# GS01 0163
# Analysis of Microarray Data

Keith Baggerly and Brad Broom
Department of Bioinformatics and Computational Biology
UT M. D. Anderson Cancer Center
kabagg@mdanderson.org
bmbroom@mdanderson.org

5 October 2010

# Lecture 10: Differential Expression, Borrowing, and Modelling

- Testing Redux, and One More
- Comparing three or more groups
- Pairing
- Incorporating covariates
- Models
- Example

## A Rehash

Comparing two groups:

- t-tests, Wilcoxon tests

Correcting for multiple testing:

- permutation tests
- Bonferroni, BUM and Empirical Bayes

## One More Difference Measure...

Still looking at one gene, and two groups of measurements for that gene

t-tests let us say “these are different”, but do not necessarily let us say anything about “how different are they?”

We can form confidence intervals corresponding for a given difference (eg, diff in log ratios) and convert that confidence interval into another interval on a scale that is more meaningful to us (such as fold change).

## Combining Many Criteria

Now, there's a neat trick that can be used here by combining confidence intervals with the quantity of interest.

Our question till now has been "is this gene differentially expressed between the two groups?", but we can expand this to include another criterion by asking "is this gene differentially expressed between the two groups by at least a minimal amount $k$?"

## The dChip Approach

For each group, assemble point estimates of the expression levels. These point estimates are assumed to have normal distributions. We can then form a confidence interval for the ratio, and we can focus our attention just on those genes where the *lower bound* of this confidence interval is more than $k$-fold. Thus, not only are we pretty sure that the gene is differentially expressed, but we believe that it is different by at least a minimal amount that we can specify.

## Is this the way to go?

I don't necessarily think the dChip answers are right, because I think that their model has the wrong error structure, but I do think that the confidence interval idea has some merit.

It has the practical advantage of using more than one filtering criterion to assess "significance".

Applying Bonferroni requires setting a very wide confidence interval. Permutation tests still work.

## Expanding our Focus

Say we have data from 3 groups that were run at the same time, as opposed to 2. Does this change the outcome of our initial comparison of two groups?

- Given microarray experiments on
  - $N_A$ sample of type $A$
  - $N_B$ sample of type $B$
  - $N_C$ sample of type $C$
- Decide which of the $G$ genes on the microarray are differentially expressed between groups $A$ and $B$.

## Expanding our Focus

The $t$-statistic from before

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

The numerator doesn't change, but what about the denominator?

The pooled estimate of the standard deviation initially includes data from just $A$ and $B$, but it can be expanded to include data from all of the groups

## The Broader Pool...

For two groups:

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

For three groups:

$$s_P^2 = \frac{(N_A - 1)s_A^2 + (N_B - 1)s_B^2 + (N_C - 1)s_C^2}{N_A + N_B + N_C - 3}.$$

## What Does This Buy Us?

- A more precise estimate of the variation gives us more degrees of freedom for the $t$-test.
- More degrees of freedom gives us a more sensitive test.
- Extreme case: $N_A = 2, N_B = 2, N_C = 10$.

  How many differences do we see?

## Some Simulations

No differences in the data...

```
n.genes <- 2000
an <- 2; bn <- 2; cn <- 10
n.samples <- an + bn + cn;
type <- factor(rep(c('A', 'B', 'C'),
              times=c(an, bn, cn)))
data <- matrix(rnorm(n.genes*n.samples),
  nrow=n.genes)
am <- apply(data[, type=='A'], 1, mean)
bm <- apply(data[, type=='B'], 1, mean)
```

## Some Simulations

```
av <- apply(data[, type=='A'], 1, var)
bv <- apply(data[, type=='B'], 1, var)
cv <- apply(data[, type=='C'], 1, var)

sp2.ab  <- ((an-1)*av + (bn-1)*bv)/
                 (an+bn-2)
sp2.abc <- ((an-1)*av + (bn-1)*bv +
                  (cn-1)*cv)/(an+bn+cn-3)
```

## Some Simulations

```
t.stat.ab  <- (bm - am)/
  (sqrt(sp2.ab)*sqrt(1/an+1/bn))
t.stat.abc <- (bm - am)/
  (sqrt(sp2.abc)*sqrt(1/an+1/bn))

p.val.ab <- sapply(t.stat.ab, function(
  tv, df) {
   2*(1-pt(abs(tv), df))
  }, an + bn - 2)
p.val.abc <- sapply(t.stat.abc, function(
  tv, df) {
   2*(1-pt(abs(tv), df))
  }, an + bn +cn - 3)
```

## What Differences Are There?

None.

Added variability makes it harder to see stuff that is there, but not easier to see stuff that isn’t there.

The benefits associated with more precision are linked to increased sensitivity.

## Introduce some Differences

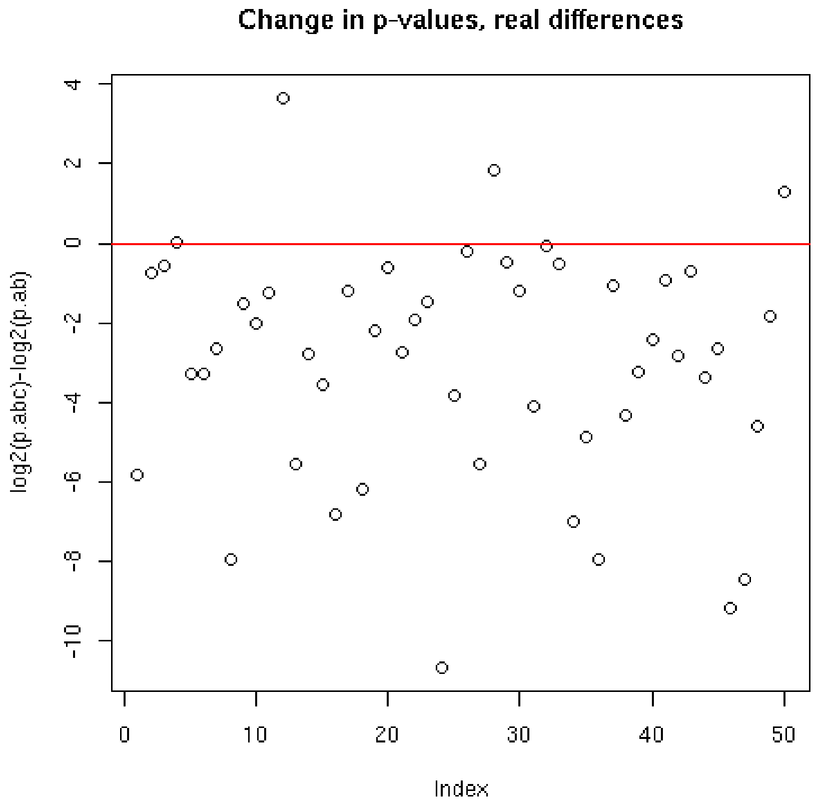
```
data[1:50,type=="A"] <-
    data[1:50,type=="A"] + 3;
```

recompute means, vars, t-values and p-values

```
  sum(p.val.ab  < 0.01); # gives 19
  sum(p.val.abc < 0.01); # gives 45
  sum(p.val.ab[1:50]  < 0.01); # gives  2
  sum(p.val.abc[1:50] < 0.01); # gives 21
```

