Basic Statistical Analysis of Array Data

BST 226
Statistical Methods for Bioinformatics
David M. Rocke

Normalization

- Sometimes even single-analyte assays are normalized.
- Measure TNF-α with a western blot using optical density of the band
- Sometimes "housekeeping" proteins like β-actin or GAPDH (Glyceraldehyde 3-phosphate dehydrogenase) are used to normalize and account for variations in the amount of protein loaded

$$y_{\text{TNF}} = \mu + a_P + b_{\text{TNF}} + \epsilon_{\text{TNF}}$$

$$y_{\text{Actin}} = \mu + a_P + c_{\text{Actin}} + \epsilon_{\text{Actin}}$$

$a_P = N(0, \sigma_P^2)$ Variability due to protein loading

$\epsilon = N(0, \sigma_\epsilon^2)$ Measurement error

$$y_{\text{TNF}} = N(\mu + b_{\text{TNF}}, \sigma_P^2 + \sigma_\epsilon^2)$$

$$y_{\text{TNF}} - y_{\text{Actin}} = N(b_{\text{TNF}}, 2\sigma_\epsilon^2)$$

Normalization is better when the error due to protein loading is greater than the measurement error

Multi-analyte Normalization

- In a western blot, we measure one or a few analytes and perhaps a loading control like β-actin
- In other assays we measure many analytes and there may be no control, or none we want to use for normalization
- Instead, we may use some measure of the overall response of the sample to normalize.
- For example, we may compute the mean or median value across analytes for each sample (M_i) and the overall mean or median M of the M_i across samples, and then normalize the value y_{ij} for analyte j from sample i to $y_{ij} - M_i + M$.
- For gene expression arrays, we often normalize in an intensity dependent way so that the averages are only for genes with similar spot intensities; this avoids level-dependent biases.

Normalization methods

- Use of the mean or sum can cause trouble because this may be driven completely by a few large values
- Thus, total ion current for mass spec is not a good normalization method even though it is a good measure of the total throughput
- We often use the median across the sample for a small number of analytes as in Luminex
- We use lowess smoothing for expression arrays—this normalizes across regions of similar intensity.
- `rma()` uses quantile normalization, which makes each array have the same values, just in a different order

Background correction

- If the target transcript is not present in the sample, the spot will still fluoresce.
- This is due to things like non-specific hybridization
- We can try to adjust for this by subtracting an estimate of background from the value on each spot (and then adding back the average background)
- This is not as important as some other adjustments.

Data Transformations.

