dissimilarity” matrix. Distances are calculated between each pair of samples. For this purpose, we view each sample as a vector in “gene-space”. The first distance measure most people think of is Euclidean distance:

$$\text{sqrt} \left( \sum ((x - y)^2) \right)$$

```r
dEuclid <- dist(t(dataMatrix));
```

© Copyright 2004, Kevin R. Coombes
Alternative definitions of distance

Maximum: \( \text{abs}(\max(x-y)) \)

Manhattan: \( \text{sum}(\text{abs}(x-y)) \)

Canberra: \( \text{sum}(\text{abs}(x-y)/(\text{abs}(x)+\text{abs}(y))) \)

Binary: \( \text{sum}(\text{xor}(x!={}0, y!={}0))/\text{sum}(x!={}0 \mid y!={}0) \)

Minkowski: \( \text{sum}(\text{abs}(x-y)^p)^{1/p} \)

Correlation: \( (1 - \text{cor}(x,y))/2 \)

Absolute Correlation: \( (1 - \text{abs}(\text{cor}(x,y))) \)

Rank Correlation: \( (1 - \text{cor}(\text{rank}(x), \text{rank}(y)))/2 \)

Most clustering methods let you specify the distance measure, or construct any distance matrix you want and work with that matrix.
Hierarchical clustering

Hierarchical clustering produces a dendrogram (a binary tree structure) that displays the distance relationships between clusters.

The most common implementation is agglomerative, which is an unnecessarily big word for bottom-up. The algorithm starts by joining the two samples that are closest together into a cluster. It then keeps repeating this process (joining the two closest clusters into a bigger cluster) until everything has been linked together.

There’s only one problem: Distances were defined between individual vectors. How do you measure the distance between clusters of vectors in order to link them?
Linkage rules

**Single:** Use the minimum distance between cluster members

**Complete:** Use the maximum distance between cluster members

**Average:** Use the mean distance between cluster members

**Median:** Use the median distance between cluster members

**Centroid:** Use the distance between the “average” member of each cluster
**Simulating nothing**

One peculiarity of clustering algorithms is that they always produce clusters. This happens regardless of whether there is actually any meaningful clustering structure present in the data. So, let’s simulate some unstructured data an see what happens.

```r
> n.genes <- 1000
> n.samples <- 50
> descr <- paste('S', rep(c('0',''), times=c(9,41)), 1:50, sep='')
> dataMatrix <- matrix(rnorm(n.genes*n.samples),
+   nrow=n.genes)
```

© Copyright 2004, Kevin R. Coombes
Clustering nothing

```r
> dEuc <- dist(t(dataMatrix))
> hAvgEuc <- hclust(dEuc, method='average')
> plclust(hAvgEuc, labels=descr)
```
Single linkage often produces “stringlike” clusters

> hSinEuc <- hclust(dEuc, method='single')
> plclust(hSinEuc, labels=descr)
Complete linkage tends to find compact clusters

> hComEuc <- hclust(dEuc, method='complete')
> plclust(hComEuc, labels=descr)

Average linkage tends to produce clusters somewhere in between single and complete linkage.
Clustering with correlation also finds structure

> dCor <- as.dist( (1-cor(dataMatrix))/2 )
> hComCor <- hclust(dCor, method='complete')
> plclust(hComCor, labels=descr)
What is a cluster?

If we cut the dendrogram at height $h$, then the sub-branches of each cut branch define clusters. Within a cluster, everything is closer than $h$ to the rest. By varying the cut height, we can produce an arbitrary number of clusters.
What is a cluster?

If we cut the dendrogram at height $h$, then the sub-branches of each cut branch define clusters. Within a cluster, everything is closer than $h$ to the rest. By varying the cut height, we can produce an arbitrary number of clusters.
When is a cluster valid?

In other words, where should we cut the tree in order to say that the branches at this point represent something real?

To convince you that this is a real problem, recall that we are using data that was simulated to be completely random. Nevertheless, hierarchical clustering (with complete linkage and either Euclidean distance or correlation) apparently finds structure here.
Bootstrap resampling

Testing cluster validity requires “perturbing” the data.

A cluster consists of pairs of items that are grouped together. If we repeatedly perturb the data, and the pairs still cluster together, this is a good sign that the cluster is “stable”. Samples that cluster in other groups are more questionable.

The simplest way to perturb the data is to “bootstrap” the individual genes, or rows of the data matrix.

```r
> nGenes <- nrow(dataMatrix)
> tempData <- dataMatrix[sample(nGenes, + replace=TRUE),]
> tempCorr <- as.dist((1 - cor(tempData))/2)
> tempCorrTree <- hclust(tempCorr,
+   method = 'average');
```
Disturbing the universe

Using 3 groups, 50 samples, and an outer loop of 1000 bootstrap samples:

```r
tempMatch <- matrix(0, nrow=nSam, ncol=nSam);
tempCut <- cutree(tempCorrTree, k = nGroups);
for(i2 in 1:nGroups){
    tempMatch[tempCut==i2,tempCut==i2] = 1;
}
bootMatch <- bootMatch + tempMatch;
```
Sometimes it’s good to find nothing...
Additional Notes

We need to specify the number of clusters to bootstrap, since we record how many times samples are paired. This method extends directly to other clustering techniques.

The image is much more interpretable if the rows and columns of the matching matrix are reordered to match the ordering supplied by the clustering.

Instead of resampling the genes, we can “add noise” to the data from a normal distribution. The scale of noise to use is not obvious with real data.

We can also use “bootstrap subsampling”. Instead of reconstructing a sample of the same size as the number of genes on the array, make smaller samples to see how widely the clustering information is spread across the genes.
Clustering with fewer genes

Many times, we cluster using a subset of genes. Maybe we think that other genes are just contributing noise, or maybe we want to focus on genes on a specific chromosome or genes in a specific pathway.

Occasionally, you see papers comparing two known classes of samples that perform the following analysis:

1. Find a list of differentially expressed genes.

2. Cluster the data using only the differentially expressed genes.

3. Discover that you can successfully distinguish the known classes.

Should this be surprising?
Bending reality to your will

Let’s try this on our simulated data. We’ll divide the 50 samples into two classes (the first 25 and the last 25). Next, we’ll perform t-tests to see how well each gene separates the two classes, and cluster the data using the top 50 genes:
Even the bootstrap doesn’t save us...
Filtering notes

Filters should not be related to a specific contrast if an overall view is desired.

