GS01 0163
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

18 October 2005
Lecture 13: Microarrays in R: Start to Finish

- Source of the Data Set
- Understanding the Sample Information
- First Pass Using dChip
- Starting in R
- Just RMA
Source of the Data Set

In today’s lecture, we’re going to perform a complete start-to-finish analysis of a microarray data set. We have chosen to use a leukemia data set that we looked at briefly in an earlier lecture. The data set consists of U95A microarray experiments on

1. 24 patients with acute lymphocytic leukemia (ALL)
2. 28 patients with acute myeloid leukemia (AML)
3. 20 patients with mixed lineage leukemia (MLL)
**MLL** translocations specify a distinct gene expression profile that distinguishes a unique leukemia

Scott A. Armstrong, Jane E. Staunton, Lewis B. Silverman, Rob Pieters, Monique L. den Boer, Mark D. Minden, Stephen E. Sallan, Eric S. Lander, Todd R. Golub & Stanley J. Korsmeyer

Published online: 3 December 2001, DOI: 10.1038/ng765

Acute lymphoblastic leukemias carrying a chromosomal translocation involving the mixed-lineage leukemia gene (**MLL, ALL1, HRX**) have a particularly poor prognosis. Here we show that they have a characteristic, highly distinct gene expression profile that is consistent with an early hematopoietic progenitor expressing select multilineage markers and individual **HOX** genes. Clustering algorithms reveal that lymphoblastic leukemias with **MLL** translocations can clearly be separated from conventional acute lymphoblastic and acute myelogenous leukemias. We propose that they constitute a distinct disease, denoted here as **MLL**, and show that the differences in gene expression are robust enough to classify leukemias correctly as **MLL**, acute lymphoblastic leukemia or acute myelogenous leukemia. Establishing that **MLL** is a unique entity is critical, as it mandates the examination of selectively expressed genes for urgently needed molecular targets.
Results reported in the paper

- MLL is distinct from conventional ALL
  - Based on arrays, \( \sim 1200 \) differentially expressed genes

- MLL shows multilineage gene expression
  - Looking at the expressed genes, find some from B cells and some from myeloid cells

- MLL is arrested at an early stage of hematopoiesis

- Some HOX genes are overexpressed in MLL
Results reported in the paper

- MLL is distinct from AML as well as from conventional ALL
  - Principal components analysis
  - Selected genes using one-versus-all comparisons

- Gene expression profiles correctly classify ALL, AML, MLL
  - K-nearest-neighbors (KNN) predictor
Getting the data

Note: The links to supplementary data listed in the original paper no longer work (as of the morning of 18 October 2005). However, the data is available at: http://www.broad.mit.edu/cgi-bin/cancer/datasets.cgi
Getting the data

Scroll down (or search for “translocations”) to find the data set. You’ll need all six collections of CEL files along with the “scaling_factors_and_fig_key.txt” that contains the sample information.
Understanding the Sample Information