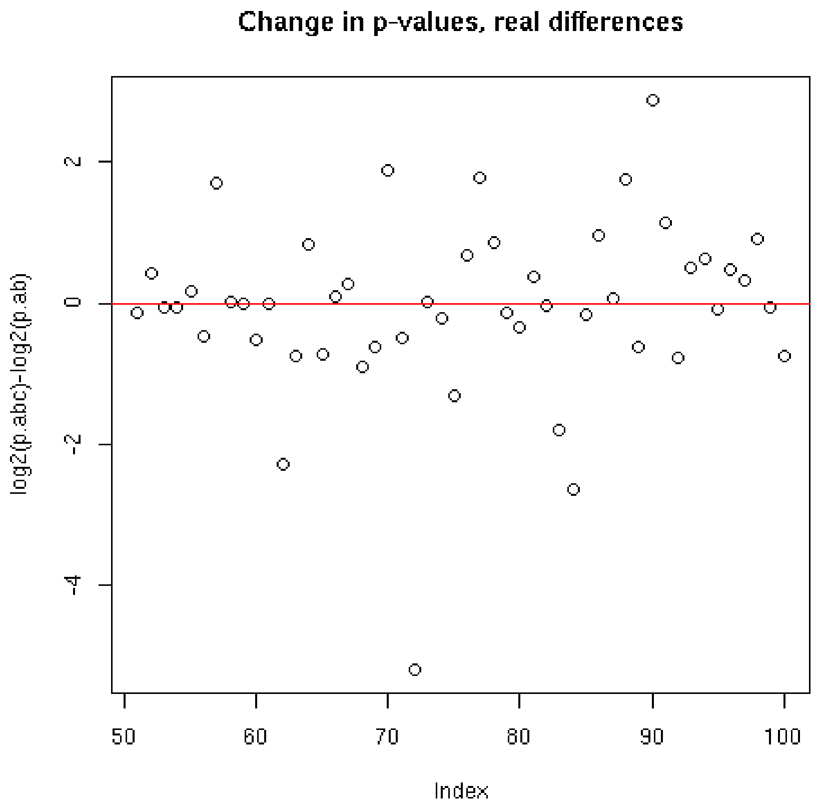
## Plot P-Value Differences

```
plot(-log2(p.val.ab[1:50]) +
      log2(p.val.abc[1:50]), ... );
```



## Plot P-Value Differences

```
plot(-log2(p.val.ab[51:100]) +
      log2(p.val.abc[51:100]), ... );
```



## What Assumptions are We Making?

The variance structures do not change between the three groups (the means can be different).

We are already making this assumption implicitly with the two-sample t-test.

This assumption means that I would restrict the other groups used to those run about the same time, with the same chip lot, etc.

That the data looks approximately normal (work on the log scale).

## Corrections

Rank tests also work.

Bonferroni still works just fine.

BUM still works just fine.

Empirical Bayes still works just fine.

Permutations?

Permute residuals from the null model

## Another Extension: Chip Lot?

Say we have data from arrays from two different lots, 1 and 2, and that we have samples from groups A and B run on arrays from both lots. How should we look at this?

Well, we can still use a two-sample t-test (assuming run order was randomized), but this might break if there are big differences between lots.

(I'll assume for now that the number of samples in each group/lot combination is the same).

## Some more Simulations

```
n.genes <- 2000
a1n <- 2;  b1n <- 2
a2n <- 2;  b2n <- 2
an <- a1n + a2n;  bn <- b1n + b2n;
n.samples <- an + bn;
type <- factor(rep(c('A', 'B'),
     times=c(a1n + a2n, b1n + b2n)))
group <- factor(c(rep(c('G1', 'G2'),
                  times=c(a1n,a2n)),
                  rep(c('G1', 'G2'),
                  times=c(b1n,b2n))));
```

## Add Some Big Differences

```
data <- matrix(rnorm(n.genes*n.samples)
  nrow=n.genes)

data[,group=="G2"] <-
    data[,group=="G2"] + 8;

data[1:50,type=="A"] <-
    data[1:50,type=="A"] + 4;
```

Is this realistic? Can groups overshadow types?

## How do we fit both type and group?

Start with an overall mean

measure deviations associated with type

measure deviations associated with group

```
mu <- apply(data,1,mean);
delta.type  <- apply(
    data[,type=="A"]-mu,1,mean);
delta.group <- apply(
    data[,group=="G1"]-mu,1,mean);
```

## How do we fit both type and group?

fit the data, and sum the squared residuals

```
our.fit <- data;
our.fit[,type=="A" & group=="G1"] <-
  mu + delta.type + delta.group;
our.fit[,type=="A" & group=="G2"] <-
  mu + delta.type - delta.group;
our.fit[,type=="B" & group=="G1"] <-
  mu - delta.type + delta.group;
our.fit[,type=="B" & group=="G2"] <-
  mu - delta.type - delta.group;
```

## Some numbers

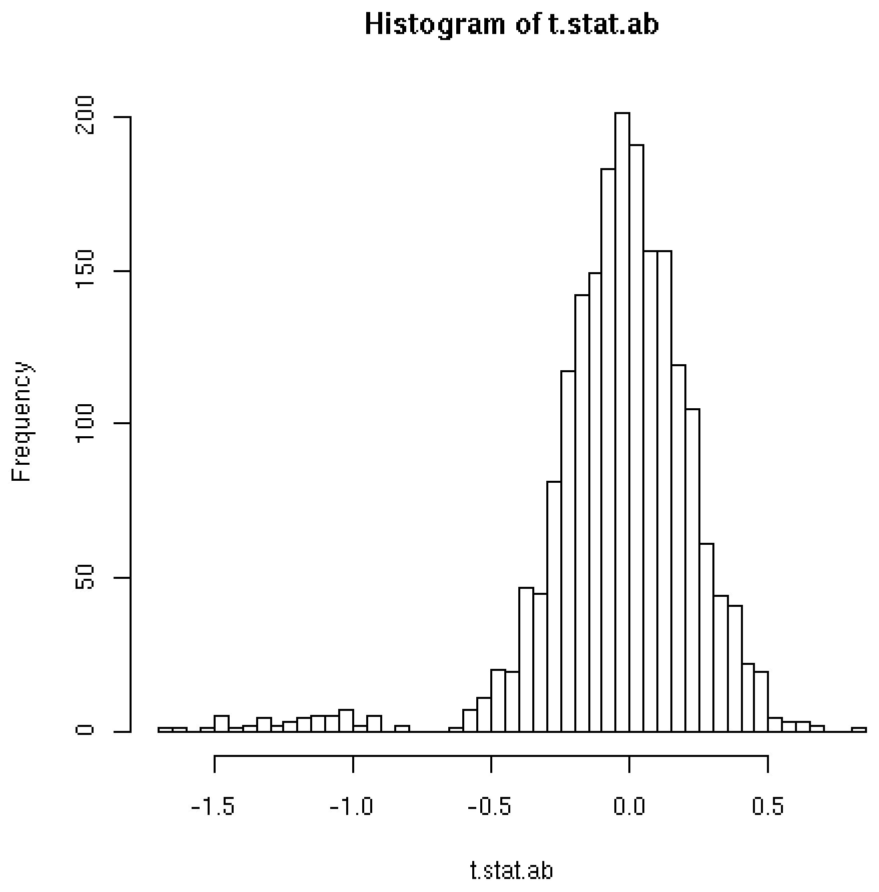
```
> our.resid <- data - our.fit;
> our.se <- sqrt(apply(our.resid^2,
                    1, sum)/5);
> data[1,]
[1]  3.81  3.59 11.11 12.54
[5] -0.67 -0.17  7.05  8.95
> mu[1]
[1] 5.78
> delta.type[1]
[1] 1.99
> delta.group[1]
[1] -4.14
```