More natural filters exist:

- total variation
- all genes on a given chromosome
- all genes in a given ontology category

Filtering serves a practical purpose – it reduces the number of genes a lot. This is important because we may want to cluster the genes as well as the samples, and clustering thousands of things may make R complain.
Simulating something

Next, we simulated data with 1000 genes and 5 different sample classes containing different numbers of samples. Here's a two-dimensional picture of the truth:
Hierarchical clusters (correlation; average)

Three of the classes (B, C, D) are mostly correct. The other two classes are less concentrated.
Bootstrap clusters
Bootstrap clusters ordered by true groups
Review

1. Hierarchical clustering always finds clusters.

2. Bootstrap resampling can show that the clusters are fake.

3. Filtering to show a particular grouping can lead to incorrect results.

4. Hierarchical clustering with bootstrap validation can partially uncover real structure.
Where do we go next?

1. Are there other methods that do a better job of finding the real structure in the simulated data?

2. How do clustering methods perform on real microarray data?
Clustering and Project Normal

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

10 November 2004
Clustering and Project Normal

- Project Normal
- Project Normal and Hierarchical Clustering
- K-Means
- Partitioning around medoids (PAM)
- Silhouette width
- Abnormal Behavior
- Problems and Solution
Project Normal

- Eighteen samples
  - Six C57BL6 male mice
  - Three organs: kidney, liver, testis

- Reference material
  - Pool RNA from all eighteen mouse organs

- Replicate experiments on two-color arrays with common reference
  - Four experiments per mouse organ
  - Dye swaps: two red samples, two green samples
Original analysis of Project Normal


- Print-tip specific intensity dependent loess normalization
- Scale adjusted (using MAD)
- Work with log ratios (experimental/reference)
- Perform F-test for each gene to see if mouse-to-mouse variance exceeds the array-to-array variance.
First steps

We chose to process the data using a simple global normalization (to the $75^{th}$ percentile) instead of loess normalization, since we believed that the mixed reference RNA should have a different distribution of intensities than RNA from a single organ. We then transformed the intensities in each channel by computing their base-two logarithm.

**Main Question:** Can we determine from the project normal data set which genes are specifically expressed in each of the three organs?
Clustering Methods

If we cluster the data, what should we expect to see?

Which clustering method would be most appropriate for a first look at the data?

- Hierarchical clustering
- Partitioning around medoids
- K-means
- Principal components analysis
Hierarchical clustering

Euclidean distance, average linkage
Hierarchical clustering

Correlation distance, average linkage
Hierarchical clustering with bootstrap validation

Back to clustering methods
K-Means Clustering

Input: A data matrix, $X$, and the desired number of clusters, $K$.

Output: For each sample $i$, a cluster assignment $C(i) \leq K$.

Idea: Minimize the within-cluster sum of squares

$$\sum_{c=1}^{K} \sum_{C(i)=c, C(j)=c}^{N} \sum_{\ell=1}^{N} (x_{i\ell} - x_{j\ell})^2$$

- Algorithm:
  1. Make an initial guess at the centers of the clusters.
  2. For each data point, find the closest cluster (Euclidean).
  3. Replace each cluster center by averaging data points that are closest to it.
  4. Repeat until the assignments stop changing.
Last time, we simulated data with 1000 genes and 5 different sample classes.
K-Means, Take 1

Perfect clustering! (Circles = starting group centers, X = final group centers)
K-Means, Take 2

Oops: bad starting points may mean bad clusters!
K-Means, Take 3
Local minima may not be global

K-means can be very sensitive to the choice of centers used as seeds for the algorithm. The problem is that the algorithm only converges to a local minimum for the within-cluster sum of squares, and different runs with randomly chosen centers (which is the default in the \texttt{kmeans} function in R) can converge to different local optima. You can see which of these three runs is better:

```r
> sum(kres1$withinss)
[1] 25706.57
> sum(kres2$withinss)
[1] 25736.84
> sum(kres3$withinss)
[1] 25926.12
```
Local minima may not be global

There are two ways around the fact that local minima need not be global.

One is to find better starting seeds for the algorithm. For example, start with hierarchical clustering. Then cut the tree into five branches, and use the average of each branch as the starting points.

Alternatively, you can run the algorithm with many random seeds, keeping track of the within-cluster sum of squares.
Multiple runs of the K-means algorithm

kcent <- sample(n.samples, 5)
kres <- kmeans(t(ldata), t(ldata[,kcent]))
withinss <- sum(kres$withinss)
for (i in 1:100) {
tcent <- sample(n.samples, 5)
tres <- kmeans(t(ldata), t(ldata[,tcent]))
print(sum(tres$withinss))
if (sum(tres$withinss) < withinss) {
kres <- tres
kcent <- tcent
withinss <- sum(kres$withinss)
}
}
K-means for Project Normal

Number of channels in each cluster:

<table>
<thead>
<tr>
<th>Channel</th>
<th>C1</th>
<th>C2</th>
<th>C3</th>
<th>C4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Experiment</td>
<td>4</td>
<td>24</td>
<td>20</td>
<td>24</td>
</tr>
<tr>
<td>Reference</td>
<td>28</td>
<td>0</td>
<td>44</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Organ</th>
<th>C1</th>
<th>C2</th>
<th>C3</th>
<th>C4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Kidney</td>
<td>24</td>
<td>24</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Liver</td>
<td>0</td>
<td>0</td>
<td>24</td>
<td>24</td>
</tr>
<tr>
<td>Testis</td>
<td>8</td>
<td>0</td>
<td>40</td>
<td>0</td>
</tr>
</tbody>
</table>

Best of 50 runs with four clusters
K-means

Number of channels in each cluster:

<table>
<thead>
<tr>
<th>Channel</th>
<th>C1</th>
<th>C2</th>
<th>C3</th>
<th>C4</th>
<th>C5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Experiment Reference</td>
<td>4</td>
<td>16</td>
<td>20</td>
<td>8</td>
<td>24</td>
</tr>
<tr>
<td>20</td>
<td>0</td>
<td>44</td>
<td>8</td>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Organ</th>
<th>C1</th>
<th>C2</th>
<th>C3</th>
<th>C4</th>
<th>C5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Kidney</td>
<td>16</td>
<td>16</td>
<td>0</td>
<td>16</td>
<td>0</td>
</tr>
<tr>
<td>Liver</td>
<td>0</td>
<td>0</td>
<td>24</td>
<td>0</td>
<td>24</td>
</tr>
<tr>
<td>Testis</td>
<td>8</td>
<td>0</td>
<td>40</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Best of 50 runs with five clusters
K-means with bootstrap validation

Back to clustering methods
Partitioning Around Medoids (PAM)

The K-means clustering algorithm has another limitation. (This is not the last one we will consider).

As described, it always uses Euclidean distance as the measure of dissimilarities between sample vectors. As we saw last time with hierarchical clustering, there are a large number of possible distances that we might want to use. Fortunately, a simple adjustment to the algorithm lets us work with any distance measure.
**PAM algorithm**

Input: Data matrix, $X$, distance $d$, number of clusters, $K$.

Output: For each sample $i$, a cluster assignment $C(i) \leq K$.

Idea: Minimize the within-cluster distance

$$
\sum_{c=1}^{K} \sum_{C(i)=c, C(j)=c} d(x_i, x_j)
$$

- Algorithm:
  1. Make an initial guess at the centers of the clusters.
  2. For each data point, find the closest cluster.
  3. Replace each cluster center by the data point minimizing the total distance to other members in its cluster.
  4. Repeat until the assignments stop changing.
PAM in R

To use PAM in R, you must load another package:

```r
> require(cluster)
> dist.matrix <- as.dist(1-cor(ldata)/2)
> pamres <- pam(dist.matrix, 5)
```

Unlike \texttt{kmeans}, the implementation of \texttt{pam} only lets you specify the number of clusters you want, not the starting point. It also apparently always uses the same method to choose the starting point, so it does not help to run the algorithm multiple times. If their heuristic chooses a poor starting configuration, there is no way to fix it.
PAM results on simulated data

Not very good on our example data...
How many clusters are there?

Both \texttt{kmeans} and \texttt{pam} require you to specify the number of clusters before running the algorithm. In the simulated example, we knew before we stated that there were five clusters. In real life, we rarely (if ever) know the number of real clusters before we start. How do we figure out the correct number?