<table>
<thead>
<tr>
<th>sample_name</th>
<th>scan name</th>
<th>linear scaling factor</th>
<th>paper_figure</th>
</tr>
</thead>
<tbody>
<tr>
<td>ALL_1</td>
<td>CL2001011101AA</td>
<td>1</td>
<td>TRAIN/figs.1,4,5</td>
</tr>
<tr>
<td>ALL_2</td>
<td>CL2001011104AA</td>
<td>0.9399</td>
<td>TRAIN/figs.1,4,5</td>
</tr>
<tr>
<td>ALL_3</td>
<td>CL2001011105AA</td>
<td>1.6781</td>
<td>TRAIN/figs.1,4,5</td>
</tr>
<tr>
<td>ALL_4</td>
<td>CL2001011108AA</td>
<td>1.0635</td>
<td>TRAIN/figs.1,4,5</td>
</tr>
<tr>
<td>ALL_5</td>
<td>CL2001011109AA</td>
<td>1.3875</td>
<td>TRAIN/figs.1,4,5</td>
</tr>
<tr>
<td>ALL_6</td>
<td>CL2001011111AA</td>
<td>1.1869</td>
<td>TRAIN/figs.1,4,5</td>
</tr>
<tr>
<td>ALL_7</td>
<td>CL2001011112AA</td>
<td>1.1951</td>
<td>TRAIN/figs.1,4,5</td>
</tr>
<tr>
<td>ALL_8</td>
<td>CL2001011116AA</td>
<td>1.2615</td>
<td>TRAIN/figs.1,4,5</td>
</tr>
<tr>
<td>ALL_9</td>
<td>CL2001011113AA</td>
<td>1.5606</td>
<td>TRAIN/figs.1,4,5</td>
</tr>
<tr>
<td>ALL_10</td>
<td>CL2001011124AA</td>
<td>1.2855</td>
<td>TRAIN/figs.1,4,5</td>
</tr>
<tr>
<td>ALL_11</td>
<td>CL2001011150AA</td>
<td>1.1064</td>
<td>TRAIN/figs.1,4,5</td>
</tr>
<tr>
<td>ALL_12</td>
<td>CL2001011153AA</td>
<td>1.2399</td>
<td>TRAIN/figs.1,4,5</td>
</tr>
<tr>
<td>ALL_13</td>
<td>CL2001011154AA</td>
<td>1.4928</td>
<td>TRAIN/figs.1,4,5</td>
</tr>
<tr>
<td>ALL_14</td>
<td>CL2001011114AA</td>
<td>1.0762</td>
<td>TRAIN/figs.1,4,5</td>
</tr>
<tr>
<td>ALL_15</td>
<td>CL2001011118AA</td>
<td>1.3057</td>
<td>TRAIN/figs.1,4,5</td>
</tr>
<tr>
<td>ALL_16</td>
<td>CL2001011119AA</td>
<td>1.1453</td>
<td>TRAIN/figs.1,4,5</td>
</tr>
<tr>
<td>ALL_17</td>
<td>CL2001011120AA</td>
<td>1.1352</td>
<td>TRAIN/figs.1,4,5</td>
</tr>
<tr>
<td>ALL_18</td>
<td>CL2001011121AA</td>
<td>1.1639</td>
<td>TRAIN/figs.1,4,5</td>
</tr>
<tr>
<td>ALL_19</td>
<td>CL2001011122AA</td>
<td>1.2322</td>
<td>TRAIN/figs.1,4,5</td>
</tr>
<tr>
<td>ALL_20</td>
<td>CL2001011123AA</td>
<td>1.2835</td>
<td>TRAIN/figs.1,4,5</td>
</tr>
<tr>
<td>ALL_21</td>
<td>CL2001011134AA</td>
<td>1.1707</td>
<td>test/figs 4,5</td>
</tr>
<tr>
<td>ALL_22</td>
<td>CL2001011151AA</td>
<td>1.2464</td>
<td>test/figs 4,5</td>
</tr>
<tr>
<td>ALL_23</td>
<td>CL2001011110AA</td>
<td>1.3895</td>
<td>test/figs 4,5</td>
</tr>
</tbody>
</table>
Including critical factors