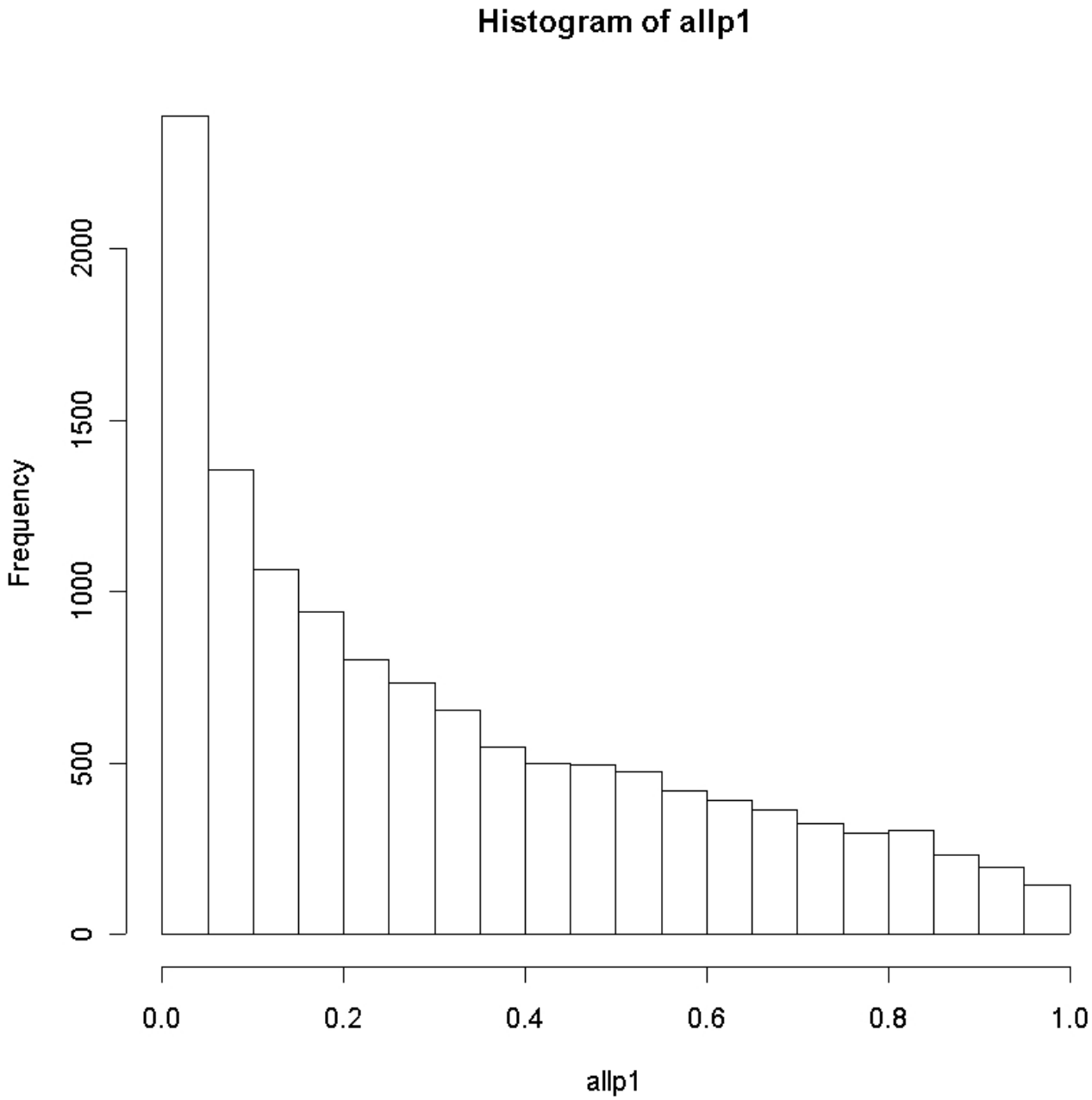
- In a gene expression array, and in other assays, the variance rises generally with the mean.
- For high level data, the log will stabilize the variance.
- For low level data, this causes problems
- Good transformations include the generalized log and the started log.
- Often, for Affymetrix data, the `rma()` method is good enough, though it does not stabilize the variance as well.

Fitting a model to genes

- We can fit a model to the data of each gene after the whole arrays have been background corrected, transformed, and normalized, for example by `rma()`.
- Each gene is then test for whether there is differential expression
- Significance levels are determined in the usual way, or we can "borrow strength" from other genes if the sample size is small.

```
> dim(exprs(eset))
[1] 12625    12
> exprs(eset)[942,]
LN0A.CEL LN0B.CEL LN1A.CEL LN1B.CEL LN2A.CEL LN2B.CEL LN3A.CEL LN3B.CEL
9.063619 9.427203 9.570667 9.234590 8.285440 7.739298 8.696541 8.876506
LN4A.CEL LN4B.CEL LN5A.CEL LN5B.CEL
9.425838 9.925823 9.512081 9.426103
> group <- as.factor(c(0,0,1,1,2,2,3,3,4,4,5,5))
> group
 [1] 0 0 1 1 2 2 3 3 4 4 5 5
Levels: 0 1 2 3 4 5
> anova(lm(exprs(eset)[942,] ~ group))
Analysis of Variance Table

Response: exprs(eset)[942, ]
          Df Sum Sq Mean Sq F value   Pr(>F)
group      5 3.7235  0.7447  10.726 0.005945 **
Residuals  6 0.4166  0.0694
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