## Some numbers

```
> our.fit[1,]
[1]  3.63  3.63 11.90 11.90
[5] -0.35 -0.35  7.93  7.93
> our.resid[1,]
[1]  0.18 -0.04 -0.79 0.64
[5] -0.32  0.17 -0.88 1.03
> our.se[1]
[1] 0.79
our.t.type <- delta.type[1]/
             (our.se[1]/sqrt(8));
```

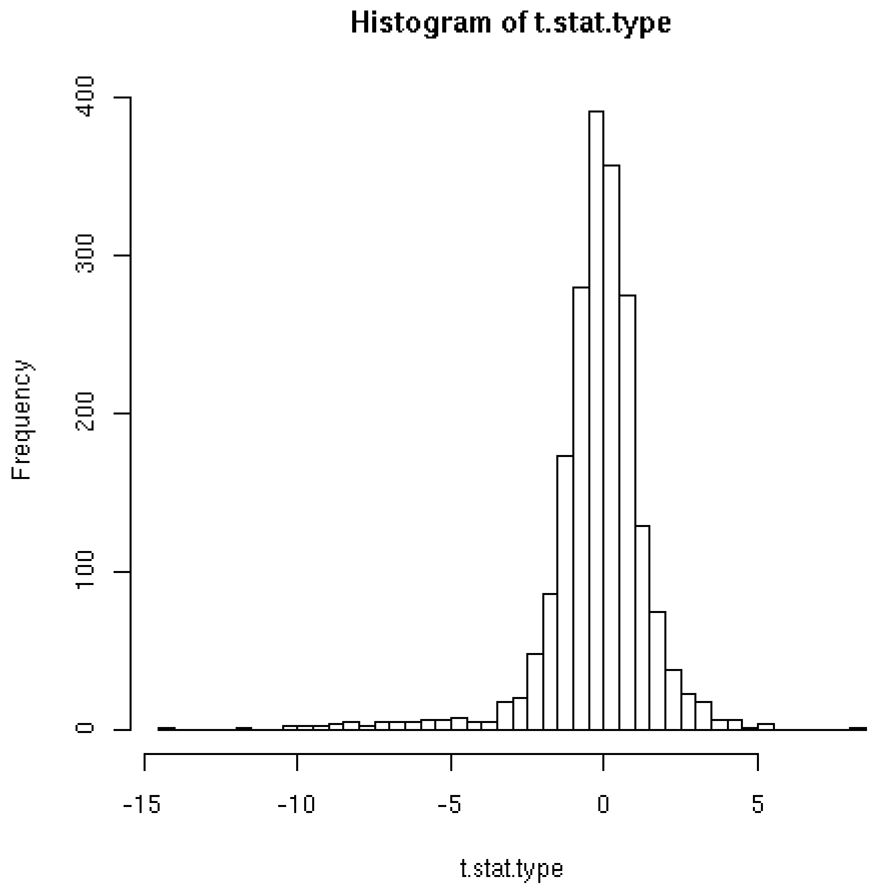
## What do the t-stats look like?

```
hist(t.stat.ab,breaks=50);
```



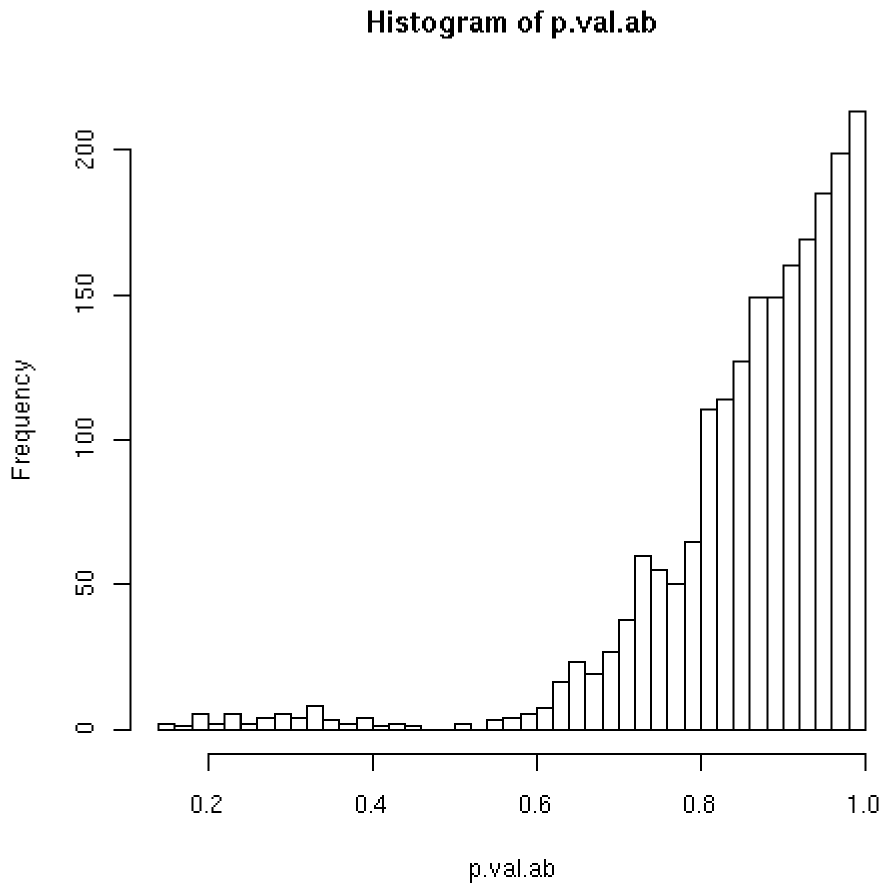
## What do the t-stats look like?

```
hist(t.stat.type,breaks=50);
```



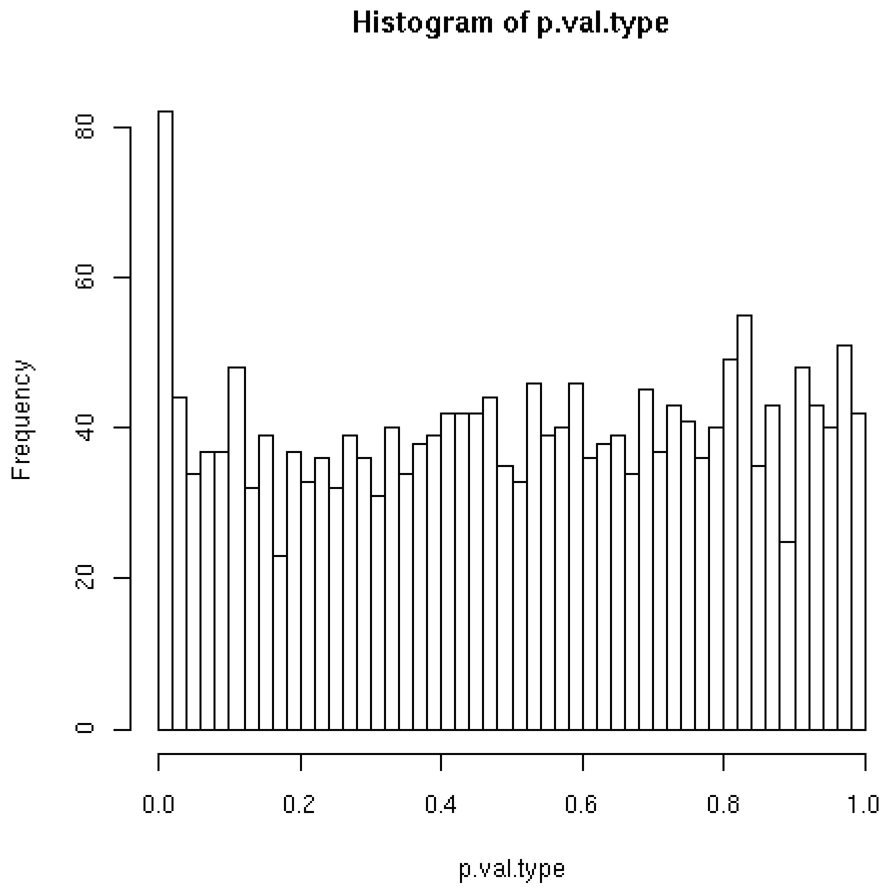
## What do the p-values look like?

```
hist(p.val.ab,breaks=50);
```