<table>
<thead>
<tr>
<th>Scan name</th>
<th>Sample Name</th>
<th>type</th>
<th>Split</th>
</tr>
</thead>
<tbody>
<tr>
<td>CL2001011101AA</td>
<td>ALL01</td>
<td>ALL</td>
<td>Training</td>
</tr>
<tr>
<td>CL2001011104AA</td>
<td>ALL02</td>
<td>ALL</td>
<td>Training</td>
</tr>
<tr>
<td>CL2001011105AA</td>
<td>ALL03</td>
<td>ALL</td>
<td>Training</td>
</tr>
<tr>
<td>CL2001011108AA</td>
<td>ALL04</td>
<td>ALL</td>
<td>Training</td>
</tr>
<tr>
<td>CL2001011109AA</td>
<td>ALL05</td>
<td>ALL</td>
<td>Training</td>
</tr>
<tr>
<td>CL2001011111AA</td>
<td>ALL06</td>
<td>ALL</td>
<td>Training</td>
</tr>
<tr>
<td>CL2001011112AA</td>
<td>ALL07</td>
<td>ALL</td>
<td>Training</td>
</tr>
<tr>
<td>CL2001011116AA</td>
<td>ALL08</td>
<td>ALL</td>
<td>Training</td>
</tr>
<tr>
<td>CL2001011113AA</td>
<td>ALL09</td>
<td>ALL</td>
<td>Training</td>
</tr>
<tr>
<td>CL2001011124AA</td>
<td>ALL10</td>
<td>ALL</td>
<td>Training</td>
</tr>
<tr>
<td>CL2001011150AA</td>
<td>ALL11</td>
<td>ALL</td>
<td>Training</td>
</tr>
<tr>
<td>CL2001011153AA</td>
<td>ALL12</td>
<td>ALL</td>
<td>Training</td>
</tr>
<tr>
<td>CL2001011154AA</td>
<td>ALL13</td>
<td>ALL</td>
<td>Training</td>
</tr>
<tr>
<td>CL2001011114AA</td>
<td>ALL14</td>
<td>ALL</td>
<td>Training</td>
</tr>
<tr>
<td>CL2001011118AA</td>
<td>ALL15</td>
<td>ALL</td>
<td>Training</td>
</tr>
<tr>
<td>CL2001011119AA</td>
<td>ALL16</td>
<td>ALL</td>
<td>Training</td>
</tr>
<tr>
<td>CL2001011120AA</td>
<td>ALL17</td>
<td>ALL</td>
<td>Training</td>
</tr>
<tr>
<td>CL2001011121AA</td>
<td>ALL18</td>
<td>ALL</td>
<td>Training</td>
</tr>
<tr>
<td>CL2001011122AA</td>
<td>ALL19</td>
<td>ALL</td>
<td>Training</td>
</tr>
<tr>
<td>CL2001011123AA</td>
<td>ALL20</td>
<td>ALL</td>
<td>Training</td>
</tr>
<tr>
<td>CL2001011134AA</td>
<td>ALL21</td>
<td>ALL</td>
<td>Test</td>
</tr>
<tr>
<td>CL2001011151AA</td>
<td>ALL22</td>
<td>ALL</td>
<td>Test</td>
</tr>
<tr>
<td>CL2001011110AA</td>
<td>ALL23</td>
<td>ALL</td>
<td>Test</td>
</tr>
</tbody>
</table>
First Pass Using dChip

Early in the course, we used this data set in dChip. One of the things we discovered was that the samples were run on two different iterations of the U95A GeneChip. The ALL and MLL samples were run on version 1 of the U95A and the AML samples were run on the U95Av2 chip.

Our first step in dealing with this issue was to install the CEL files into two different directories. In dChip, we handle this by first producing a “data file list” that tells dChip which directories to look into:

```
G:\ShortCourse\CELFiles
G:\ShortCourse\AMLCELFiles
```
Probe Set Mask File

We also have to produce a “probe set mask” file that tells dChip which probe sets to ignore (since they changed from one version of the chip to the next).

```
160030_at
160039_at
160025_at
160024_at
160023_at
160022_at
160021_r_at
160020_at
160037_at
160038_s_at
160043_at
160035_at
160042_s_at
160027_s_at
160041_at
160040_at
160032_at
160033_s_at
160036_at
160034_s_at
160031_at
160029_at
160028_s_at
160026_at
160026_at
160044_g_at
```
dChip analysis

1. Load the files into dChip using
   - The sample information file
   - The data file list
   - The U95Av2 CDF file
   - The probe set mask

2. Normalize to array ALL21 (median brightness)

3. Quantify using the PM-only model
dChip quality check

- dChip flags several arrays as outliers:
  - ALL10, which has the highest overall brightness
  - MLL09, which has the lowest overall brightness
  - ALL04, ALL16, AML08
dChip Comparison: ALL vs MLL

Obtained 626 genes; False Discovery Rate (Number of 50 permutations, Median: 1.3% (8), 90th percentile: 4.6% (29)
628 differentially expressed genes