```
getp <- function(y)
{
  tmp <- anova(lm(y ~ group))$P[1]
  return(tmp)
}

allp <- function(array)
{
  tmp2 <- apply(array,1,getp)
  return(tmp2)
}

> source("allgenes.r")
> allp1 <- allp(exprs(eset))
> length(allp1)
[1] 12625
```

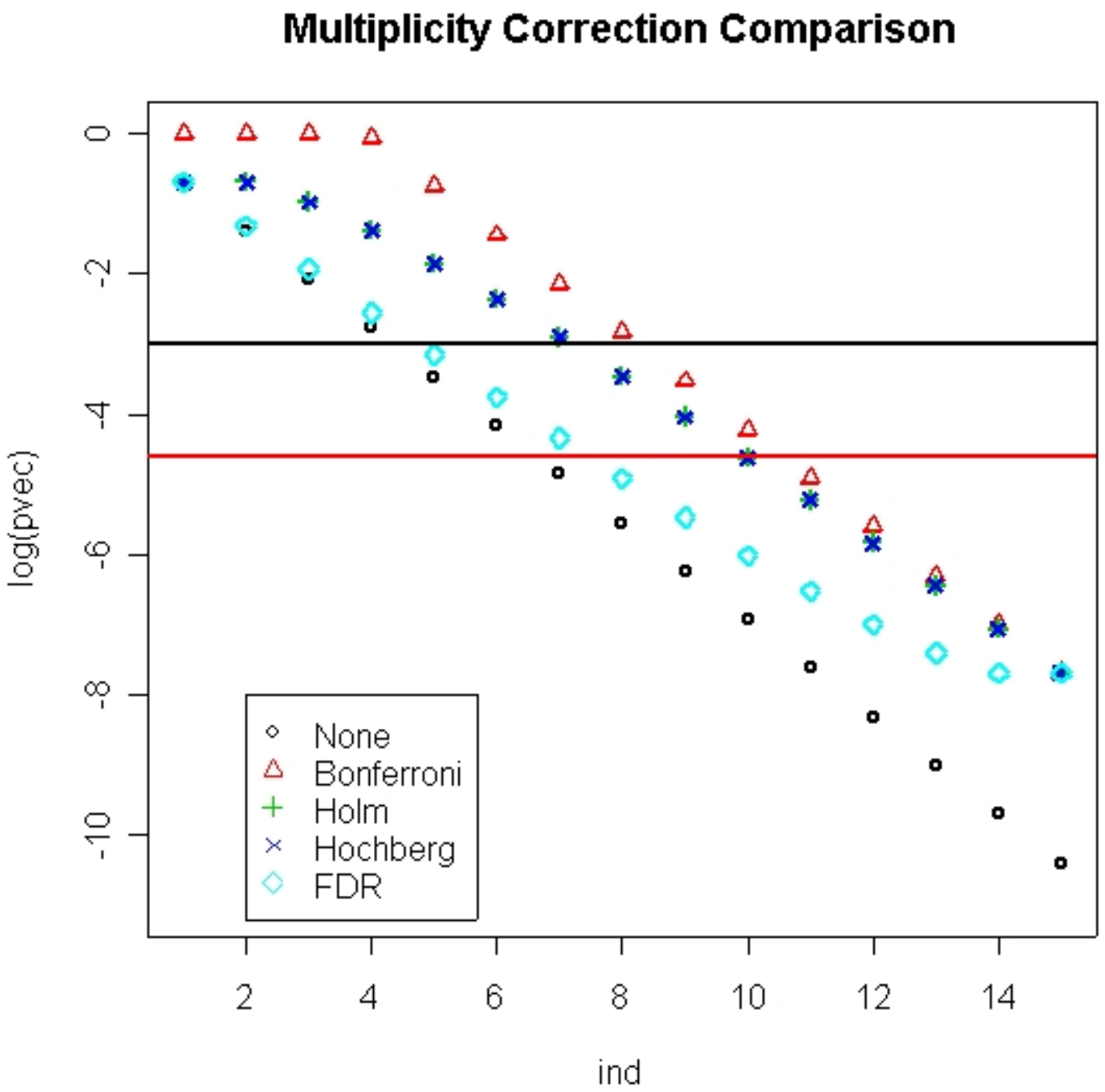

Histogram of allp1
Frequency
0
500
1000
1500
2000
0.0
0.2
0.4
0.6
0.8
1.0
allp1

Multiplicity Adjustments

- If we test thousands of genes and pick all the ones which are significant at the 5% level, we will get hundreds of false positives.
- Multiplicity adjustments winnow this down so that the number of false positives is smaller

Types of Multiplicity Adjustments

- The Bonferroni correction aims to detect no significant genes at all if there are truly none, and guarantees that the chance that any will be detected is less than .05 under these conditions
- Generally, this is too conservative
- Less conservative versions include methods due to Holm, Hochberg, and Benjamini and Hochberg (FDR)


Multiplicity Correction Comparison
log(pvec)
0
-2
-4
-6
-8
-10
2
4
6
8
10
12
14
ind
None
Bonferroni
Holm
Hochberg
FDR

