Example Analysis of an Affymetrix Dataset Using AFFY and LIMMA

4/4/2011

Example Dataset

- Available at Gene Expression Omnibus (GEO) Website as GSE 6297
- Somel M, Creely H, Franz H, Mueller U et al. (2008). Human and chimpanzee gene expression differences replicated in mice fed different diets. *PLoS One* 30;3(1):e1504. PMID: 18231591

Experiment Description from GEO

We fed four groups of six genetically identical 8-week-old female NMR1 mice one of four diets ad libidum:

(1) a diet consisting of vegetables, fruit and yogurt identical to the diet fed to chimpanzees in our ape facility ('Chimpanzee');

(2) a diet consisting exclusively of McDonalds fast food ('FastFood');

(3) a diet consisting of cooked food eaten by our staff in the Institute cafeteria ('HumanCafe');

(4) the mouse pellet diet on which they were raised ('Pellet').

Experiment Description from GEO (continued)

- At the end of a 2-week period, mice were euthanized by cervical dislocation and both liver and brain (right cerebral hemisphere) tissue were dissected.
- RNA was extracted from the 24 liver and brain samples as per established lab protocols and processed in two batches (containing equal numbers of individuals from all groups).

Example Analysis of Liver Samples

Batch 1	1	2	3	4
	1	2	3	4
	1	2	3	4
Batch 2	1	2	3	4
	1	2	3	4
	1	2	3	4


```
#Load affy package.
library(affy)
#Examine documentation.
vignette("affy")
#Set the path to a directory containing the CEL files.
setwd("C:\\z\\GEOdata")
#Read the CEL files.
D=ReadAffy()
```

```
D

AffyBatch object
size of arrays=1002x1002 features (15 kb)
cdf=Mouse430_2 (45101 affyids)
number of samples=24
number of genes=45101
annotation=mouse4302
notes=
Warning message:
package 'mouse4302cdf' was built under R version 2.12.1

attributes(D)
$.__classVersion__
        R   Biobase      eSet AffyBatch
 "2.12.0"  "2.10.0"   "1.3.0"   "1.2.0"

$cdfName
[1] "Mouse430_2"
```

7

```
$nrow
Rows
1002

$ncol
Cols
1002

$assayData
<environment: 049207f0>

$phenoData
An object of class "AnnotatedDataFrame"
  sampleNames: GSM144707.CEL GSM144708.CEL ...
               GSM144730.CEL (24 total)
  varLabels: sample
  varMetadata: labelDescription

$featureData
An object of class "AnnotatedDataFrame": none
```

8

```
$featureData
An object of class "AnnotatedDataFrame": none

$experimentData
Experiment data
  Experimenter name:
  Laboratory:
  Contact information:
  Title:
  URL:
  PMIDs:
  No abstract available.
  Information is available on: preprocessing
  notes:
   :


$annotation
[1] "mouse4302"
```

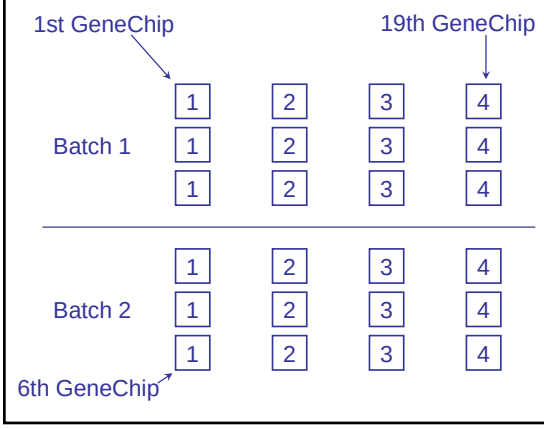
9

```
$protocolData
An object of class "AnnotatedDataFrame"
  sampleNames: GSM144707.CEL GSM144708.CEL ...
               GSM144730.CEL (24 total)
  varLabels: ScanDate
  varMetadata: labelDescription

$class
[1] "AffyBatch"
attr(,"package")
[1] "affy"
```