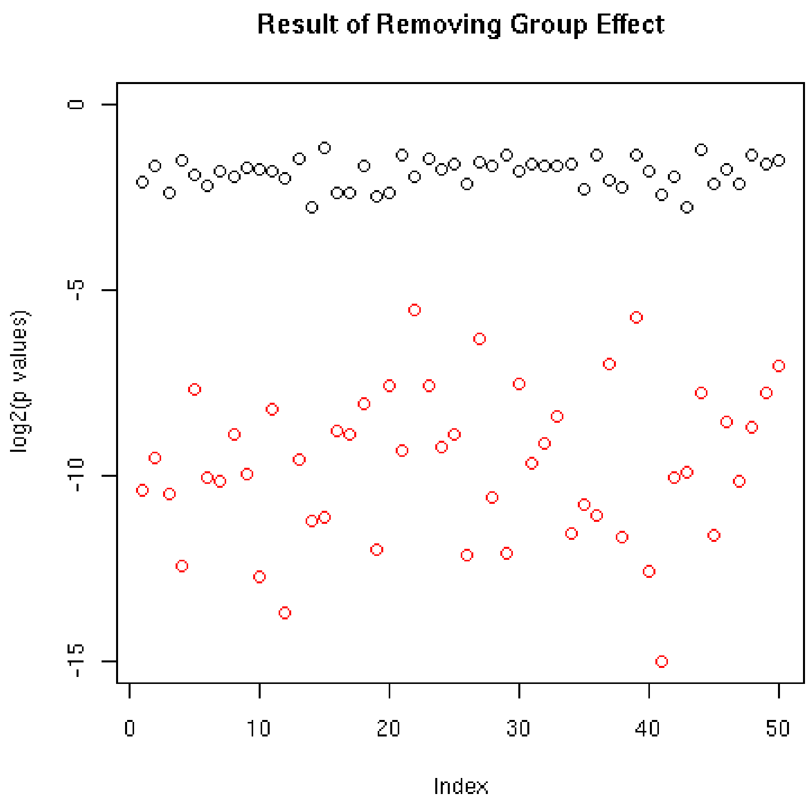
## What do the p-values look like?

```
hist(p.val.type,breaks=50);
```



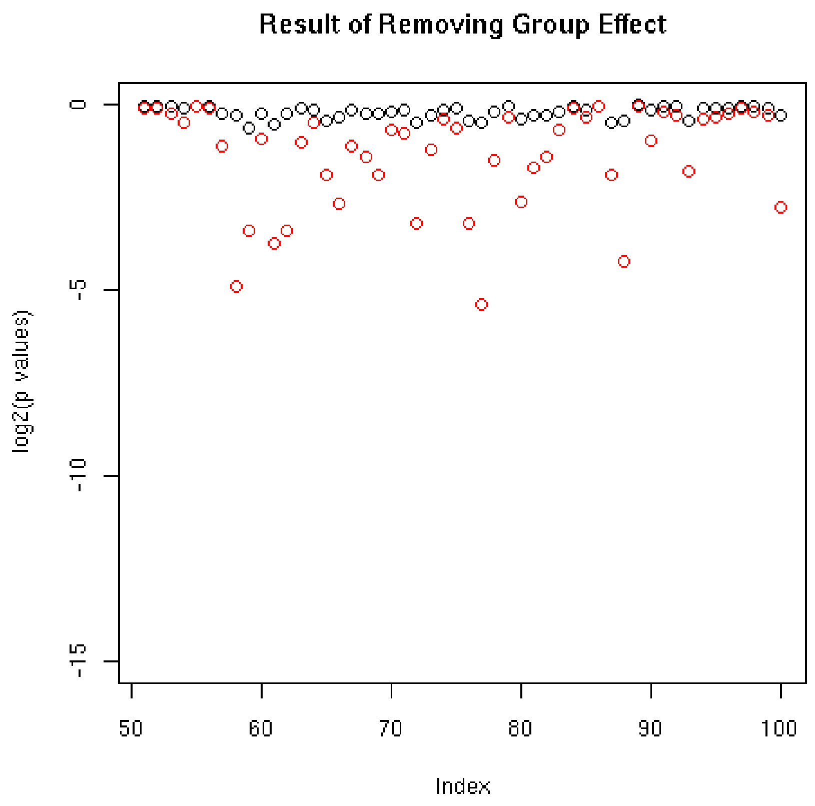
## Changes When Different

```
plot(log2(p.val.ab[1:50]),ylim=c(-15,0),...);
points(log2(p.val.type[1:50]),col='red');
```



## Changes When Different

```
plot(c(51:100),log2(p.val.ab[1:50]),...);
points(c(51:100),log2(p.val.type[1:50]),...);
```



## Partitioning Variance: ANOVA

This general procedure of apportioning the observed variation to the effects that gave rise to it is known as the Analysis of Variance (ANOVA). It was introduced by R.A. Fisher in the 1920s.

Using other groups to stabilize the variance may not be that big a deal. Splitting off variation due to external causes before assessing our effect of interest can be vital.

## ANOVA in R

```
our.lm.1 <- lm(data[1,] ~ type + group);
our.anova.1 <- anova(our.lm.1);
our.anova.1
Analysis of Variance Table

Response: data[1, ]
          Df  Sum Sq Mean Sq F value
type       1  31.557  31.557  52.008
                Pr(>F) 0.000799 ***
group      1 136.818 136.818 225.487
                Pr(>F) 2.372e-05 ***
Residuals  5   3.034   0.607
```

## An Extreme Case: Pairing

In many cases, we have data that are paired: treated/untreated, before/after, primary/metastasis (same patient), or case/control studies matched on a variety of factors.

In this case the math simplifies rather considerably, and we can use a simple one-sample t-test applied to the paired differences:

$$\frac{\bar{x}_A - \bar{x}_B}{sqrt(var(data[A] - data[B])/(n_A - 1))}$$

## The Rank Equivalent: Signed Rank Tests

As with the ANOVA table discussed above, the paired t-test also has a rank analog, arrived at by ranking the differences and applying a sign as A is greater than B or vice-versa. The sum of the positive ranks gives the test statistic.

```
wilcox.test(data[A],data[B],paired=TRUE);
```

## Three Groups, Two Lots?

What if we have both scenarios at once?

Multiple groups, and known external factors?

What is the general rule?

## Including Covariates

The general extension of ANOVA is supplied by the linear model and regression. This was actually used above:

```
our.lm.1 <- lm(data[1,] ~ type + group);
```

where we are fitting the response (data[1,]) as a function of the covariates at hand (type and group). The final significance value is that associated with the effect of interest in the full model.

## Some Standard Factors

What things might we include as explanatory covariates?