<table>
<thead>
<tr>
<th>probe set</th>
<th>gene</th>
<th>baseline mean</th>
<th>experiment mean</th>
<th>fold change</th>
<th>lower bound</th>
<th>upper bound</th>
<th>difference</th>
</tr>
</thead>
<tbody>
<tr>
<td>37680_at</td>
<td>A kinase (PRKA) at</td>
<td>2991.81</td>
<td>147.65</td>
<td>-20.26</td>
<td>-13.19</td>
<td>-29.88</td>
<td>-2844.16</td>
</tr>
<tr>
<td>37250_at</td>
<td>MAD, mothers against</td>
<td>9355.77</td>
<td>697.15</td>
<td>-13.42</td>
<td>-9.62</td>
<td>-17.36</td>
<td>-8658.62</td>
</tr>
<tr>
<td>1325_at</td>
<td>MAD, mothers against</td>
<td>7714.68</td>
<td>617.24</td>
<td>-12.50</td>
<td>-8.54</td>
<td>-17.12</td>
<td>-7097.44</td>
</tr>
<tr>
<td>1488_at</td>
<td>protein tyrosine phosphatase</td>
<td>4121.6</td>
<td>579.89</td>
<td>-7.11</td>
<td>-3.86</td>
<td>-10.50</td>
<td>-3541.71</td>
</tr>
<tr>
<td>34194_at</td>
<td>Homo sapiens mRNA</td>
<td>194.54</td>
<td>200.61</td>
<td>-7.08</td>
<td>-3.32</td>
<td>-13.62</td>
<td>-1219.93</td>
</tr>
<tr>
<td>1077_at</td>
<td>recombination activity</td>
<td>6897.9</td>
<td>892.66</td>
<td>-7.02</td>
<td>-4.26</td>
<td>-11.30</td>
<td>-5915.24</td>
</tr>
<tr>
<td>753_at</td>
<td>nitrogen 2 (osteonic)</td>
<td>2389.96</td>
<td>342.13</td>
<td>-6.99</td>
<td>-2.78</td>
<td>-11.31</td>
<td>-2047.83</td>
</tr>
<tr>
<td>37908_at</td>
<td>guanine nucleotide</td>
<td>6909.7</td>
<td>1035.23</td>
<td>-6.67</td>
<td>-4.15</td>
<td>-11.67</td>
<td>-5874.47</td>
</tr>
<tr>
<td>35614_at</td>
<td>translation factor</td>
<td>7521.88</td>
<td>1216.05</td>
<td>-6.19</td>
<td>-4.07</td>
<td>-8.59</td>
<td>-6305.83</td>
</tr>
<tr>
<td>34800_at</td>
<td>leucine-rich repeats</td>
<td>5226.23</td>
<td>844.98</td>
<td>-6.19</td>
<td>-4.00</td>
<td>-9.80</td>
<td>-4381.25</td>
</tr>
<tr>
<td>31892_at</td>
<td>protein tyrosine phosphatase</td>
<td>868.91</td>
<td>145.9</td>
<td>-5.96</td>
<td>-2.01</td>
<td>-10.04</td>
<td>-723.00</td>
</tr>
<tr>
<td>41266_at</td>
<td>integrin, alpha 6</td>
<td>7952.9</td>
<td>1466.32</td>
<td>-5.42</td>
<td>-3.85</td>
<td>-7.47</td>
<td>-6486.57</td>
</tr>
<tr>
<td>31786_at</td>
<td>KH domain containing</td>
<td>2488.25</td>
<td>472.29</td>
<td>-5.27</td>
<td>-3.29</td>
<td>-9.34</td>
<td>-2015.95</td>
</tr>
<tr>
<td>38408_at</td>
<td>transmembrane 4</td>
<td>6455.32</td>
<td>1264.74</td>
<td>-5.10</td>
<td>-3.47</td>
<td>-7.59</td>
<td>-5190.58</td>
</tr>
<tr>
<td>36650_at</td>
<td>cyclin D2</td>
<td>10001.23</td>
<td>2021.61</td>
<td>-4.95</td>
<td>-3.47</td>
<td>-7.76</td>
<td>-7979.62</td>
</tr>
<tr>
<td>38578_at</td>
<td>tumor necrosis factor</td>
<td>4050.73</td>
<td>828.6</td>
<td>-4.89</td>
<td>-3.33</td>
<td>-7.03</td>
<td>-3222.13</td>
</tr>
<tr>
<td>32778_at</td>
<td>inositol 1,4,5-triphosphate</td>
<td>2099.86</td>
<td>433.92</td>
<td>-4.84</td>
<td>-3.61</td>
<td>-6.41</td>
<td>-1665.94</td>
</tr>
<tr>
<td>40570_at</td>
<td>forkhead box O1A</td>
<td>10307.45</td>
<td>2130.08</td>
<td>-4.84</td>
<td>-3.37</td>
<td>-7.52</td>
<td>-8177.37</td>
</tr>
<tr>
<td>39878_at</td>
<td>protocadherin 9</td>
<td>12520.62</td>
<td>2670.21</td>
<td>-4.69</td>
<td>-3.07</td>
<td>-7.31</td>
<td>-9850.42</td>
</tr>
<tr>
<td>31886_at</td>
<td>5'-nucleotidase, ec</td>
<td>2593.49</td>
<td>567.74</td>
<td>-4.57</td>
<td>-2.82</td>
<td>-7.59</td>
<td>-2025.76</td>
</tr>
<tr>
<td>41690_at</td>
<td>Homo sapiens mRNA</td>
<td>5279.81</td>
<td>1166.43</td>
<td>-4.53</td>
<td>-3.01</td>
<td>-6.55</td>
<td>-4113.38</td>
</tr>
<tr>
<td>33386_at</td>
<td>H1 histone family</td>
<td>6216.48</td>
<td>1381.24</td>
<td>-4.50</td>
<td>-2.46</td>
<td>-10.81</td>
<td>-4835.24</td>
</tr>
<tr>
<td>37780_at</td>
<td>piccolo (presynaptic)</td>
<td>2867.65</td>
<td>638.28</td>
<td>-4.49</td>
<td>-2.35</td>
<td>-6.71</td>
<td>-2229.37</td>
</tr>
</tbody>
</table>
Filter genes