```
> allp1adj <- p.adjust(allp1,"fdr")
> sum(allp1adj<.05)
[1] 119
> featureNames(eset)[allp1adj < .05]
  [1] "120_at"                          "1288_s_at"
  [3] "1423_at"                         "1439_s_at"
  [5] "1546_at"                         "1557_at"
.....................................................
[101] "41058_g_at"                      "411_i_at"
[103] "41206_r_at"                      "41501_at"
[105] "41697_at"                        "41733_at"
[107] "476_s_at"                        "613_at"
[109] "646_s_at"                        "672_at"
[111] "769_s_at"                        "777_at"
[113] "801_at"                          "922_at"
[115] "952_at"                          "AFFX-BioB-M_at"
[117] "AFFX-HUMGAPDH/M33197_3_at" "AFFX-M27830_5_at"
[119] "AFFX-M27830_M_at"
```

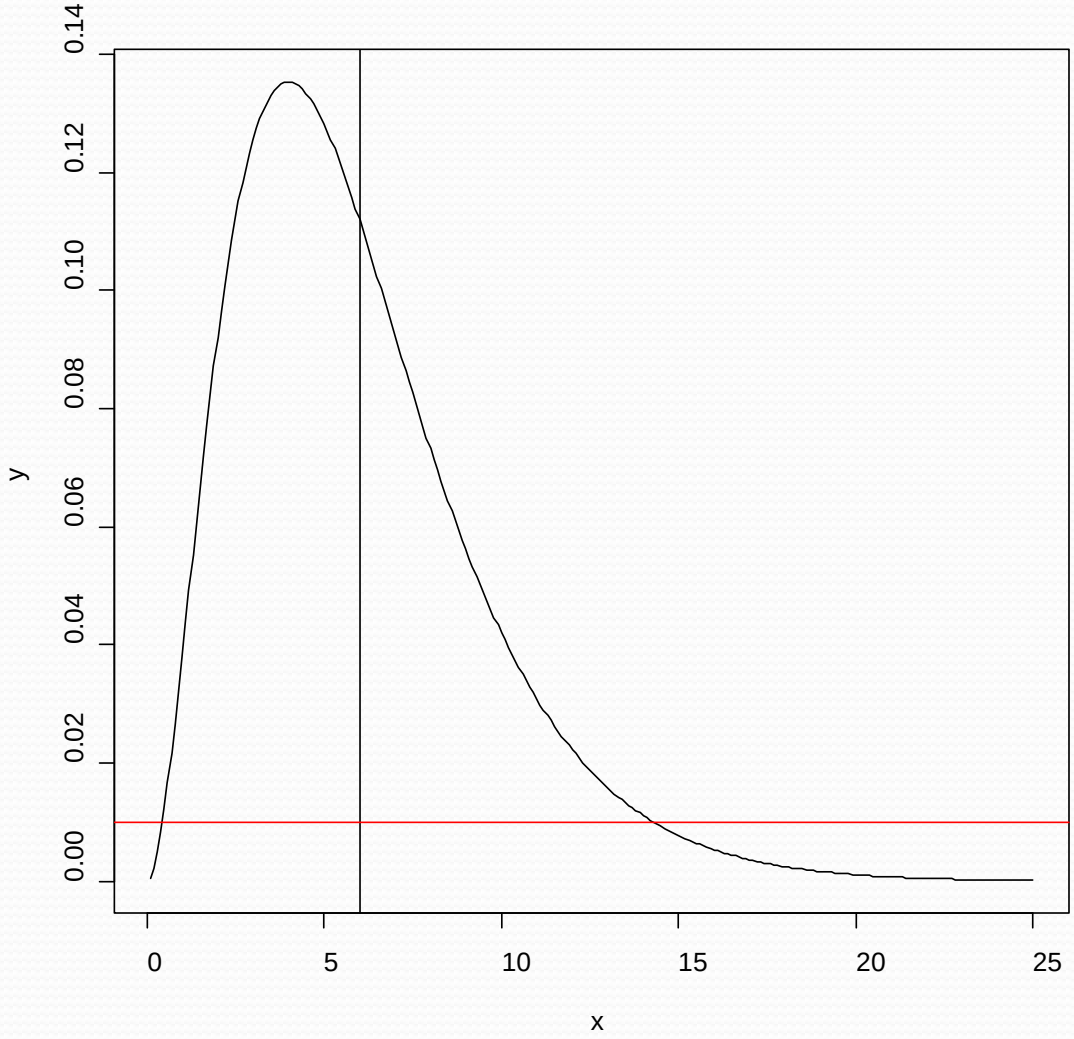
LMGene

- LMGene is a Bioconductor package for linear model analysis of gene expression data.
- It can duplicate the small program which does a one-way ANOVA for each gene, or any other linear model.
- It also can compute the “moderated” t or F statistic, in which small denominators are made larger and large denominators are made smaller.