One way is to run the algorithm with different values of $K$, and then try to decide which methods gives the best results. The problem that remains is how we measure “best”.

Silhouette Widths

Kaufman and Rousseeuw (who wrote a book on clustering that describes pam along with quite a few other methods) recommend using the **silhouette width** as a measure of how much individual elements belong to the cluster where they are assigned. To compute the silhouette width of the $i^{th}$ object, define

$$a(i) = \text{average distance to other elements in the cluster}$$

$$b(i) = \text{smallest average distance to other clusters}$$

$$sil(i) = (b(i) - a(i))/\max(a(i), b(i)).$$

Interpretation: If $sil(i)$ is near 1, then the object is well clustered. If $sil(i) < 0$, then the object is probably in the wrong cluster. If $sil(i)$ is near 0, then it’s on the border between two clusters.
PAM: two clusters

```r
> pam2 <- pam(dmat, 2)
> plot(pam2)
```

Silhouette plot of pam(x = dmat, k = 2)

- 2 clusters $C_j$
  - $j : \{ n_j \mid \text{ave}_{i \in C_j} s_i \}$
  - 1: 29 | 0.027
  - 2: 24 | 0.011

Average silhouette width: 0.02
PAM : three clusters

> pam3 <- pam(dmat, 3)
> plot(pam3)

Silhouette plot of pam(x = dmat, k = 3)

n = 53

3 clusters C_i
j: n_j \{ \text{ave}_{i \in C_j} s_i \}

1: 23 | 0.039

2: 20 | 0.0027

3: 10 | 0.049

Average silhouette width: 0.03
PAM : four clusters

```r
> pam4 <- pam(dmat, 4)
> plot(pam4)
```

![Silhouette plot of pam(x = dmat, k = 4)](image)

Silhouette plot of pam(x = dmat, k = 4)

- 4 clusters $C_i$
- $j$: $n_j \mid \text{ave}_{i \in C_j} s_i$

- 1: 19 | 0.037
- 2: 17 | 0.0017
- 3: 8  | 0.031
- 4: 9  | 0.063

Average silhouette width: 0.03
PAM : five clusters

> pam5 <- pam(dmat, 5)
> plot(pam5)
PAM : six clusters

> pam6 <- pam(dmat, 6)
> plot(pam6)
PAM : seven clusters

> pam7 <- pam(dmat, 7)
> plot(pam7)
In general, we want to choose the number of clusters that maximizes the average silhouette width. In this case, that means 4 or 5 clusters.
Correlation distance, four clusters
Correlation distance, five clusters
PAM results on Project Normal

Silhouette plot of pam(x = first.cor, k = 6)

Correlation distance, six clusters

Back to clustering methods
Principal Components

You may have wondered how I produced two-dimensional plots of the simulated data that involved 1000 genes and 53 samples. The short answer is: I used principal components analysis (PCA).

As we have been doing throughout our discussion of clustering, we view each sample as a vector $x = (x_1, \ldots, x_G)$ in $G$-dimensional “gene space”. The idea behind PCA is to look for a direction (represented as a linear combination $u_1 = \sum_{i=1}^{G} w_i x_i$) that maximizes the variability across the samples. Next, we find a second direction $u_2$ at right angles to the first that maximizes what remains of the variability. We keep repeating this process. The $u_i$ vectors are the principal components, and we can rewrite each sample vector as a sum of principal components instead of as a sum of separate gene expression values.
Data reduction

PCA can be used as a data reduction method. Changing from the original $x$-coordinates to the new $u$-coordinate system doesn’t change the underlying structure of the sample vectors. However, it does let us focus on the directions where the data changes most rapidly. If we just use the first two or three principal components, we can produce plots that show us as much of the intrinsic variability in the data as possible.
Sample PCA

We have written a version of PCA in R using SVD. The first plot of our simulated data was produced using the following commands:

\[
\text{> spca <- sample.pc(ldata)}
\]
\[
\text{> plot(spca, split=group.factor)}
\]

The plots that added the X’s to mark the K-means centers were produced with:

\[
\text{> plot(spca, split=factor(kres$cluster))}
\]
\[
\text{> x1 <- spca@scores[kcent,1] # start circles}
\]
\[
\text{> x2 <- spca@scores[kcent,2]}
\]
\[
\text{> points(x1, x2, col=2:6, pch=1, cex=2)}
\]
\[
\text{> pcak <- predict(spca, t(kres$centers)) # finish X}
\]
\[
\text{> points(pckak[,1], pcak[,2], col=2:6, pch=4)}
\]
PCA for Project Normal

Euclidean distance, indicating channel and organ.

Back to clustering methods    Forward to second PCA
Abnormal Behavior

Regardless of which exploratory method we use to look at the data, we see that something strange is happening here.

We might not have noticed this behavior if we had immediately gone to the log ratios instead of clustering the separate channels.

What might explain the presence of two different kinds of reference channels? First thought: dye swaps. But this doesn’t make sense, since then we would expect the experimental channels to split the same way (giving us eight clusters in total).
Data merging

• Data was supplied in three files, one each for kidney, liver and testis.

• Each row in each file contained two kinds of annotations:
  1. Location (block, row, and column)
  2. Genetic material (IMAGE clone, UniGene ID)

• For our analysis, we merged the data using the annotations of genetic material.

• As it turns out, the locations did not agree

• So, we reordered data rows and merged on location...
PCA after merging data on location

Yuck. So why are most of the testis references so weird?

Back to first PCA  Forward to third PCA
Inspired guessing

- When the gene annotations are matched
  - Four of the testis reference channels are close to the kidney reference
  - Twenty of the testis reference are close to the liver reference

- When the location annotations are matched
  - Kidney, liver, and 4 testis references are close
  - The other 20 testis reference are off by themselves

- Conclusion: A data processing error occurred partway through the testis experiments.
Principal components, take 3

Finally, the picture we expected to start with!

Back to second PCA
Every solution creates a new problem

- **Solution:** After reordering all liver experiments and twenty testis experiments by location
  - Can distinguish between the three organs
  - The reference samples all cluster together

- **New Problem:** There are now two competing ways to map from locations to genetic annotations (one from the kidney data, one from the liver data). Which one is correct?
How big is the problem?

- Microarray contains 5304 spots
- Only 3372 (63.6%) spots have UniGene annotations that are consistent across the files
- So, 1932 (36.4%) spots have ambiguous UniGene annotations
UniGene Example

UniGene Cluster Mm.4010 Mus musculus

Vil Villin

SEE ALSO

LocusLink: 22349
Mouse Genome Informatics: MGI:98930
HomoloGene: Mm.4010

SELECTED MODEL ORGANISM PROTEIN SIMILARITIES

organism, protein and percent identity and length of aligned region

H. sapiens: sp:P09327 - 89% / 826 aa
VILL_HUMAN Villin 1 (see ProtEST)

M. musculus: sp:Q62468 - 100% / 826 aa
VILL_MOUSE Villin 1 (see ProtEST)

R. norvegicus: ref:NP_077377.1 - 59% / 810 aa
nervin [Rattus] (see ProtEST)
### UniGene Example

<table>
<thead>
<tr>
<th>UniGene ID</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BC015267</td>
<td>Mus musculus, villin, clone MGC:18506, IMAGE:4236751, mRNA, complete cds</td>
</tr>
<tr>
<td>NM_009509</td>
<td>Mus musculus villin (Vil), mRNA</td>
</tr>
<tr>
<td>M98454</td>
<td>Mus musculus villin protein mRNA, complete cds</td>
</tr>
</tbody>
</table>

#### Expression Information
- **cDNA sources:** kidney, colon, cecum, tumor, metastatic to mammary, pooled organs, egg, embryonic body between diaphragm region and neck, in vitro fertilized eggs, pancreas, intestinal mucosa, bowel, skin, whole embryo including extraembryonic tissues at 7.5-days postcoitum, embryo.
Villin Expression

Comparison of Villin expression levels across different tissues using box plots for two spots:

Spot (2, 17, 5) and Spot (4, 17, 5).