Filter based on expression level and on percent present calls.
Cluster samples
Exporting all the data from dChip

1. Use menu “Tools” – > “Export Expression Value”.
2. Select “all genes”
3. Press “OK”
Starting in R

Load the `affy` package.

```r
> require(affy)
Loading required package: affy
Loading required package: Biobase
Loading required package: tools
Welcome to Bioconductor
Vignettes contain introductory material. To view, simply type: openVignette()
For details on reading vignettes, see the openVignette help page.
Loading required package: reposTools
[1] TRUE
```
Load the Sample Information file

```r
> # remember where the data lives
> home <- 'g:/ShortCourse'
> # use the same sample info file we made for dChip
> si <- read.table(file.path(home, 'InfoFiles',
+   'krc-sample-info.xls'),
+   header=TRUE, sep='\t')
> si[1:5,]
```

<table>
<thead>
<tr>
<th>Scan.name</th>
<th>type</th>
<th>Split</th>
</tr>
</thead>
<tbody>
<tr>
<td>ALL01</td>
<td>CL2001011101AA</td>
<td>ALL Training</td>
</tr>
<tr>
<td>ALL02</td>
<td>CL2001011104AA</td>
<td>ALL Training</td>
</tr>
<tr>
<td>ALL03</td>
<td>CL2001011105AA</td>
<td>ALL Training</td>
</tr>
<tr>
<td>ALL04</td>
<td>CL2001011108AA</td>
<td>ALL Training</td>
</tr>
<tr>
<td>ALL05</td>
<td>CL2001011109AA</td>
<td>ALL Training</td>
</tr>
</tbody>
</table>
Load dChip’s array file

```r
> arrays <- read.table(file.path(home, 'Output', 'affyShortCourse arrays.xls'),
+ header=TRUE, as.is=TRUE, sep='\t')
> # Fix the column names!
> dimnames(arrays)[[2]] <-
+ c(dimnames(arrays)[[2]][2:8], 'x')
> # Use the sample name as the row name
> dimnames(arrays)[[1]] <- arrays$Array
> # Only keep useful columns
> arrays <- arrays[, 3:6]
> # Give them sensible names
> dimnames(arrays)[[2]] <- c('MedianIntensity',
+ 'PercentPresent', 'ArrayOutlier',
+ 'SingleOutlier')
```
Combine sample information