- chip lot
- chip
- dye
- run date/order

## The Broader Theme: Modelling

If we know that effects other than the ones we're interested in are likely to be present, it is generally worthwhile to recast our test to explicitly incorporate (and hopefully factor out) these other effects.

This is the idea of modelling the data.

Of course, we can't model everything. When we can't model it, randomize to balance it!

## The Modelling Punchline

Incorporating external information can help sharpen our inferences.

Incorporating such information often goes by the name of modelling, but it can also be viewed as “conditioning on relevant subsets of information”.

The crux of the problem is defining precisely what constitutes a “relevant subset”, which includes what we mean by “relevant”.

## Types of Conditioning

One of the more common types of conditioning is to assume that some other quantity being measured shares some distributional characteristics with measurements of the quantity of interest.

In shorter words, we can use other data to give us better estimates of standard deviations, or the shape of the distribution, or so on. We saw this earlier with the use of a third group of microarray measurements to sharpen inferences about differences between the first two.

## Are Other Genes Relevant?

Are there similar characteristics to microarray measurements of different genes?

If there are, how can we use them?

Most frequently, the answer to the first question is assumed to be yes based on empirical observations. Occasionally, a modelling of the underlying physical processes can further suggest the nature of the similarity.

## An Example

Our first example: normalization.

This can assume either that “most genes don’t change” (single scaling factor normalization) or, more stringently, that “the quantiles of the intensity distributions should be about the same” (loess normalization).

## Are the Assumptions Valid Here?

In general, yes. In checking normalization methods, people have produced some nice-looking smooth curves, but the latter in particular are working under the assumption that if we start with genes of the same rough level of expression, the distributions of values when nothing is going on will be about the same.

## Other Extensions of Borrowing

borrowing strength on the p-value scale.

BUM

Empirical Bayes

## Extending this idea to diff. exp.

What can we do here?

Say that we have our standard question of trying to compare the levels of a given gene in two different groups, $A$ and $B$.

How can we change the t statistic?

As before, our best guess about the central value of the gene in each of the groups is driven by the observed values for that gene:

$\bar{x}_A - \bar{x}_B$ is unchanged.

## Pooling variance estimates

What can use to improve our estimate of the variance?

▮

How about the variance of all of the genes?

▮

This is likely to be too much.

What if we just use the genes that are close by in terms of overall (average) intensity?

This type of procedure makes some of the same underlying assumptions as the loess normalization, which also works with “locally similar” data.

## What does this produce?

a stabilized variance and a “smooth” t-test.

This idea has been independently reintroduced in several forms.

Baldi and Long (2001) use a Bayesian approach to trade off between the sample variance for the gene of interest and the pooled variance estimate. This is known as a “shrinkage” estimate.

Newton et al (2001) use a Gamma-Poisson model which achieves the same effect.

## Some More Papers

The “fudge factor” in the denominator of SAM is of this variety.

Baggerly et al (2001) use a Beta-binomial model based on the use of variance derived from replicate spottings to derive the the locally pooled variance estimate; there is no weighting tradeoff with the actual variance observed.

## Are We Using It?

This last paper is the basis for some of the “standard analyses” done at MD Anderson. All of the above tests were developed in the context of cDNA microarrays.

We’ve also used it to analyze data from nylon membrane arrays (Coombes, 2001).

## Why might the assumption be valid here?

There are plausible reasons why the variance of microarray readings should change in a smooth fashion as the overall intensity increases.

These have to do with lognormal expression values, background subtraction, and thresholding.

But we're implicitly assuming that "most genes aren't too correlated" so a variance estimate derived from several genes will be close.

## Implications of Independence

We note that this assumption of independence means that in terms of trying to define the overall variance distribution, it is not a good idea to choose a bunch of genes known to be biologically related as our relevant subset. It is interesting to explore these connections, but here we are seeking reinforcement of a story by looking for groups of genes having similar expression patterns.

## Putting Some Pieces Together

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

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

This is an ExpressionSet 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”.

## Skimming the Data

```
> class(ALL)
[1] "ExpressionSet"
attr(,"package")
[1] "Biobase"
> slotNames(ALL)
[1] "assayData"  "phenoData"  "featureData"
[4] "experimentData"  "annotation"
[6] ".__classVersion__"
> phenoData(ALL)
  sampleNames: 01005, 01010, ..., LAL4 (128 total
  varLabels and varMetadata:
    cod:  Patient ID
    diagnosis:  Date of diagnosis...
```

## Learning about the Experiment

```
> experimentData(ALL)
Experiment data
  Experimenter name: Chiaretti et al.
  Laboratory: Department of Medical Oncology, Dana
  Contact information:
  Title: Gene expression profile of adult T-cell a
  URL:
  PMIDs: 14684422 16243790

  Abstract: A 187 word abstract is available. Use
> abstract(ALL)
[1] "Gene expression profiles were examined in 33
```

## Picking Something to Focus On

```
> varLabels(phenoData(ALL))
 [1] "cod"  "diagnosis"  "sex"   "age"  "BT"
 [6] "remission"   "CR"  "date.cr"   "t(4;11)"
[10] "t(9;22)"  "cyto.normal"  "citog"  "mol.biol
[14] "fusion protein"  "mdr"  "kinet"  "ccr"
[18] "relapse"  "transplant"  "f.u"  "date last se
> table(phenoData(ALL)$mol.biol)

ALL1/AF4  BCR/ABL E2A/PBX1  NEG  NUP-98  p15/p16
      10       37        5   74       1        1
```

## Subsetting the Group

```
> 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.) We tend to use MultiTtest from the ClassComparison package available on our website, but that’s only because we wrote it.

Unfortunately...

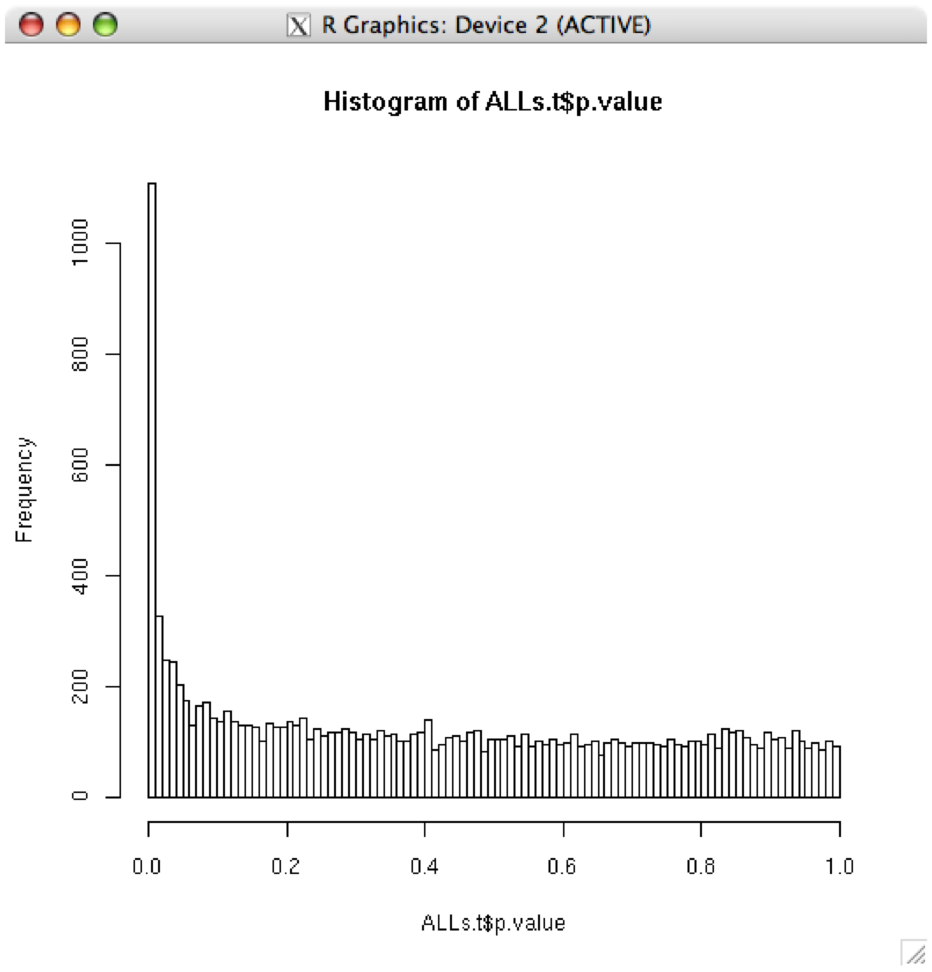
## Looking for Differences