Log Intensity

Kidney  Liver  Testis  Kidney  Liver  Testis
Definition of abundance

- If the UniGene database entry for “gene expression” says that the cDNA sources of the clones found in a cluster included “kidney”, then we will say that the gene is abundant in kidney.

- Analogous definitions obviously apply for liver, testis, or other organs.
# Abundance by consistency

<table>
<thead>
<tr>
<th>Abundance</th>
<th>All UniGene</th>
<th>Consistent</th>
<th>Ambiguous</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td>409</td>
<td>237</td>
<td>172</td>
</tr>
<tr>
<td>Kidney</td>
<td>129</td>
<td>76</td>
<td>53</td>
</tr>
<tr>
<td>Liver</td>
<td>284</td>
<td>169</td>
<td>115</td>
</tr>
<tr>
<td>Testis</td>
<td>372</td>
<td>231</td>
<td>141</td>
</tr>
<tr>
<td>Kidney, Liver</td>
<td>126</td>
<td>69</td>
<td>57</td>
</tr>
<tr>
<td>Kidney, Testis</td>
<td>226</td>
<td>146</td>
<td>80</td>
</tr>
<tr>
<td>Liver, Testis</td>
<td>960</td>
<td>609</td>
<td>351</td>
</tr>
<tr>
<td>All</td>
<td>2789</td>
<td>1835</td>
<td>963</td>
</tr>
</tbody>
</table>

© Copyright 2004, Kevin R. Coombes

CAMDA Conference 2004
Combining UniGene abundance with microarray data

• For each gene
  • Let $I = (K, L, T)$ be the binary vector of its abundance in three organs as recorded in the UniGene database.
  • Let $Y = (k, l, t)$ be the measured log intensity in the three organs.

• Model using a 3-dimensional multivariate normal distribution

$$Y | I \sim N_3(\mu_I, \Sigma_I)$$

• Average replicate experiments from same mouse with same dye to produce natural triplets of measurements.
Use consistently annotated genes to fit the model

<table>
<thead>
<tr>
<th>Abundance</th>
<th>$\mu_K$</th>
<th>$\mu_L$</th>
<th>$\mu_T$</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td>2.027</td>
<td>2.129</td>
<td>2.012</td>
</tr>
<tr>
<td>Kidney</td>
<td>2.445</td>
<td>1.880</td>
<td>1.822</td>
</tr>
<tr>
<td>Liver</td>
<td>1.911</td>
<td>2.909</td>
<td>1.743</td>
</tr>
<tr>
<td>Testis</td>
<td>1.734</td>
<td>1.809</td>
<td>2.872</td>
</tr>
<tr>
<td>Kidney, Liver</td>
<td>3.282</td>
<td>3.051</td>
<td>1.961</td>
</tr>
<tr>
<td>Kidney, Testis</td>
<td>2.410</td>
<td>2.129</td>
<td>2.521</td>
</tr>
<tr>
<td>Liver, Testis</td>
<td>2.438</td>
<td>2.563</td>
<td>2.526</td>
</tr>
<tr>
<td>All</td>
<td>3.202</td>
<td>3.121</td>
<td>2.958</td>
</tr>
</tbody>
</table>

The estimates support the idea that (UniGene) abundant genes are expressed at higher levels than (UniGene) “rare” genes.
Distinguishing between competing sets of annotations

- Use parameters estimated from the genes with consistent annotations

- At the ambiguous spots, compute the log-likelihood of the observed data for each possible triple of abundance annotations

- Given a complete set of annotations, sum the log-likelihood values over all genes
  - Log-likelihood that the kidney data file contains the correct annotations is equal to $-52, 241$
  - Log-likelihood that the liver data file contains the correct annotations is equal to $-60, 183$
Scrambled rows

- Our “inspired guess” earlier was motivated by the idea that the rows containing the annotations had somehow been reordered.

- We permuted the rows 100 times to obtain empirical p-values for the observed log-likelihoods
  - \( P(\text{kidney is correct}) < 0.01 \)
  - \( P(\text{liver is correct}) = 0.57 \).

- The log-likelihood of the kidney file annotations was not particularly close to the maximum of \(-33,491\). This suggests that we can use the array data to refine the notion of “abundance” on a gene-by-gene basis.
Statistical Tests for Differential Expression

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

10 November 2004
Statistical Tests for Differential Expression

- Student’s t-test
- Simulating nothing
- Family-wise error rate (FWER)
- Permutation tests
- Is FWER too conservative?
- Beta-uniform mixture model
Class Comparison

Perhaps the most common use of microarrays is to determine which genes are differentially expressed between prespecified classes of samples. In general, we refer to this as the class comparison problem. In this lecture, we start looking at the simplest case:

- Given microarray experiments on
  - \( N_A \) sample of type \( A \)
  - \( N_B \) sample of type \( B \)
- Decide which of the \( G \) genes on the microarray are differentially expressed between the two groups.
Student’s t-test

In many cases, we analyze microarrays starting with the “one gene at a time” approach. That is, we first look for a reasonable way to analyze the same problem when we only have one gene, and then figure out how to adapt that method to thousands of genes.

The one-gene version of the class comparison problem with two classes simply asks, “is this gene different in the two classes?” A classical answer is provided by Student’s t-test. We start by estimating the mean and standard deviation in both classes:

$$\bar{x}_A = \frac{1}{N_A} \sum_{i=1}^{N_A} x_i,$$
$$s^2_A = \frac{1}{N_A - 1} \sum_{i=1}^{N_A} (x_i - \bar{x})^2.$$
Weighted difference in means

Next, we pool the estimates of standard deviation from the two groups:

\[ s_P^2 = \frac{(N_A - 1)s_A^2 + (N_B - 1)s_B^2}{N_A + N_B - 2}. \]

The two-sample t-statistic is the difference in means, weighted by the pooled estimate of the standard deviation and the number of samples:

\[ t = \frac{\bar{x}_B - \bar{x}_A}{s_P \sqrt{1/N_A + 1/N_B}}. \]

Question: Why not just use the difference in means?
Microarray aside: which scale is best?

