Basic Statistical Analysis of Array Data

SPH 247

Statistical Analysis of Laboratory Data

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.

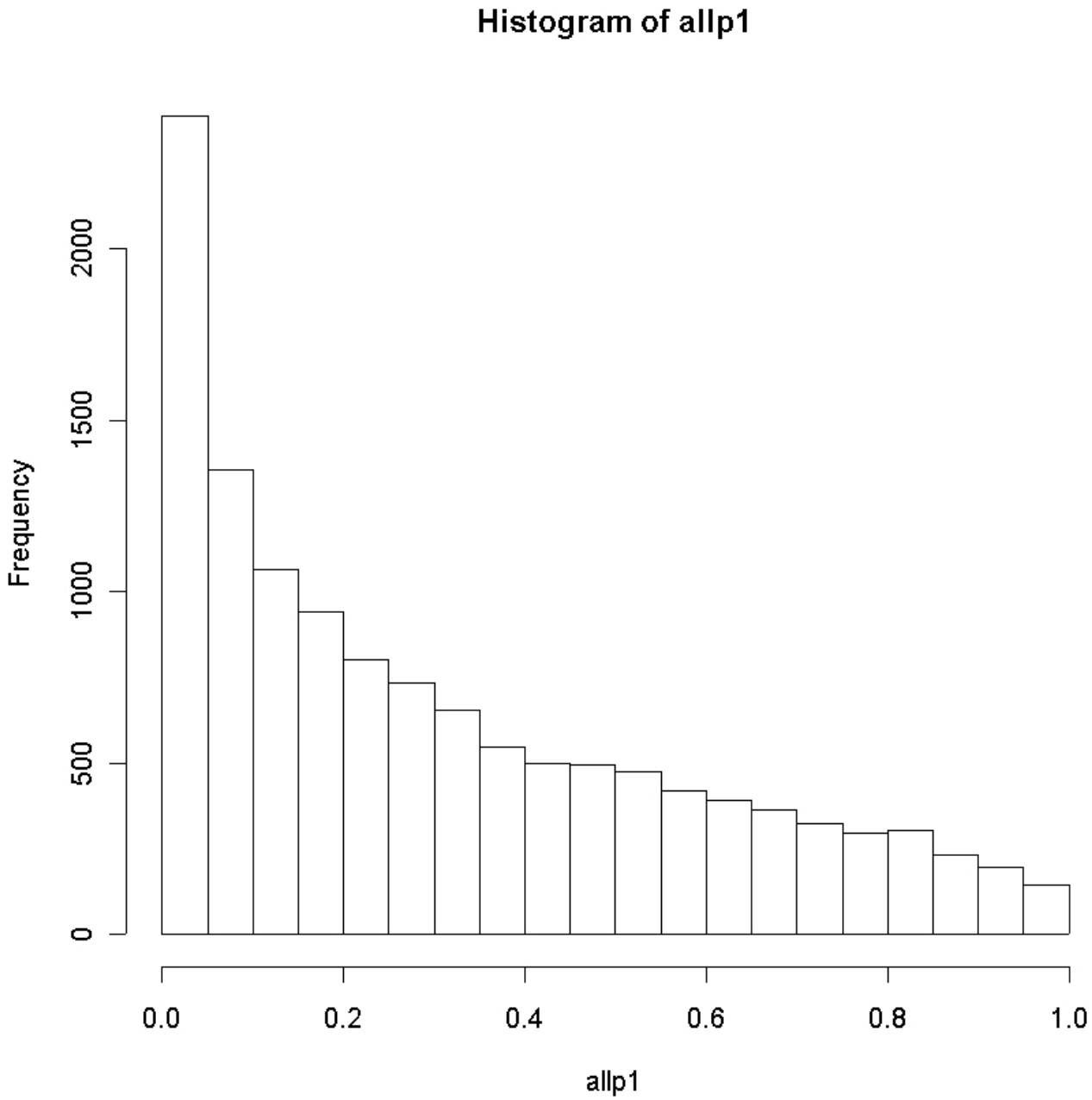
- In a gene expression array, and in other assays, the variance rises generally with the mean.
- For high level data, the log will stabilize the variance.
- For low level data, this causes problems
- Good transformations include the generalized log and the started log.
- Often, for Affymetrix data, the `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("G0","G0","G1","G1","G2","G2","G3",
                       "G3","G4","G4","G5","G5"))
> group
 [1] G0 G0 G1 G1 G2 G2 G3 G3 G4 G4 G5 G5
Levels: G0 G1 G2 G3 G4 G5
> 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){
# Conducts an ANOVA on one row of a matrix
  tmp <- anova(lm(y ~ group))$P[1]
  return(tmp)
}