> # Merge sample info with dChip info
> si <- merge(si, arrays, by='row.names',
+ sort=FALSE)
> # Sigh. Fix the row names yet again.
> dimnames(si)[[1]] <- si$Row.names
> # Remove redundant columns
> si <- si[, 2:8]
> rm(arrays) # cleanup
Note that the order has changed!

```r
> si[1:5,]

<table>
<thead>
<tr>
<th>Scan.name</th>
<th>type</th>
<th>Split</th>
<th>MedianIntensity</th>
</tr>
</thead>
<tbody>
<tr>
<td>ALL01</td>
<td>CL20010111101AA</td>
<td>ALL</td>
<td>Training</td>
</tr>
<tr>
<td>ALL24</td>
<td>CL20010111102AA</td>
<td>ALL</td>
<td>Test</td>
</tr>
<tr>
<td>ALL02</td>
<td>CL20010111104AA</td>
<td>ALL</td>
<td>Training</td>
</tr>
<tr>
<td>ALL03</td>
<td>CL20010111105AA</td>
<td>ALL</td>
<td>Training</td>
</tr>
<tr>
<td>ALL04</td>
<td>CL20010111108AA</td>
<td>ALL</td>
<td>Training</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>PercentPresent</th>
<th>ArrayOutlier</th>
<th>SingleOutlier</th>
</tr>
</thead>
<tbody>
<tr>
<td>ALL01</td>
<td>48.2</td>
<td>1.648</td>
</tr>
<tr>
<td>ALL24</td>
<td>38.3</td>
<td>3.778</td>
</tr>
<tr>
<td>ALL02</td>
<td>49.5</td>
<td>1.450</td>
</tr>
<tr>
<td>ALL03</td>
<td>36.8</td>
<td>3.152</td>
</tr>
<tr>
<td>ALL04</td>
<td>38.7</td>
<td>5.299</td>
</tr>
</tbody>
</table>
```
Specialized factors

> # make a factor to compare ALL vs MLL
> temp <- si$type
> temp[temp=='AML'] <- NA
> si$ALLvMLL <- factor(temp)
>
> temp <- si$type
> temp[temp=='MLL'] <- NA
> si$ALLvAML <- factor(temp)
>
> temp <- si$type
> temp[temp=='ALL'] <- NA
> si$MLLvAML <- factor(temp)
> temp <- si$type
> temp[temp=='AML'] <- 'Other'
> temp[temp=='MLL'] <- 'Other'
> si$ALLvOther <- factor(temp)
>
> temp <- si$type
> temp[temp=='ALL'] <- 'Other'
> temp[temp=='MLL'] <- 'Other'
> si$AMLvOther <- factor(temp)
>
> temp <- si$type
> temp[temp=='AML'] <- 'Other'
> temp[temp=='ALL'] <- 'Other'
> si$MLLvOther <- factor(temp)
>
> si$type <- factor(si$type)
> summary(si)

<table>
<thead>
<tr>
<th></th>
<th>type</th>
<th>Split</th>
</tr>
</thead>
<tbody>
<tr>
<td>Scan.name</td>
<td>ALL:24</td>
<td>Length:72</td>
</tr>
<tr>
<td>Length:72</td>
<td>ALL:24</td>
<td>Length:72</td>
</tr>
<tr>
<td>Class :character</td>
<td>AML:28</td>
<td>Class :character</td>
</tr>
<tr>
<td>Mode :character</td>
<td>MLL:20</td>
<td>Mode :character</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>MedianIntensity</th>
<th>PercentPresent</th>
<th>ArrayOutlier</th>
</tr>
</thead>
<tbody>
<tr>
<td>Min. : 804</td>
<td>Min. :28.30</td>
<td>Min. : 0.253</td>
</tr>
<tr>
<td>1st Qu.:1222</td>
<td>1st Qu.:36.80</td>
<td>1st Qu.: 0.729</td>
</tr>
<tr>
<td>Median :1442</td>
<td>Median :41.10</td>
<td>Median : 1.085</td>
</tr>
<tr>
<td>Mean :1483</td>
<td>Mean :40.46</td>
<td>Mean : 1.724</td>
</tr>
<tr>
<td>3rd Qu.:1727</td>
<td>3rd Qu.:44.83</td>
<td>3rd Qu.: 1.697</td>
</tr>
<tr>
<td>Max. :3097</td>
<td>Max. :49.80</td>
<td>Max. :14.337</td>
</tr>
</tbody>
</table>
### SingleOutlier