```
> 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...

```
> 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?

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));
[1]   0 111
```

```
> rowMeans(exprs(ALLs))[1:3]
  1000_at   1001_at 1002_f_at
 7.565085  5.019850  3.884797
```

## Some Filtering (Take Logs!)

```
> meanThresh <- log2(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 <- log2(200);
...
> dim(exprs(ALLfilt)); # 1771 by 111, better
```

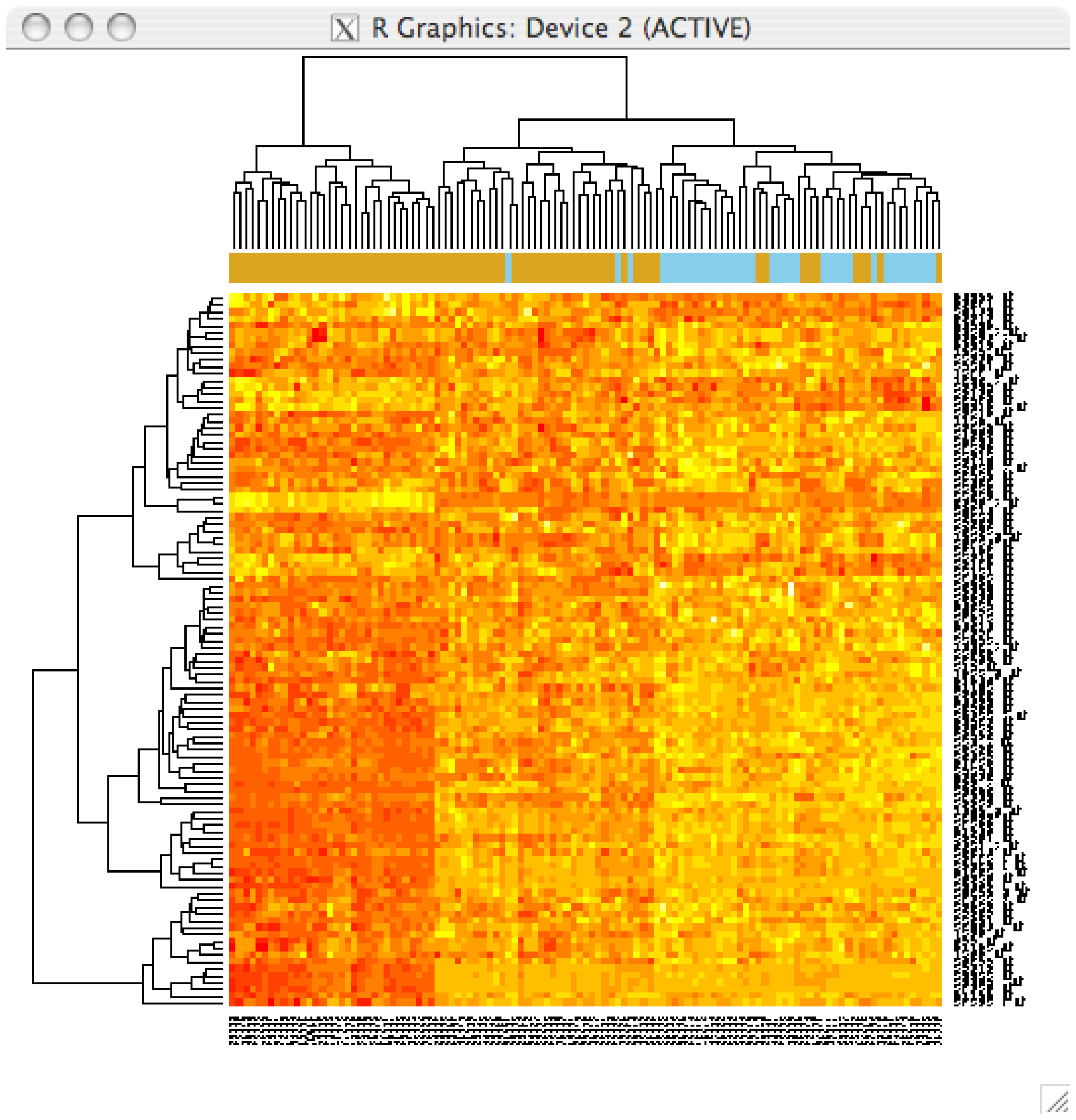
## Focus on the Interesting Ones

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

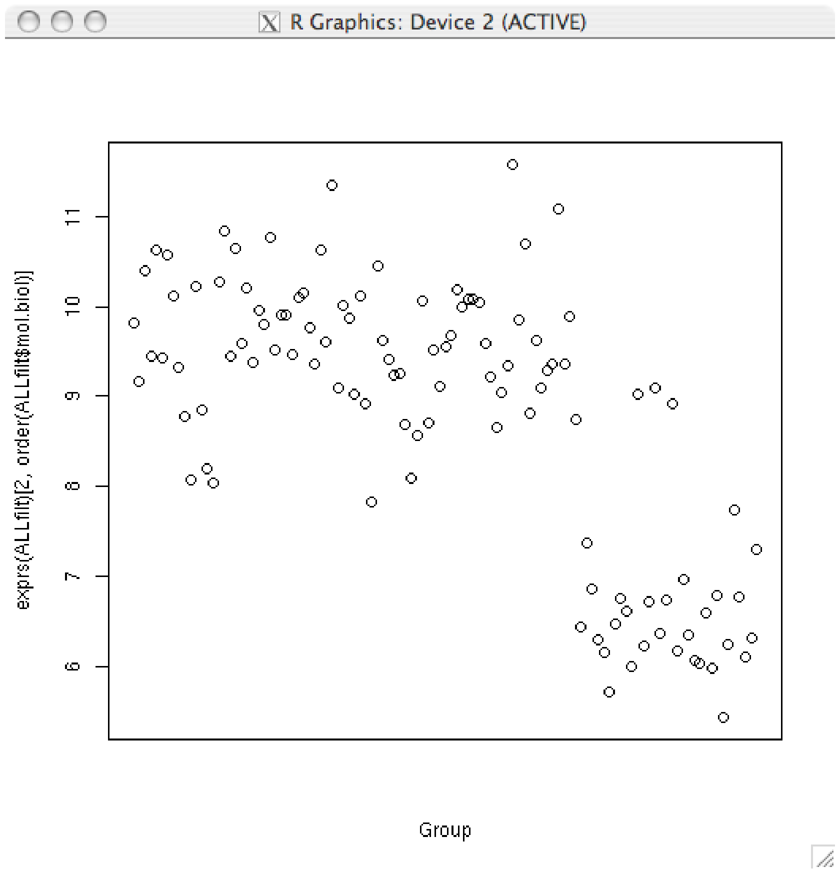
Try picturing this...

```
> spcol <- ifelse(ALLfilt$mol.biol == "NEG",
+                 "goldenrod", "skyblue")
> heatmap(exprs(ALLfilt), ColSideColors=spcol);
```