allp <- function(array){
#Conducts ANOVAs on all rows of a matrix and gets p-values
  tmp2 <- apply(array,1,getp)
  return(tmp2)
}
-----------------------------------------------

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

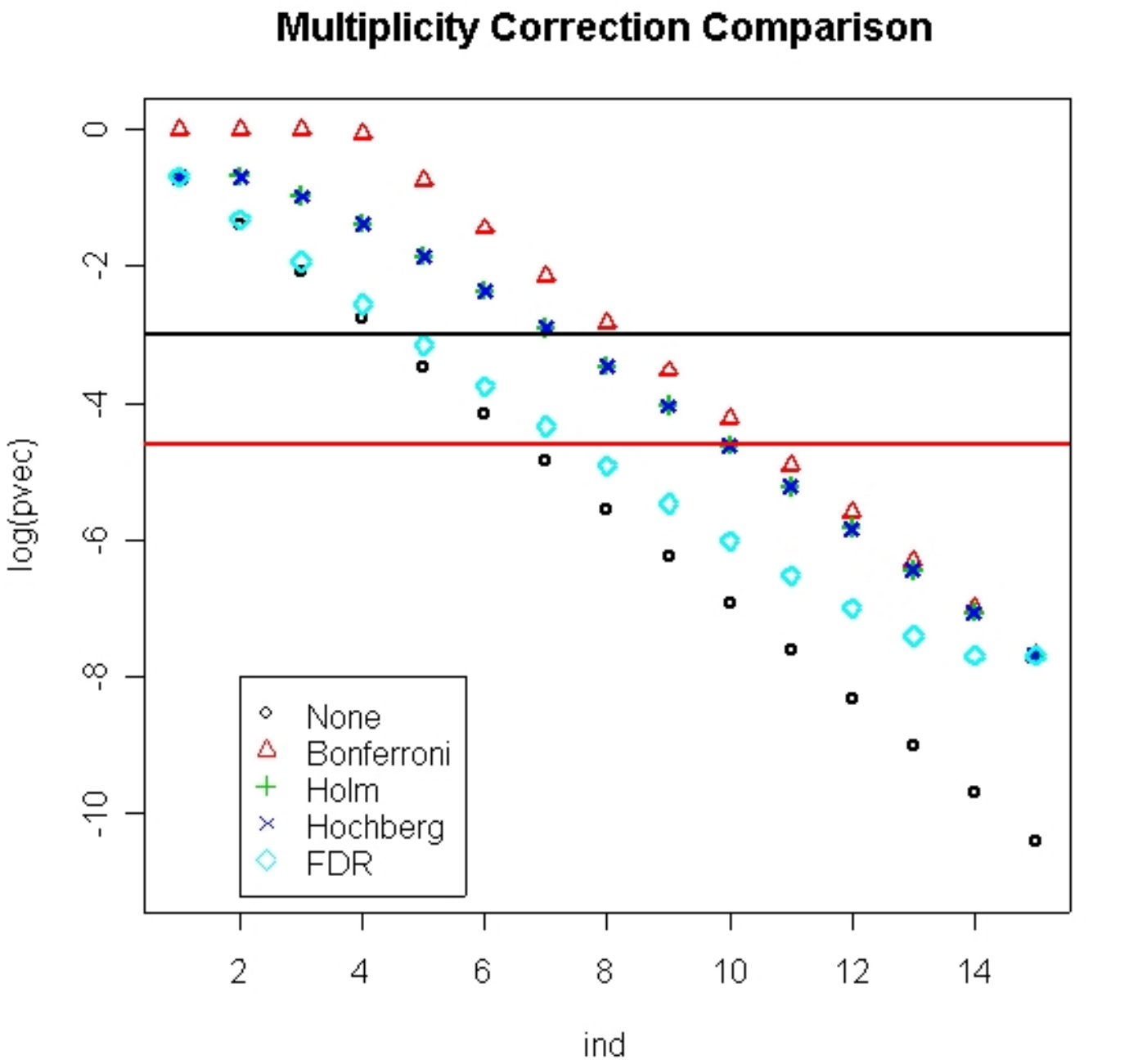

Histogram of allp1
Frequency
0
500
1000
1500
2000
0.0
0.2
0.4
0.6
0.8
1.0
allp1

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

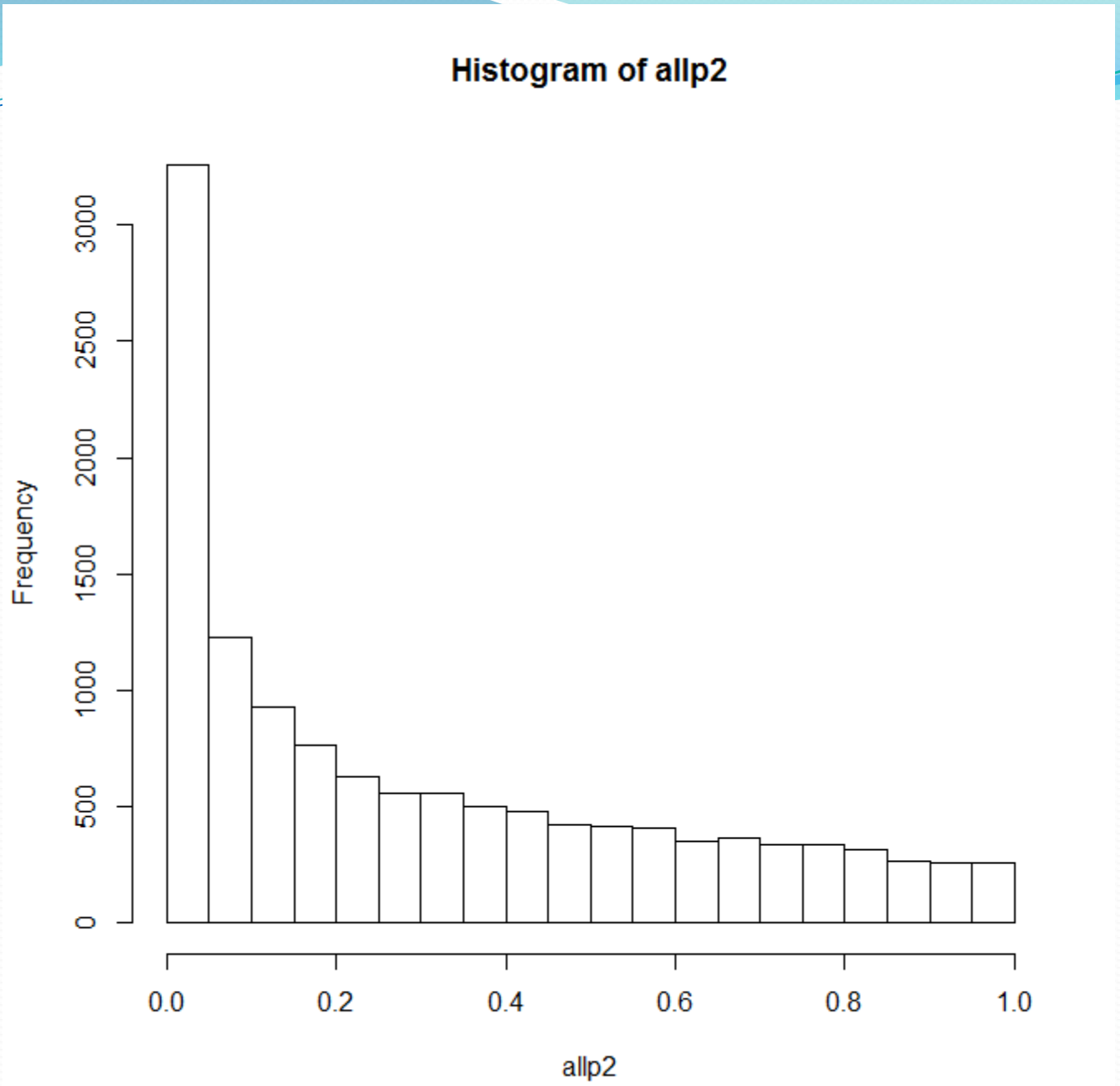
- The Bonferroni correction aims to detect no significant genes at all if there are truly none, and guarantees that the chance that any will be detected is less than .05 under these conditions
- Generally, this is too conservative
- Less conservative versions include methods due to Holm, Hochberg, and Benjamini and Hochberg (FDR)
- The following graph shows the result of applying a multiplicity correction to a vector (0.1, 0.01, … , 1e-15).