Before answering the question, we offer a slight reinterpretation. Most (but not all) analysts believe that microarray data should be transformed by computing logarithms before testing for differential expression. The key mathematical fact supporting this belief is that the logarithm turns multiplication into addition:

\[ \log(xy) = \log(x) + \log(y). \]

In particular

\[ \log(2x) = \log(x) + \log(2), \quad \log(x/2) = \log(x) - \log(2). \]

Differences on the log scale can be interpreted as “fold change” on the original scale of the data. Increases and decreases by the same fold change are treated equally on the log scale.
Why the standard deviation matters
Why the standard deviation matters

sigma = 1

sigma = 1/2
Why the standard deviation matters

\[ \text{Frequency} \]

\[ \text{sigma} = 1 \]

\[ \text{sigma} = \frac{1}{2} \]

\[ \text{sigma} = 2 \]
T-statistics

Recall the definition of the t-statistic:

\[
t = \frac{\bar{x}_B - \bar{x}_A}{s_P \sqrt{1/N_A + 1/N_B}}.
\]

Three ways to get a larger t-statistic:

- Bigger difference in means
- Smaller standard deviation
- More samples
What about p-values?

**Null hypothesis**: The difference in mean expression between the two groups is zero.

**Two-sided alternative hypothesis**: The difference in mean expression is non-zero.

P-value = probability of seeing a t-statistic this extreme under the null hypothesis = area in both tails of the distribution.

Interpretation: if you repeat the same experiment many times (with the same number of samples in the two groups), the p-value represents the proportion of times that you would expect to see a t-statistic this large.

BUT: Computing a t-statistic for each gene on a microarray is analogous to performing the same experiment many times.
Simulating nothing

We simulated a microarray data set with no differences:

```
library(Marray)

n.genes <- 10000
an <- 10
bn <- 10
n.samples <- an + bn
type <- factor(rep(c('A', 'B'), times=c(an, bn)))
data <- matrix(rnorm(n.genes*n.samples), nrow=n.genes)
am <- apply(data[, type=='A'], 1, mean)
bm <- apply(data[, type=='B'], 1, mean)
av <- apply(data[, type=='A'], 1, var)
bv <- apply(data[, type=='B'], 1, var)
sp2 <- ((an-1)*av + (bn-1)*bv)/(an+bn-2)
```
The t distribution

t.stat <- (bm - am)/sqrt(sp2)/sqrt(1/an+1/bn)
hist(t.stat, breaks=100, xlab='')
P-values are uniformly distributed

p.val <- sapply(t.stat, function(tv, df) {
  2*(1-pt(abs(tv), df))
}, an + bn - 2)
hist(p.val, breaks=100, xlab='')
How significant is nothing?

```r
> sum(p.val < 0.05) # observed
[1] 505
> 0.05 * n.genes # expected
[1] 500
> sum(p.val < 0.01) # observed
[1] 109
> 0.01 * n.genes # expected
[1] 100
```

If there are no real differences, and if we can treat different genes as though they are “replicates” of the same experiment, then

1. Number of genes with $p < \alpha$ is approximately $\alpha N$.

2. The distribution of p-values is uniform.
Statistical error types

Statisticians are (on average) obsessed with errors. They also tend to use circumlocutions that make it more difficult for non-statisticians to understand them. For example, “rejecting the null hypothesis” means “calling a gene differentially expressed”.

<table>
<thead>
<tr>
<th>Test Result</th>
<th>Truly Different</th>
<th>Truly Unchanged</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Positive</strong></td>
<td>True Positive (TP)</td>
<td>False Positive (FP)</td>
</tr>
<tr>
<td><strong>Negative</strong></td>
<td>False Negative (FN)</td>
<td>True Negative (TN)</td>
</tr>
</tbody>
</table>

P-value = Prob(Type I Error)

To control Type II Errors (FN), you have to increase the sample size to ensure enough power to detect the true differences.
Family-wise error rate (FWER)

FWER = probability of getting at least one FP when performing many statistical test = probability of making at least one mistake

Bonferroni adjustment: To achieve $FWER \leq \alpha$ when looking at $G$ genes, restrict on a per-gene basis to $p \leq \alpha/G$.

> bonferroni <- 0.05/n.genes
> bonferroni
[1] 5e-06
> sum(p.val < bonferroni)
[1] 0
What happens with real data?


This paper uses two-color microarrays to study prostate cancer. Processed with local background subtraction, loess normalization, then taking log ratios with the reference channel.

- 41 samples of apparently normal prostate
- 62 samples of prostate cancer
- 9 samples of lymph node metastases from prostate cancer

We randomly selected ten samples of normal prostate and ten samples of prostate cancer, and performed two-sample t-tests.
There seems to be an overabundance of small p-values, causing the distribution to differ considerably from uniform.
> n.genes <- nrow(data)
> n.genes
[1] 42129
> sum(p.val < 0.05) # observed
[1] 6316
> 0.05 * n.genes # expected
[1] 2106.45
> sum(p.val < 0.01) # observed
[1] 2931
> 0.01 * n.genes # expected
[1] 421.29
> bonferroni <- 0.05/n.genes
> bonferroni
[1] 1.186831e-06
> sum(p.val < bonferroni)
[1] 42
Simulating something

We also simulated two data sets with differences:

1. Data Set I
   - 10 arrays per group, 2000 genes per array
   - Gene expressions in each group are independent, $N(\mu, 1)$.
   - In group A, take all $\mu_A = 0$.
   - 50 genes are different, with $|\mu_A - \mu_B| \sim 5 \times Beta(2, 8)$.

2. Data Set II
   - 10 arrays per group, 10,000 genes per array
   - Mean expression $\mu_A \sim Exp(1/20)$.
   - 100 genes are different, with $\mu_A/\mu_B \sim 1 + 9 \times Beta(3, 7)$.
Bonferroni Correction: Results

- Data Set I (normal model)
  - Truth: 50 genes differ out of 2000
  - With $\alpha = 0.05$, makes 21 positive calls, 21 correct.

- Data Set II (exponential + noise)
  - Truth: 100 genes differ out of 10,000
  - With $\alpha = 0.05$, makes 25 positive calls, 25 correct.

In both cases, there are many false negatives.
Begining to assess the model

A key assumption of the Bonferroni approach is that a uniform distribution adequately describes the p-values when there are no differentially expressed genes present.

We can start testing how good the uniform model is by performing a permutation test. In this case, we simply scramble the labels on the samples.

In the prostate example, we have ten normal and ten cancer samples. We choose ten samples at random to call “normal”, and call the other ten “cancer”, and we repeat the analysis with the two-sample t-test.
P-values for scrambled sample labels

Nearly uniform, with a slight bulge near $p = 0.01$. This might be attributable to an imbalance of “truth” in the permuted groups.
Scrambled data is insignificant

> sum(p.val < 0.05)  # observed
[1] 2257

> 0.05 * n.genes  # expected
[1] 2106.45

> sum(p.val < 0.01)  # observed
[1] 406

> 0.01 * n.genes  # expected
[1] 421.29

> sum(p.val < bonferroni)
[1] 0
Should we believe the p-values?

There is another potential difficulty with using the Bonferroni approach: in order to get a significant gene, we need extremely small p-values. That means we have to estimate the tails of the distribution very accurately, which is a difficult thing to do unless one of two fairly unlikely things happens:

1. The number of samples is extremely large, or

2. The distribution of expression values is almost perfectly described by a normal distribution.

We can use permutations to get around the second problem, but that only makes the first problem worse.
Dudoit’s permutation p-values


- Perform t-test for each gene $g$ and sort the absolute t-statistics, $|t_g|$.