## Huzzah! (Right?)

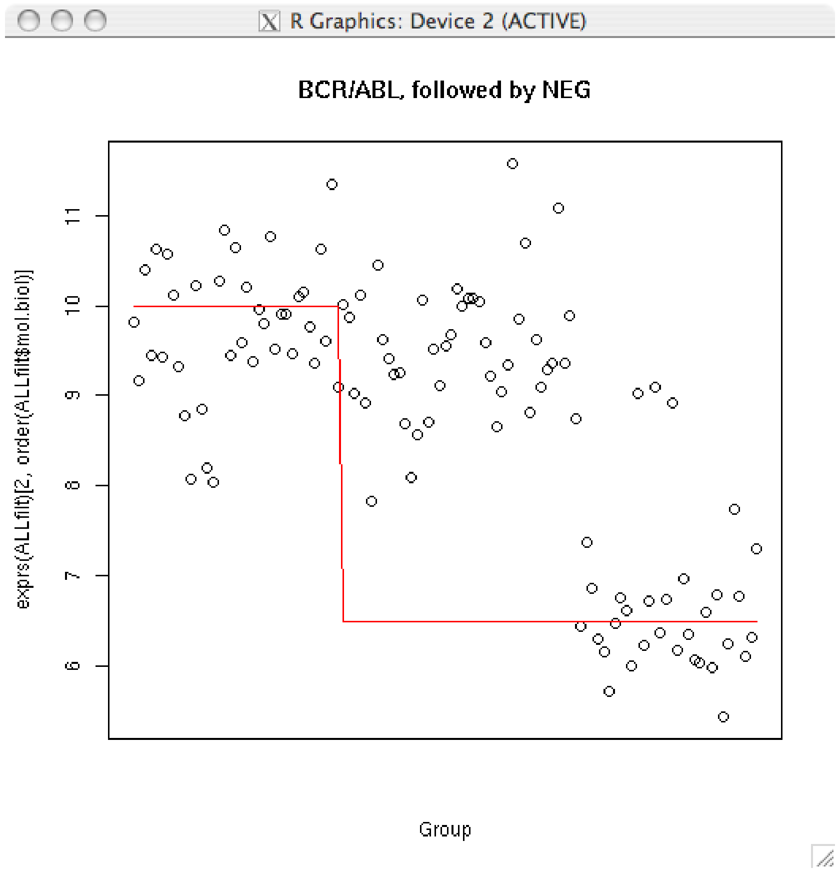


## That Was Odd...



```
> plot(exprs(ALLfilt)[2,order(ALLfilt$mol.biol)],
    xaxt='n', xlab='Group'); # row 1 was boring.
```

## That Was Odd... Right?



```
> lines(10 - 3.5*(ALLfilt$mol.biol[order(
    ALLfilt$mol.biol)] == '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?

```
> ALLfilt$BT
  [1] B2 B2 B4 B2 B1 B1 B1 B2 B2 B3
 [11] B3 B2 B3 B  B2 B3 B2 B3 B2 B2
 [21] B2 B1 B2 B2 B2 B  B  B2 B2 B2
 [31] B2 B2 B2 B2 B2 B4 B2 B2 B2 B4
 [41] B2 B2 B3 B3 B3 B3 B4 B3 B3 B1
 [51] B1 B3 B3 B3 B3 B3 B3 B3 B3 B3
 [61] B1 B2 B2 B1 B3 B4 B4 B2 B2 B3
 [71] B4 B4 B4 B2 B2 B2 B1 B2 B  T
 [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
```

## Another View

```
> table(pData(ALLfilt)$BT, pData(ALLfilt)$mol.biol
     BCR/ABL NEG
  B        2   2
  B1       1   8
  B2      19  16
  B3       8  14
  B4       7   2
  T        0   5
  T1       0   1
  T2       0  15
  T3       0   9
  T4       0   2
```

## Matching Patterns

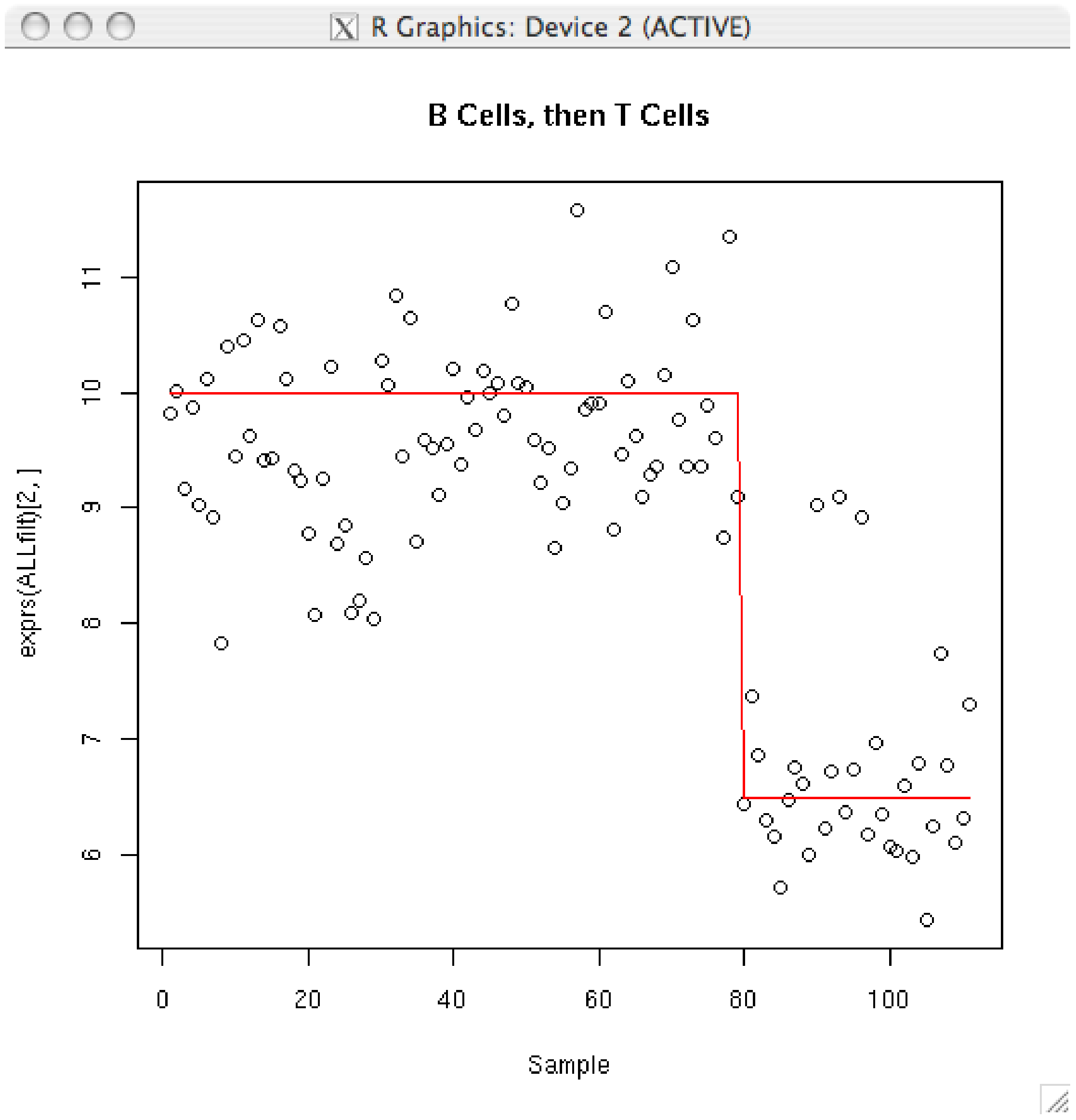
We want entries that begin with B. This is a “regular expression”, and one of the tools for extracting these is “grep”.