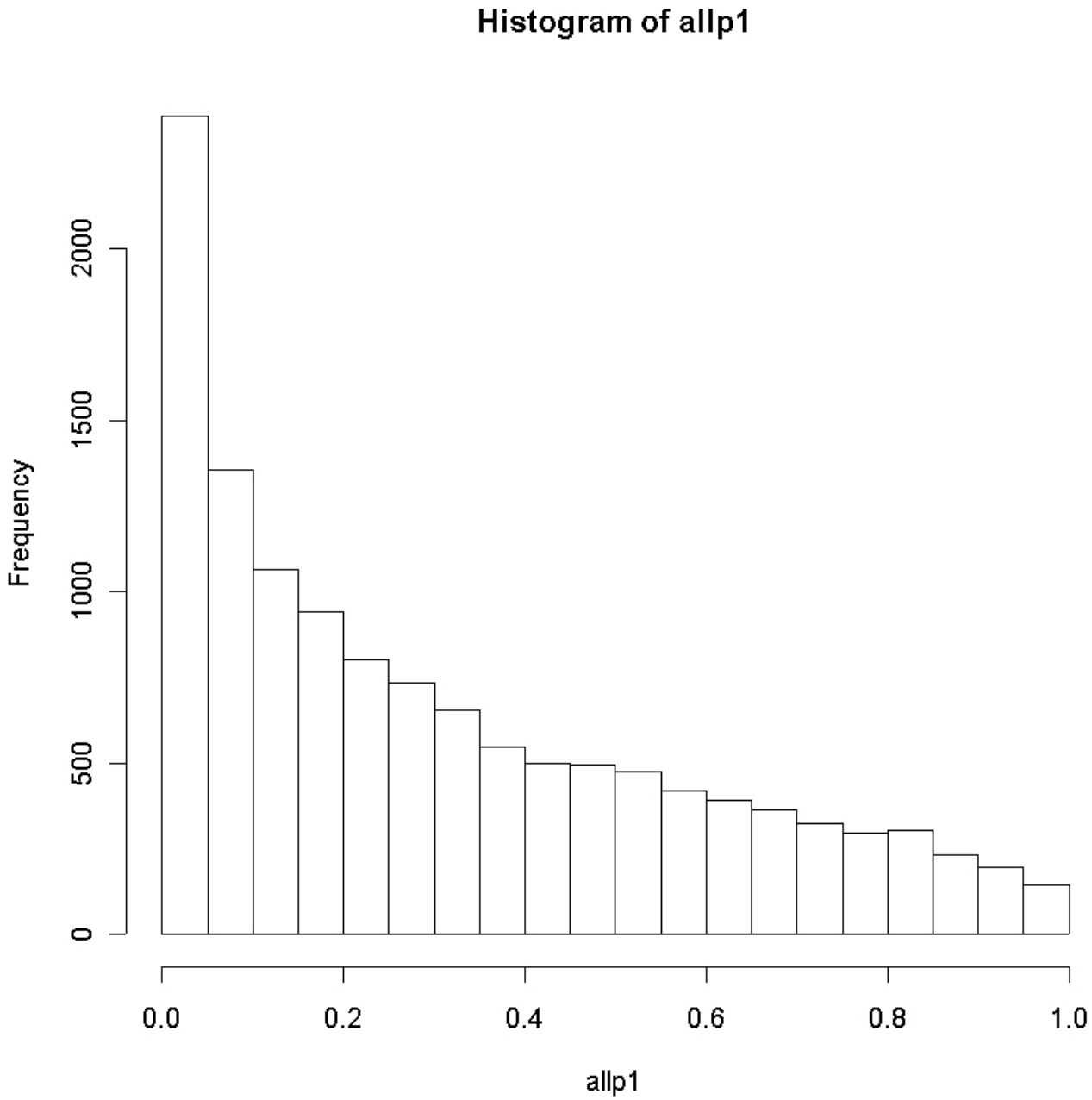

Multiplicity Correction Comparison
log(pvec)
0
-2
-4
-6
-8
-10
2
4
6
8
10
12
14
ind
None
Bonferroni
Holm
Hochberg
FDR

```
> allp1.adj <- p.adjust(allp1,"fdr") #built-in function
> sum(allp1.adj<.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"
```

```
> require(affy)
> require(limma)
> rrdata <- ReadAffy()
> eset <- rma(rrdata)
> group <- as.factor(c("G0","G0","G1","G1","G2","G2",
                       "G3","G3","G4","G4","G5","G5"))

#The code below is enough for a two-group test
#but only can test coefficients

design <- model.matrix(~group)
fits <- lmFit(eset, design)
fits.eb <- eBayes(fits)
allp2 <- fits.eb$p.value[2,]  # Tests group 1 vs. group 0
allp2.adj <- p.adjust(allp2,"fdr")
```

```
> design <- model.matrix(~0 + group) #fit with no intercept
> colnames(design) <- levels(group)
> fits2 <- lmFit(eset, design)

# The contrasts statement sets up an overall F-test
# If all the contrasts are true, then all the means are equal
> contrast.matrix <- makeContrasts(G0-G1,G0-G2,G0-G3,
  G0-G4,G0-G5, levels = design)

> fits2c <- contrasts.fit(fits2, contrast.matrix)
> fits2.eb <- eBayes(fits2c)
> allp2 <- fits2.eb$F.p.value
> allp2.adj <- p.adjust(allp2,"fdr")
```


Histogram of allp2
Frequency
0
500
1000
1500
2000
2500
3000
0.0
0.2
0.4
0.6
0.8
1.0
allp2