- Repeat many times:
  - Randomly permute sample labels.
  - Compute new t-statistics

- Adjust p-values based on empirical joint distribution of t-statistics to control $FWER$. 
Adjusted p-values, Data Set I
Adjusted p-values, Data Set II

![Graph showing p-values against T-statistic for adjusted and unadjusted data.](image)
Dudoits Method: Results

• Data Set I (normal model)
  • Truth: 50 genes differ, out of 2000.
  • With $\alpha = 0.05$, makes 21 positive calls, 21 correct.

• Data Set II (exponential + noise)
  • Truth: 100 genes differ, out of 10,000.
  • With $\alpha = 0.05$, makes 21 positive calls, 21 correct.

Performance on simulated data is similar to Bonferroni correction, but may be slightly more conservative on the second data set.
Is FWER too conservative?

1. In the prostate data set, Bonferroni with $FWER \leq 5\%$ flagged 42 genes.

2. With an uncorrected $p \leq 1\%$, the model underlying the Bonferroni correction predicts only 421 genes, but we actually observe 2931.

3. With an uncorrected $p \leq 5\%$, the model underlying the Bonferroni correction predicts only 2106 genes, but we actually observe 6316.

Are there only 42 differentially expressed genes among the 42, 129 spots on this array, or are there $2510 = 2931 - 421$? Or maybe even $4210 = 6316 - 2106$?
Opportunity cost

The Bonferroni correction only considers Type I Errors. Microarray experiments, however, are often used for discovery. Findings are usually confirmed by performing additional experiments (typically, real-time PCR). In some cases, the “opportunity cost” of missing out on a discovery (by making a Type II Error) is greater than the “validation cost” of finding some false positives (Type I Errors) in your list of genes.

Like anything else, there are trade-offs. By choosing a smaller significance cutoff for the p-values, you get fewer false positives but more false negatives. By choosing a larger cutoff, you get more false positives and fewer false negatives.
The false discovery rate

\[ FDR = \frac{FP}{TP + FP} = \text{fraction of false positives among all genes called differentially expressed by the test.} \]

Here is a crude way to estimate FDR: Assume the uniform model for p-values holds under the null hypothesis. The expected number of false discoveries at a given p-value cutoff is \( pG \). If the total number of discoveries is \( V \), then we can estimate \( FDR = \frac{pG}{V} \).

In the prostate cancer example, this gives

- When \( p = 0.05 \), \( FDR = 0.3334 = \frac{2106}{6316} \).
- When \( p = 0.01 \), \( FDR = 0.1436 = \frac{421}{2931} \).

This estimate is not very good; it overestimates the number of errors by not accounting for the fact that there seem to be some true discoveries.
Significance analysis of microarrays (SAM)


- Compute modified t-statistics (increase \( \sigma \) to minimize coefficient of variation across the array).

- Recompute t-statistics based on balanced permutations (each group equally represented) of the sample labels.

- Decide on significance cutoff based on quantile-quantile plot of observed versus expected t-statistics.

- Estimate FDR from the permutations.
SAM, Data Set II

![Graph showing observed T statistics versus expected T statistics (Empirical)](image)

- Observed T Statistics vs. Expected T Statistics (Empirical)
- Data points plotted in a scatter plot
- Lines indicating different statistical thresholds
SAM: Results

- Data Set I (normal model)
  - Truth: 50 genes differ, out of 2000.
  - With $FDR = 0.10$, makes 32 positive calls, 30 correct.

- Data Set II (exponential + noise)
  - Truth: 100 genes differ, out of 10,000
  - With $FDR = 0.10$, makes 41 positive calls, 37 correct.

Detects more true positives in simulated data than Bonferroni or Dudoit, at some cost in false positives. Like Dudoit’s method, it is computationally intensive.
Beta-uniform mixture model (BUM)

Idea: Model the p-values as a mixture of a uniform distribution and a beta distribution. Estimate mixture parameters. Obtain estimates of TP, FP, FN, TN as a function of significance cutoff.
BUM, Data Set I
BUM, Data Set II
BUM: Results

- **Data Set I (normal model)**
  - Truth: 50 genes differ, out of 2000.
  - With $FDR = 0.10$, makes 33 positive calls, 31 correct.
  - Estimates that 2.8% of genes are different (truth = 2.5%)

- **Data Set II (exponential + noise)**
  - Truth: 100 genes differ, out of 10,000
  - With $FDR = 0.10$, makes 40 positive calls, 37 correct.
  - Estimates that 0.7% of genes are different (truth = 1.0%)

Results equivalent to SAM, with much less computation.
BUM results on prostate data

We have already seen the histogram, and the fit of the beta-uniform mixture.

- With $FDR < 0.01$, calls 427 genes differentially expressed.
- With $FDR < 0.05$, calls 1513 genes differentially expressed.
- With $FDR < 0.10$, calls 2727 genes differentially expressed.

Overall, BUM estimates that 26% of the genes are differentially expressed at some level. (That’s more than 10,000 genes!)
Rank-Based Tests of Differential Expression

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

10 November 2004
Rank-Based Tests of Differential Expression

- Wilcoxon rank-sum test
- Empirical Bayes
- The Tail-Rank Test
- Looking at the results
Nonparametric tests

The t-test for differences in mean expression between two groups of samples assumes that the measurements in each group are normally distributed. If this assumption is far from the truth, then the t-statistics and p-values you get may be meaningless. (Actually, departures from normality tend to increase the Type II error, especially when the sample size is small.)

Statistically, the dispute over log-transforming microarray data reduces to whether a normal distribution better describes the data on the raw scale or on the log scale.

We can avoid this problem entirely by using a statistical test that does not assume anything about the distributions. These tests are usually called distribution-free or nonparametric.
Wilcoxon rank-sum test

The most common nonparametric test for a difference in mean expression is the Wilcoxon rank-sum test, which is also known as the Mann-Whitney test.

We assume that we have sample measurements from two groups:

\[ X_1, X_2, \ldots, X_{n_X}, \]

\[ Y_1, Y_2, \ldots, Y_{n_Y}, \]

We then rank these values from smallest to largest, getting something like

\[
\begin{align*}
X_3 & \leq Y_5 & \leq X_1 & \leq X_{10} & \leq Y_1 & \leq \cdots & \leq X_2 \\
1 & \quad 2 & \quad 3 & \quad 4 & \quad 5 & \quad n_X + n_Y
\end{align*}
\]
Computing rank-sums

Next, we compute a statistic $W$ by summing the ranks of the measurements from the first group. In our example,

$$W = 1 + 3 + 4 + \cdots + (n_X + n_Y).$$

$W$ is always an integer, and it is easy to compute its minimum and maximum values. The minimum occurs when all the $X$ values are smaller than all the $Y$ values. Thus, all the $X$ values are at the beginning of the list, and we have

$$W \geq 1 + 2 + \cdots + n_X = \frac{n_X(n_X + 1)}{2}.$$
The maximum occurs when all the $X$ values come after all the $Y$ values, giving

\[
W \leq (n_Y + 1) + (n_Y + 2) + \cdots + (n_Y + n_X)
= n_X n_Y + (1 + 2 + \cdots n_X)
= n_X n_Y + \frac{n_X(n_X + 1)}{2}
= \frac{n_X(2n_Y + n_X + 1)}{2}.
\]

Those formulas are very nice, but let’s see what happens when we have 10 samples in each group. The range of values for $W$ in this case is

\[
(1 + 2 + \cdots + 10) = 55 \leq W \leq (11 + 12 + \cdots + 20) = 155
\]
When is a rank-sum significant?