```
> BT <- as.character(ALLfilt$BT);
> grep("B", BT); # returns 1..79
> grep("ˆB", BT); # same
> grep("ˆT", BT); # 80..111
> grep("B*", BT); # 1..111 everything!
> grep("B.*",BT); # 1..79
> grep("B$", BT); # 14,26,27,79
> grep("ˆB$",BT); # same
> grep("ˆb", BT); # null
> grep("ˆb", BT, ignore.case=TRUE);
```

## Once More Unto the Breach!

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

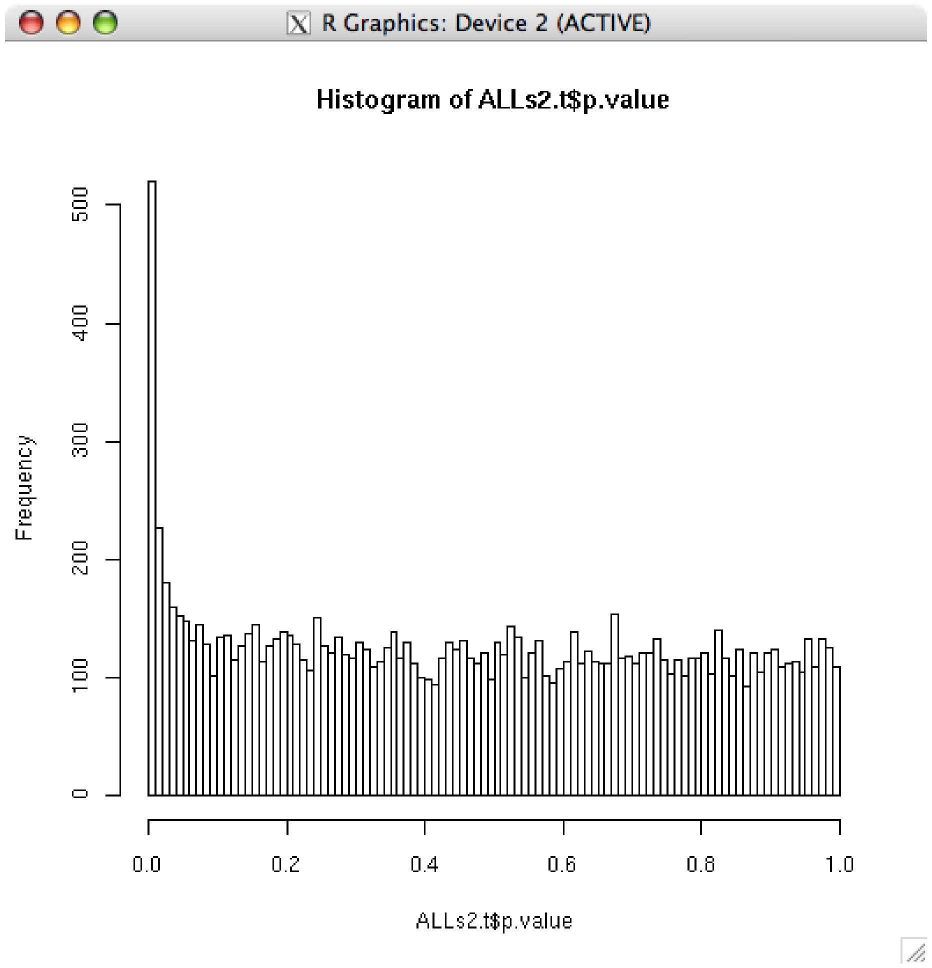
## Finally!


R Graphics: Device 2 (ACTIVE)
B Cells, then T Cells
exprs(ALLfilt)[2, ]
Sample
0
20
40
60
80
100
6
7
8
9
10
11

## Analysis Redux 1

```
> mySubset1 <- grep("^B", ALL$BT);
> ALLs1 <- ALL[,mySubset1];
> dim(exprs(ALLs1))
[1] 12625    95
> mySubset2 <- ALLs1$mol.biol %in% c("BCR/ABL", "N
> 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);
```

## Analysis Redux 2

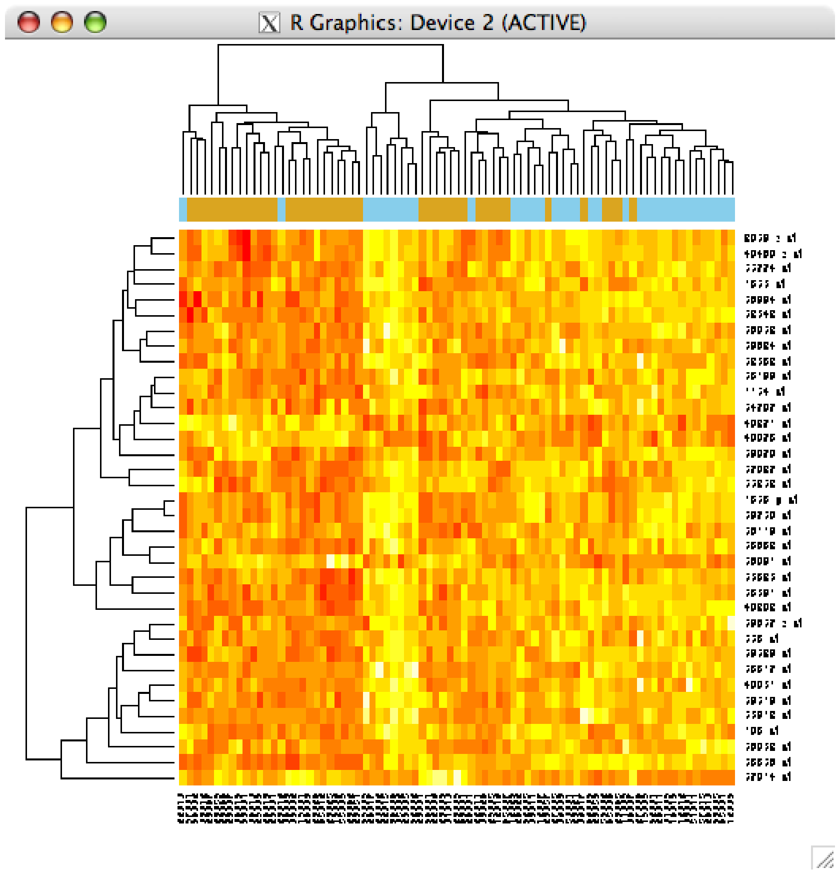


```
> hist(ALLs2.t$p.value, breaks=100);
```

## Analysis Redux 3

```
> 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))
[1]  36 79
```

## A Better Figure



```
> 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.

```
> featureNames(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 <- featureNames(ALLs2Filt)[index]
> probeids[1:3]
[1] "1636_g_at" "39730_at"  "1635_at"
```