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

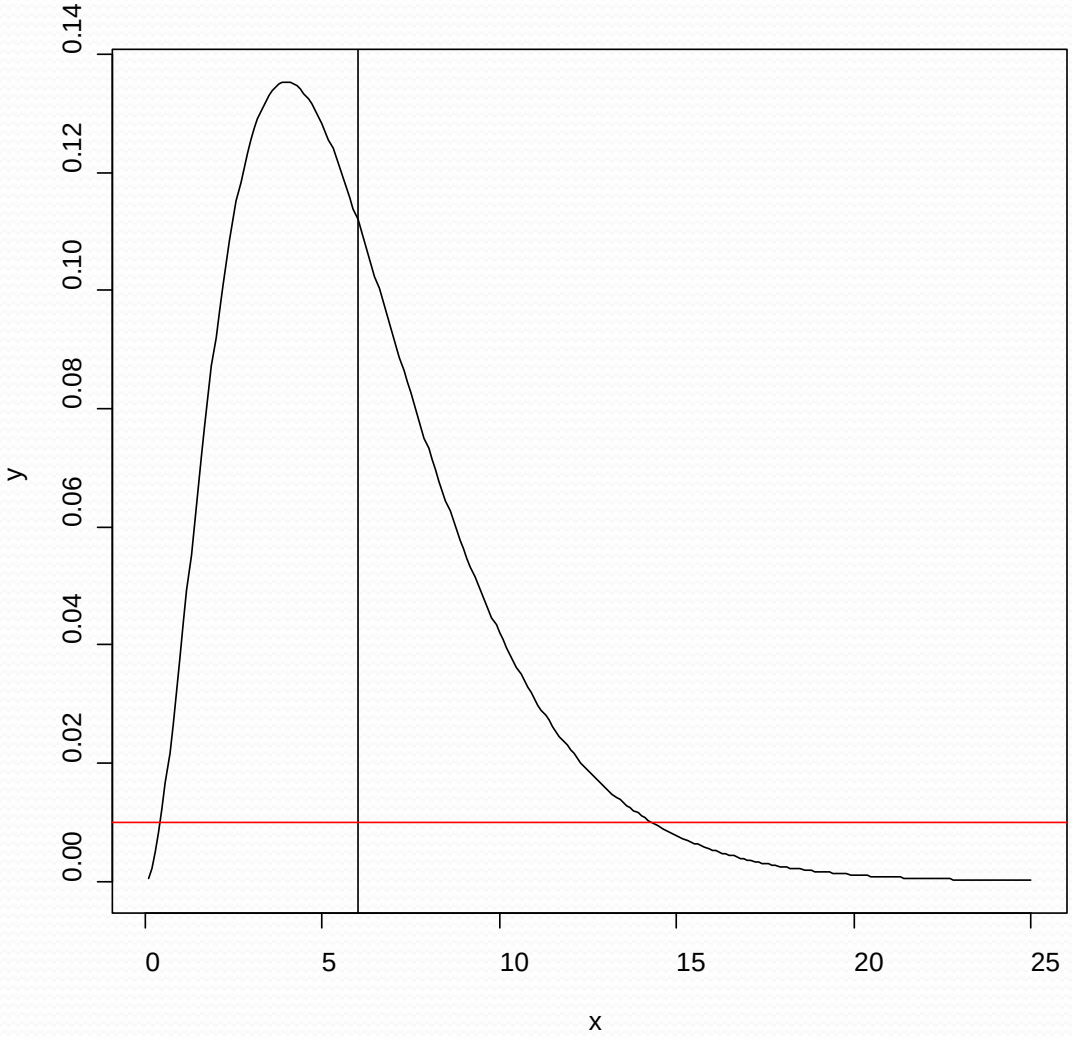
limma

- limma 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 except that it computes 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



Using Limma

- Data need to be normalized and transformed before using limma.
- The log transform may be ok, but check how low the values get and maybe add something to all the values first.
- The rma method in the Affy package does this both of these as part of the process

Using Limma

- You need to specify a linear model. If the predictors are numeric or have only two categories then you can use model.matrix(~x1+x2+…)
- Then fit all the linear models with lmFit and then use eBayes to improve the denominators.
- If there are factors with more then two levels, then you need to fit the model without an intercept (~0+…) and make a set of contrasts that encompases the entire effect of the factor. With 6 levels, this is 6*5/2 = 15/

Exercises

- For the sample affy data, fit the oneway ANOVA model to the RMA processed data using limma. Adjust the p-values for FDR. Try googling some of the affy feature names for the significant genes.