Intuitively, if we get a value of $W$ near its extreme values, then we strongly suspect that the two groups are different. If, however, we get a value near the middle, then we suspect that there is no difference. How can we make this idea more precise?

First, let’s do some exploration. We start by generating an unstructured random data matrix:

```r
> n.genes <- 40000
> n.samples <- 20
> type <- rep(c('A', 'B'), each=10)
> data <- matrix(rnorm(n.genes*n.samples),
+   ncol=n.samples)
```
Next, we rank the values in each row:

```r
> ranked.data <- apply(data, 1, rank)
> dim(data)
[1] 40000 20
> dim(ranked.data)
[1] 20 40000
```

Notice that the matrix of ranks is transposed when compared to the original data matrix. So, we can compute the Wilcoxon rank-sum statistics by summing the correct ranks by column:

```r
> wilstat <- apply(ranked.data[type=='A',], 2, sum)
> summary(wilstat)
    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
   56.0   96.0  105.0  105.1  114.0  153.0
```

We don’t quite get to the extremes...
Null distribution of Wilcoxon rank-sum

```r
> hist(wilstat, breaks=seq(54.5, 155.5, by=1))
```
What happens with real data?

Let’s return to the prostate cancer data set used in the last lecture. Recall that this data set contains the log ratios of 42,129 genes measured using two-color fluorescent microarrays and a common reference channel. Recall also that we selected a subset of 10 samples from normal prostate and 10 samples of prostate cancer.

We compute the Wilcoxon rank-sum statistics for this data set:

```r
> ranked.data <- apply(exprs(prostate), 1, rank)
> dim(ranked.data)
[1] 20 42129
> status <- pData(pheno)$Status
> ranksum <- apply(ranked.data[status=='N',], 2, sum)
```
Real data yields extreme statistics

> summary(ranksum)

Min. 1st Qu. Median Mean 3rd Qu. Max.
55.0 92.0 104.0 103.9 116.0 155.0

> hist(ranksum, breaks=seq(54.5, 155.5, by=1))
Distributions matter

When we simulated data, we got values of the rank-sum statistic between 56 and 153.

When we looked at real data, we got rank-sum statistics that spanned the full range of possible values from 55 to 155.

One (pessimistic) interpretation of this result is that rank-sum statistics are only useful in microarray experiments if they find genes where all the values in one group are less than all the values in the other group.

We can do better by looking more carefully at the distributions.
R contains functions to explore the distribution of the rank-sum statistics:

- `rwilcox` generate random values from the Wilcoxon distribution
- `dwilcox` probability density function
- `pwilcox` cumulative probability function
- `qwilcox` quantile function

This set of functions parallels those for other distributions (like `rnorm`, `dnorm`, `pnorm`, and `qnorm` for the normal distribution).

One should note, however, that the rank-sum statistics in R are shifted so that the smallest value is 0 instead of $n_X$. 
The null distribution

```r
> minW <- sum(1:10); maxW <- sum(11:20)
> breaker <- seq(minW-0.5, maxW+.5, by=1)
> hist(wilstat, breaks=breaker, prob=TRUE)
> lines(minW:maxW, dwilcox(0:(maxW-minW), 10, 10),
+        col='red', lwd=3)
```

![Histogram of wilstat](image)
The real distribution

> hist(ranksum, breaks=breaker, prob=TRUE, 
+     ylim=c(0, 0.03))
> lines(minW:maxW, dwilcox(0:(maxW-minW), 10, 10), 
+       col='red', lwd=3)
> wilp <- sapply(ranksum-minW, function(w, m, n) {
+   if (w > m*n/2) 2*(1-pwilcox(w, m, n))
+   else 2*pwilcox(w, m, n)
+ }, 10, 10)
> hist(wilp, breaks=100)
Empirical Bayes

The discreteness of the values of the Wilcoxon statistics makes the distribution of p-values problematic for the application of something like BUM to sort out the significance in the face of multiple testing. Instead, we’re going to use a different approach.

Basic idea

Assume that there are two classes of genes, **Different** and **Not Different**. We assume prior probabilities

- $p_0 = \text{Prob}(\text{Not Different})$
- $p_1 = 1 - p_0 = \text{Prob}(\text{Different})$

and density functions

- $f_0(y)$, known Wilcoxon, if Not Different
- $f_1(y)$, unknown, if Different
Mixtures

The overall probability density function is a mixture

\[ f(y) = p_0 f_0(y) + p_1 f_1(y). \]

Bayes Theorem: \( P(H|D) = P(D|H)P(H)/P(D) \)

Applying Bayes Theorem gives posterior estimates:

\[ p_1(y) \equiv \text{Prob}(\text{Diff}|Y = y) = 1 - p_0 f_0(y)/f(y) \]

and

\[ p_0(y) \equiv \text{Prob}(\text{NotDiff}|Y = y) = p_0 f_0(y)/f(y) \]

We can use the observed data to estimate the overall density function by \( \hat{f}(y) \) (typically by log-transforming the observed function and fitting a curve.)
Empirical Bayes

The “empirical” nature of this Bayesian idea is that we can adjust the “prior” $p_0$ after looking at the data, and thus obtain some reasonable values for it. First, here is how well we fit the distribution:

![Graph showing empirical and theoretical fits](image)
This graph assumes $p_0 = 1$, so no genes are different. The posterior probability of being different becomes negative in the middle of the graph. This results from the “empirical” nature of the estimate without imposing a full model. We can, however, adjust $p_0$ to prevent seeing any negative probabilities.
This shows posterior probabilities with $p_0 = 0.7, 0.8, 0.9, 1.0$. Somewhere between $p_0 = 0.7$ and $p_0 = 0.8$, all the posterior probabilities become positive.
This plot uses $p_0 = 0.75$, which is essentially the largest value we can use for $p_0$ and ensure that all the posterior probabilities are positive. The horizontal line indicates a posterior probability of 90% that a gene is differentially expressed.
How does this work in R?

We have written a package that implements the empirical Bayes method with Wilcoxon statistics. For the computations just shown, do the following:

```R
> efron <- ET.empirical.bayes(exprs(prostate),
+    status)
> hist(efron)
> plot(efron, p0=c(0.7, 0.8, 0.9, 1.0), signif=NULL)
> abline(h=0)
> plot(efron, p0=0.75, ylim=c(0,1))
> abline(h=0)
```
How many genes are differentially expressed?

As a crude estimate, the fact that we had to take $p_0 = 0.75$ suggests that 25% of the genes may be different; this is consistent with the BUM estimate from last time. With a cutoff of 90% on the posterior probability, we get:

```r
> sig.range(efron, 0.75, 0.9)
$low
[1] 68
$high
[1] 143
> sum(efron$wilstats<68)
[1] 839
> sum(efron$wilstats>143)
[1] 825
```
The Wilcoxon rank-sum test and the t-test both look at the same property: is the mean expression the same? When looking for cancer biomarkers, this may well be the wrong question.