<table>
<thead>
<tr>
<th></th>
<th>ALLvMLL</th>
<th>ALLvAML</th>
<th>MLLvAML</th>
</tr>
</thead>
<tbody>
<tr>
<td>Min.</td>
<td>0.0440</td>
<td>ALL:24</td>
<td>ALL:24</td>
</tr>
<tr>
<td>1st Qu.</td>
<td>0.1610</td>
<td>MLL:20</td>
<td>AML:28</td>
</tr>
<tr>
<td>Median</td>
<td>0.2405</td>
<td>NA’s:28</td>
<td>NA’s:20</td>
</tr>
<tr>
<td>Mean</td>
<td>0.2741</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3rd Qu.</td>
<td>0.3460</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Max.</td>
<td>0.9520</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### ALLvOther AMLvOther MLLvOther

<table>
<thead>
<tr>
<th></th>
<th>ALL:24</th>
<th>AML:28</th>
<th>MLL:20</th>
</tr>
</thead>
<tbody>
<tr>
<td>Other:48</td>
<td>Other:44</td>
<td>Other:52</td>
<td></td>
</tr>
</tbody>
</table>

---

© Copyright 2004–2005, Kevin R. Coombes and Keith A. Baggerly

GS01 0163: ANALYSIS OF MICROARRAY DATA
Create the phenoData object

> pd <- new('phenoData', pData=si, varLabels=list(
+   Scan.name='CEL file name',
+   type='Histological classification',
+   Split='Used as training or test,'
+   MedianIntensity='Unnormalized median brightness,'
+   PercentPresent='Percentage of present calls',
+   ArrayOutlier='Percentage of Array Outliers',
+   SingleOutlier='Percentage of Single Outliers',
+   ALLvMLL='binary classifier',
+   ALLvAML='binary classifier',
+   MLLvAML='binary classifier',
+   ALLvOtherL='binary classifier',
+   AMLvOther='binary classifier',
+   MLLvOther='binary classifier'))
Create the MIAME object

MIAME = minimum information about a microarray experiment

Some of the BioConductor routines require a MIAME object, even
though they will let you submit a character string as a
description.

> miame <- new('MIAME',
+     name='SA Armstrong',
+     lab='Lander-Golub',
+     title='MLL translocations')
Read in the data from dChip

```r
> temp <- read.table(file.path(home, 'Output', 'affyShortCourse expression.xls'),
+ header=TRUE, as.is=TRUE, sep='\t',
+ quote='', comment.char='')
> # expression data in the later columns
> data <- as.matrix(temp[, 6:77])
> # gene identifiers in the first five columns
> gi <- temp[, 1:5]
> # Use probe sets as row names
> dimnames(gi)[[1]] <- gi$probe.set
> dimnames(data)[[1]] <- gi$probe.set
```
Check that the order agrees

We noticed that the order of entries in the sample info file had changed when we merged it with the dChip array information. Just to be on the safe side, we should make sure that the order of the data columns matches the sample info rows.

```r
> sum(dimnames(si)[[1]] != dimnames(data)[[2]])
[1] 0
> sum(dimnames(si)[[1]] == dimnames(data)[[2]])
[1] 72
```
Turn the dChip data into an exprSet

We can bring the dChip quantifications directly into R and turn them into an `exprSet`. Note that this avoids the memory problems by not bringing in the individual CEL files and not producing an AffyBatch.

```r
> dchip <- new('exprSet',
+   exprs=data,
+   phenoData=pd,
+   annotation='hgu95av2',
+   description=miame,
+   notes='processed by KRC in dChip')
> rm(temp, data, si) # cleanup
```
Just RMA

In order to process the data using RMA in BioConductor, we will use the `just.rma` function. This method avoids the memory problems associated with reading all the CEL files into R and keeping them around during processing. Instead, the processing is handed off to a C module that produces an `exprSet` but skips the production of an `AffyBatch` object.
Locating CEL files in multiple directories