- Install using BiocLite() in R.

Moderated Statistics

- If we conduct a one-way ANOVA for each of 12625 genes, then each F-statistic uses the 6df denominator which estimates the true MSE.
- We can do better if we assume that the true MSE varies from gene to gene, but not arbitrarily.

Distribution of denominators with 6df when the true MSE is 6

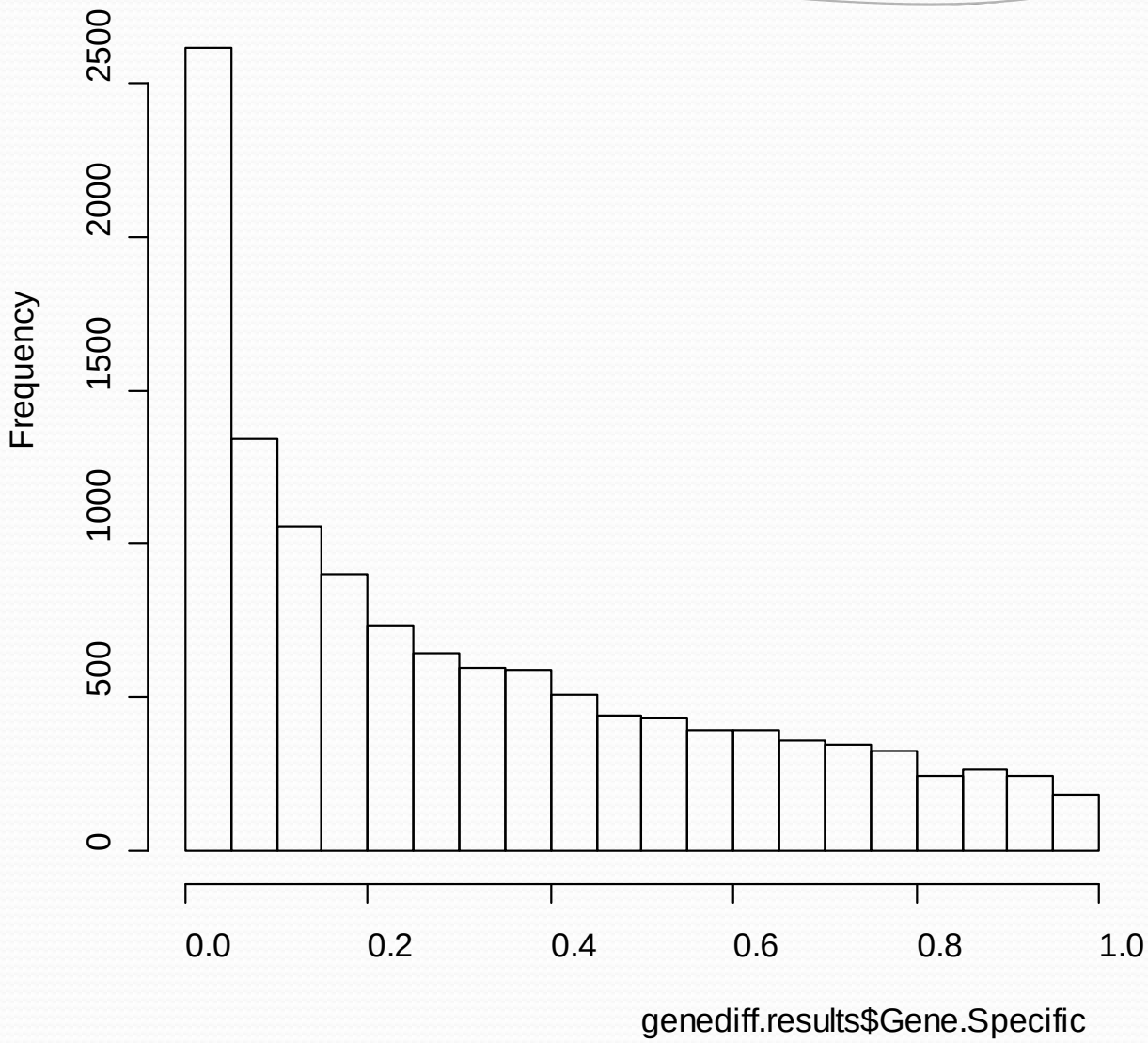


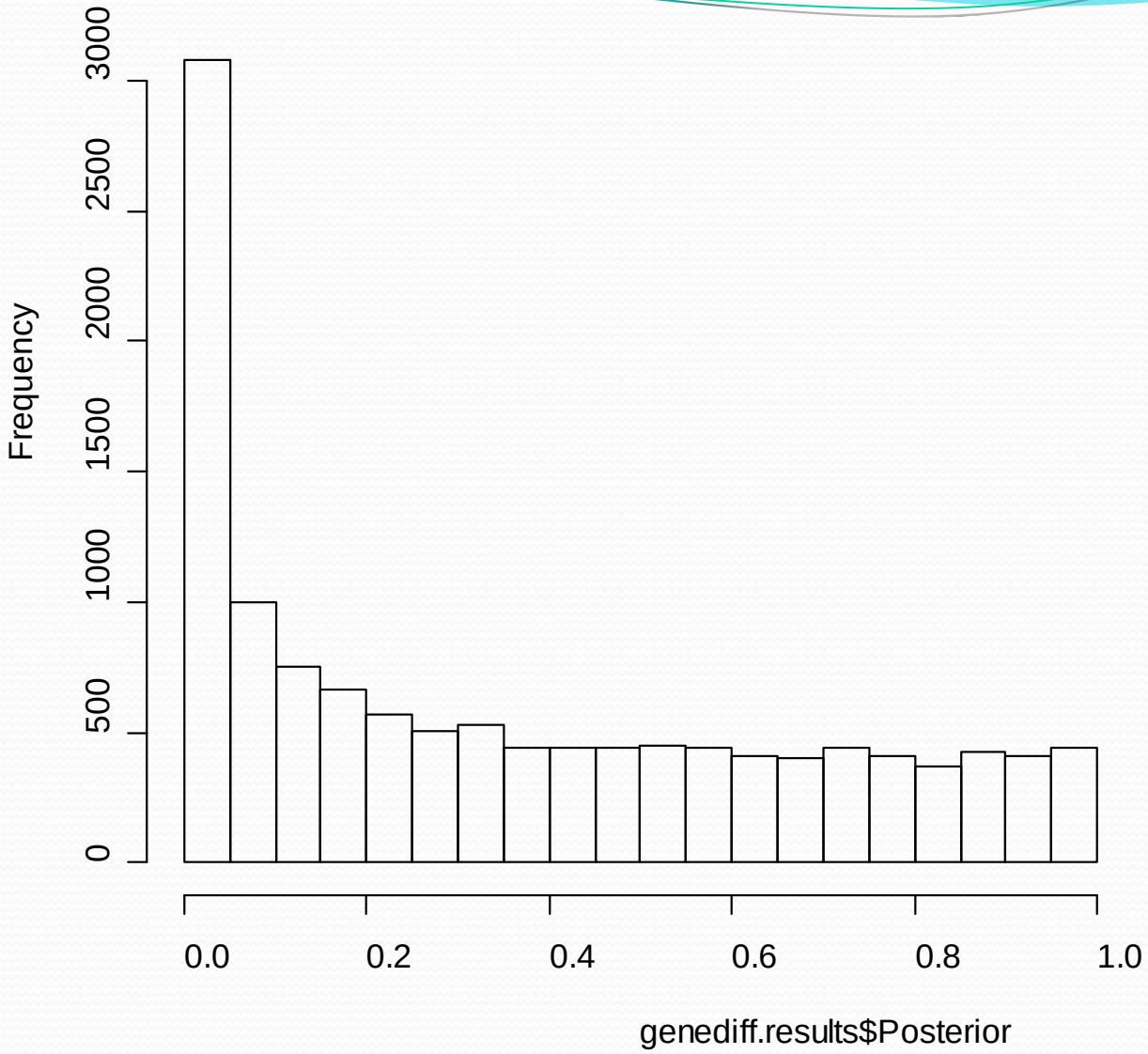
```
> colnames(exprs(eset))
 [1] "LN0A.CEL" "LN0B.CEL" "LN1A.CEL" "LN1B.CEL" "LN2A.CEL" "LN2B.CEL"
 [7] "LN3A.CEL" "LN3B.CEL" "LN4A.CEL" "LN4B.CEL" "LN5A.CEL" "LN5B.CEL"