Cancers that are histologically the same are not identical. Deletion of part of chromosome 3 (3p14-p23) is found in 50% of non-small-cell lung cancers; MYC amplification is found in 14% of stomach cancers; BRCA1 mutations are found in a subset of breast cancers; a translocation between chromosomes 11 and 14 occurs in 35% of mantle cell lymphomas. These genetic abnormalities directly causes specific differences in gene expression that only occur in a subset of cancers. Statistically, these results suggest that the distributions of gene expression in cancer patients are likely to differ from the healthy distributions in much more than the location of the center.
Motivation: Subset Biomarkers

If a biomarker is only present in 20% of the cancer samples, then the distributions might look something like this.
Outline of The Tail-Rank Test

- Collect data on $G$ genes from $n_H$ healthy individuals. Write $X_{g,i}$ for measurement of gene $g$ on individual $i$. Assume for fixed $g$ that $X_{g,i} \sim X_g$ are IID.

- Specify a target value $\psi$ for specificity.

- Estimate, for each $g$, a threshold $\tau_g$ such that $\text{Prob}(X_g < \tau_g) = \psi$.

- Collect data from $n_C$ cancer patients. Count the number $Y_g$ of cancer patients for which the measured expression level of gene $g$ exceeds $\tau_g$; we call $Y_g$ the tail-rank statistic.

- Call $g$ a biomarker if $Y_g$ exceeds a certain threshold.
The null distribution

Null hypothesis: gene $g$ is not a useful biomarker.

More precisely: the measurements on cancer patients have the same distribution as the measurements from healthy individuals. Then: all $Y_g$ have identical binomial distributions,

$$Y_g \sim Y = Binom(n_C, 1 - \psi).$$

The point here is that the probability of being in the tail is the same for healthy and cancer, and is given by $1 - \psi$, where $\psi$ was the desired specificity.
Even when we perform the same test for $G$ genes, the expected maximum value of $G$ independent instances of $Y_g$ remains small.

Let $M_G = \max_{g=1}^{G} (Y_g)$ be the maximum over $G$ IID binomial random variables. Also, let

$$\alpha = \alpha(m) = \text{Prob}(Y > m)$$

$$\gamma = \text{Prob}(M_G > m)$$

Then

$$1 - \gamma = \text{Prob}(M_G \leq m)$$

$$= \text{Prob}(Y_1 \leq m, \ldots, Y_G \leq m)$$

$$= \text{Prob}(Y \leq m)^G = (1 - \alpha)^G.$$
The maximum value expected by chance

Solving,

\[ \alpha = 1 - (1 - \gamma)^{1/G}. \]

and \( m \) is the \((1 - \alpha)^{\text{th}}\) quantile of a single binomial distribution:

<table>
<thead>
<tr>
<th>( n_C )</th>
<th>( \gamma = 0.01, \psi = 0.99 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( G = 100 )</td>
<td>100 ( \cdot ) 1000 ( \cdot ) 10000 ( \cdot ) 100000</td>
</tr>
<tr>
<td>10</td>
<td>3</td>
</tr>
<tr>
<td>20</td>
<td>3</td>
</tr>
<tr>
<td>50</td>
<td>5</td>
</tr>
<tr>
<td>100</td>
<td>6</td>
</tr>
<tr>
<td>250</td>
<td>10</td>
</tr>
<tr>
<td>500</td>
<td>15</td>
</tr>
</tbody>
</table>
Interpretation

One needs to specify two parameters in order to apply the tail-rank test.

1. $\psi$, the desired specificity of the biomarker

2. $\gamma$, the desired bound on the FWER

Then, given the number of genes and the number of cancer samples, the values $m$ in a table like the previous one represent the largest value of $Y_g$ that we would expect to see by chance over the entire microarray. Any gene where we observe $Y_g > m$ is a potential biomarker.
Tail-rank and real data

We return yet again to our prostate cancer data set. We will now start using the entire data set, which contains 41 samples from normal prostate, 62 prostate cancer samples, and 9 samples from lymph node metastases of prostate cancer. With this number of samples, taking $\gamma = 0.95$ and $\psi = 0.95$, a gene was called a biomarker if at least 16 of the 71 cancer samples were above the threshold.

We assumed that the log ratios of the normal prostate samples were normally distributed. We computed 90% tolerance bounds for the 5th and 95th percentiles, and counted the number of combined prostate cancer samples whose log ratios fell outside these boundaries.
Tail-rank results

We identified 1,766 spots that were “positive” biomarkers, since they were present at higher than normal levels in at least 16 cancer samples. We also identified 1,930 spots that were “negative” biomarkers, since they were expressed at lower than normal levels in at least 16 samples. In total, we identified 3,692 spots as candidate biomarkers.

Although the theory told us the number of false positives should be close to zero, we decided to test this using both simulations and a permutation test. We simulated completely random (IID normal) data 100 times, and we permuted the samples labels on the real data 100 times.
Pretty good, when you consider that the test with the real data detected a few thousand potential markers!
Differential expression results

We repeated the t-test analysis on the full data set (adjusting for multiple testing using BUM). With $FDR < 0.05$, we used a cutoff at $p < 0.000045$ or $|t| > 4.25$. We detected 3,522 differentially expressed spots. Of these, 1,415 spots were overexpressed in prostate cancer and 2,107 spots were underexpressed.

We also repeated the Wilcoxon test with the empirical Bayes approach. In order to get comparable results, we selected a cutoff corresponding to a posterior probability of 99.9%. We detected 3,627 differentially expressed spots. Of these, 1,498 spots were overexpressed and 2,129 spots were underexpressed in prostate cancer.
Comparing tests

The number of genes found by the three tests was very similar. Are they finding the same things?

There was good agreement between the t-test and the Wilcoxon test. More than 90% (1,905) of underexpressed and 88% (1,244) of overexpressed spots that were found by the t-test were also detected by the Wilcoxon test. So, we only need to compare one of these to the tail-rank test.
Comparing tests

Lower left and right = different by T, not by tail-rank

Upper center = different by tail-rank, not by T.
Looking at the results

Since the tests give different answers, which one should we believe?

Both, since they give answers to different questions.

Whether you perform one test or many, however, it is useful to look at the expression values for some of the genes that you find, if only to make sure you believe the results.

A useful R function for this purpose is `stripchart`. Here is an example for our data set. First, we get the clinical status as an ordered factor.

```
> x <- ordered(clinical.info$Status, 
+   levels=c('N', 'T', 'L'))
```
Selecting interesting genes

Now we look at genes with small tail-rank statistics ($< 2$) and significant t-statistics.

```r
> k.weird <- tr.stats < 2 & (abs(t.stats) > 4.25)
> sum(k.weird)
[1] 38
```

We can select one of the “weird” genes and get its expression data.

```r
> i.k.weird <- (1:length(k.weird))[k.weird]
> i <- sample(i.k.weird, 1)
> y <- as.vector(t(expression.data[i,]))
```
Outliers can throw off the estimates

```r
> label <- as.character(gene.info$Gene.Symbol[i])
> stripchart(y ~ x, xlab='', main=label,
+   method='jitter')
```
Some genes are normally variable
Selecting interesting biomarkers

Now we look at genes with significant tail-rank statistics and small t-statistics ($|t| < 1.25$).

```r
> k.weird <- tr.stats > 15 & (abs(t.stats) < 1.25)
> sum(k.weird)
[1] 52
```

We use the same idea to select some of these genes and plot stripcharts to see if the values agree with what we think the test should be doing.
GITA
When T and Tail-Rank agree
Using the tail-rank test in R

We developed the tail-rank test; the paper has been submitted. A preprint, along with an R package is available on the web at

http://bioinformatics.mdanderson.org/TailRank

Basic usage:

> tr.stats <- tail.rank.test(data, status)