The default behavior for the BioConductor routines is to read all the CEL files in the current working directory. If you want to combine files from more than one location (or if you only want to use a subset of the CEL files), then you must first prepare a list of character strings that give the complete names and locations of the files you want.

```r
> # CEL file location is a function of type
> cel.location <- list(ALL='CELFiles',
+ MLL='CELFiles', AML='AMLCELFiles')
> # Use the type of each file to find its location
> celdir <- cel.location[as.character(
+ pd@pData tipo)]
```
> # Paste the '.cel' extension at the end
> celname <- paste(pd@pData$Scan.name,+
+ 'cel', sep='.' )
> # make complete file paths for each file
> all.cel.files <- file.path(home, celdir, celname)
> # peek at the results
> all.cel.files[1:3]
[1] "g:/ShortCourse/CELFiles/CL2001011101AA.cel"
[2] "g:/ShortCourse/CELFiles/CL2001011102AA.cel"
[3] "g:/ShortCourse/CELFiles/CL2001011104AA.cel"
Running Just RMA

> rmaData <- just.rma(filenames=all.cel.files, + phenoData=pd, description=miame)

[1] "Attempting to download hgu95acdf from http://www.bioconductor.org/packages/data/annotation/stable/bin/windows/contrib/2.1"

[1] "Download complete."

[1] "Installing hgu95acdf"

[1] "Installation complete"

Background correcting

Normalizing

Calculating Expression
Unexpected glitches

The two quantified sets have different numbers of genes:

> rmaData
Expression Set (exprSet) with 12626 genes
72 samples
  phenoData object with 13 variables and 72 cases

> dchip
Expression Set (exprSet) with 12625 genes
72 samples
  phenoData object with 13 variables and 72 cases
Figuring out which probe sets differ

```r
> rma.ps <- dimnames(rmaData@exprs)[[1]]
> dchip.ps <- dimnames(dchip@exprs)[[1]]
> setdiff(rma.ps, dchip.ps)
[1] "119_at"    "1215_at"   "1216_at"   "124_i_at"
[5] "125_r_at"  "127_at"    "1301_s_at" "1302_s_at"
[9] "132_at"    "1429_at"   "1502_s_at" "1829_at"
[13] "1864_at"   "1889_s_at" "1982_s_at" "36969_at"
[17] "383_at"    "397_at"    "412_s_at"  "426_at"
[21] "439_at"    "787_at"    "788_s_at"  "972_s_at"
[25] "985_s_at"  "997_at"    "997_at"    "972_s_at"
```
Selecting the common probes

```r
> rmaData <- rmaData[is.element(rma.ps, dchip.ps),]
> rmaData
Expression Set (exprSet) with
12600 genes
72 samples

phenoData object with 13 variables and 72 cases

> rma.names <- dimnames(rmaData@exprs)[[1]]
> dchip@exprs <- dchip@exprs[rma.names,]
> dchip@exprs <- log(dchip@exprs, 2)
> dchip
Expression Set (exprSet) with
12600 genes
72 samples

phenoData object with 13 variables and 72 cases
```
Loading the ClassComparison package

> require(ClassComparison)
Loading required package: ClassComparison
Loading required package: splines
Loading required package: oompaBase
Loading required package: PreProcess
Creating a new generic function for 'plot' in 'PreProcess'
Creating a new generic function for 'print' in 'PreProcess'
Creating a new generic function for 'as.data.frame' in 'PreProcess'

[1] TRUE
T-test, take one

```r
> notAML <- pData@type != 'AML'
> dchip.t <- MultiTtest(dchip[, notAML], 'ALLvMLL')
> dchip.b <- Bum(dchip.t@p.values)
> hist(dchip.b, main='P-values using dChip')
```
T-test, take two

```r
> rma.t <- MultiTtest(rmaData[, notAML], 'ALLvMLL')
> rma.b <- Bum(rma.t@p.values)
> hist(rma.b, main='P-values using RMA')
```
DO the two methods agree?

```r
> plot(rma.t@t.statistics, dchip.t@t.statistics)
> abline(0,1, col='blue')
```