> group <- factor(c(0,0,1,1,2,2,3,3,4,4,5,5))
> vlist <- list(group=group)
> vlist
$group
 [1] 0 0 1 1 2 2 3 3 4 4 5 5
Levels: 0 1 2 3 4 5

> eset.lmg <- neweS(exprs(eset),vlist)
> lmg.results <- LMGene(eset.lmg)
```

This results in a list of 1173 genes that are differentially expressed after using the moderated F statistic. Compare to 119 if the moderated statistic is not used. We will see later how to understand the biological implications of the results

```
> genediff.results <- genediff(eset.lmg)
> names(genediff.results)
[1] "Gene.Specific" "Posterior"
> hist(genediff.results$Gene.Specific)
> hist(genediff.results$Posterior)
> pv2 <- pvadjust(genediff.results)
> names(pv2)
[1] "Gene.Specific"     "Posterior"          "Gene.Specific.FDR"
[4] "Posterior.FDR"
> sum(pv2$Gene.Specific < .05)
[1] 2615
> sum(pv2$Posterior < .05)
[1] 3082
> sum(pv2$Gene.Specific.FDR < .05)
[1] 119
> sum(pv2$Posterior.FDR < .05)
[1] 1173
```

Using genediff results in two lists of 12625 p-values. One uses the standard 6df denominator and the other uses the moderated F-statistic with a denominator derived from an analysis of all of the MSE's from all the linear models.


Histogram of genediff.results$
Frequency
0
500
1000
1500
2000
2500
0.0
0.2
0.4
0.6
0.8
1.0
genediff.results$Gene.Specific


Histogram of genediff.results$
Frequency
0
500
1000
1500
2000
2500
3000
0.0
0.2
0.4
0.6
0.8
1.0
genediff.results$Posterior

Using LMGene for More Complex Models

- The eS object contains a matrix of data and a list of variables that can be used for the linear model
- An optional second argument is the linear model that is fit to the data.
- The default is to use all the variables as main effects with no interactions.

- Suppose the data consist of 32 arrays from 8 patients at each of 4 doses (in this case of ionizing radiation) of 0, 1, 10, and 100 cGy.
- We specify each variable, make a list for the eS, and then write the model if necessary.

```
> patient <- factor(rep(1:8,each=4))
> patient
 [1] 1 1 1 1 2 2 2 2 3 3 3 3 4 4 4 4 5 5 5 5 6 6 6 6 7 7 7 7 8 8 8 8
Levels: 1 2 3 4 5 6 7 8
> dose <- rep(c(0,1,10,100),8)
> dose
 [1]   0   1  10 100   0   1  10 100   0   1  10 100   0   1  10 100
[17] 0   1  10 100   0   1  10 100   0   1  10 100   0   1  10 100
> vlist <- list(patient=patient, dose=dose)
> eset.rads <- neweS(exprs(eset),vlist)
> rads.results <- LMGene(eset.rads)
> rads.results <- LMGene(eset.rads,'patient+dose')
> rads.results <- LMGene(eset.rads,'patient*dose')
```

The + operator means an additive model, the * operator means the factors/variables and all interactions, the : operator just adds the interactions.

```
y ~ patient+dose
y ~ patient*dose == patient+dose+patient:dose
y ~ patient+dose+time+patient:dose
```