10

```
sampleNames(D)

 [1] "GSM144707.CEL" "GSM144708.CEL" "GSM144709.CEL" "GSM144710.CEL"
 [5] "GSM144711.CEL" "GSM144712.CEL" "GSM144713.CEL" "GSM144714.CEL"
 [9] "GSM144715.CEL" "GSM144716.CEL" "GSM144717.CEL" "GSM144718.CEL"
[13] "GSM144719.CEL" "GSM144720.CEL" "GSM144721.CEL" "GSM144722.CEL"
[17] "GSM144723.CEL" "GSM144724.CEL" "GSM144725.CEL" "GSM144726.CEL"
[21] "GSM144727.CEL" "GSM144728.CEL" "GSM144729.CEL" "GSM144730.CEL"

#Sometimes phenoData(D) will contain information about
#how to match the CEL files to treatment information.
#In this case, that information must be taken from
#the GEO database.

diet=rep(1:4,each=6)
batch=rep(rep(1:2,each=3),4)
rbind(diet,batch)
      [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10] [,11] [,12] [,13]
diet     1    1    1    1    1    1    2    2    2     2     2     2     3
batch    1    1    1    2    2    2    1    1    1     2     2     2     1
      [,14] [,15] [,16] [,17] [,18] [,19] [,20] [,21] [,22] [,23] [,24]
diet      3     3     3     3     3     4     4     4     4     4     4
batch     1     1     2     2     2     1     1     1     2     2     2
```

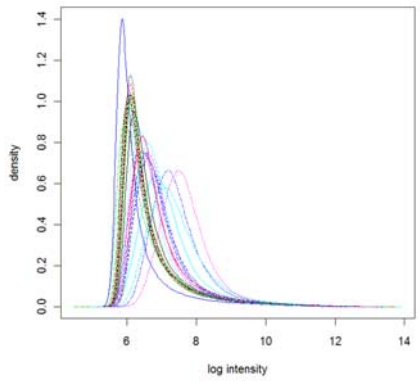
11



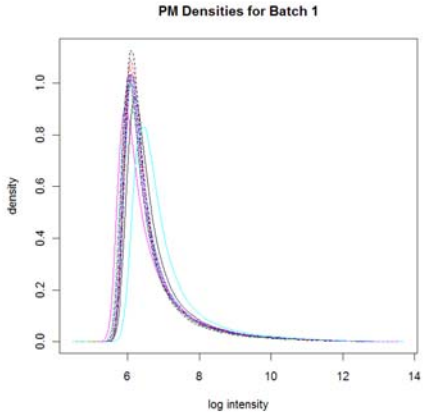
12

```
#We can examine PM densities for all GeneChips.

hist(D)

#We can examine PM densities for subsets of GeneChips.

hist(D[,batch==1],main="PM Densities for Batch 1")

hist(D[,batch==2], main="PM Densities for Batch 2")

#Make side-by-side boxplots of log2 PM
#color-coded by batch (red=1,blue=2)

boxplot(D,axes=F,xlab="GeneChip",ylab="log2 PM",
        col=2*batch)
axis(1,labels=
     paste(diet,batch,substr(sampleNames(D),8,9),sep=""),
     at=1:24,las=3)
axis(2)
box()
```

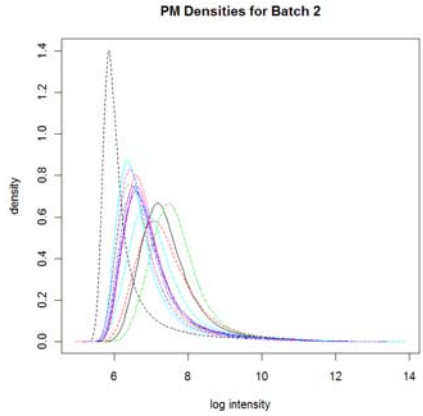
13



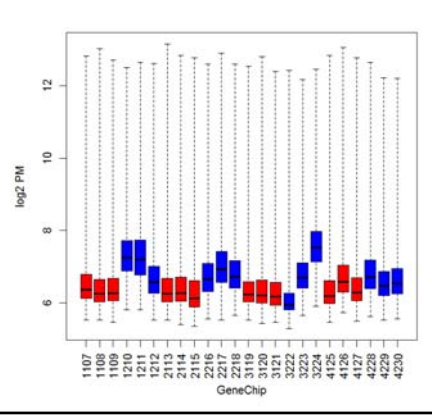
14



15

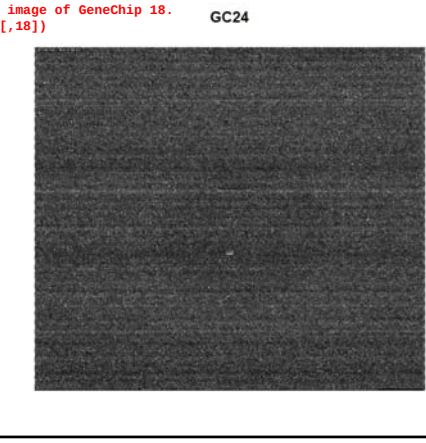


16



17

```
#See an image of GeneChip 18.
image(D[,18])
```



18

```
#Store probeset IDs and examine first 10.

gn=geneNames(D)
gn[1:10]
[1] "1415670_at"   "1415671_at"   "1415672_at"   "1415673_at"   "1415674_a_at"
[6] "1415675_at"   "1415676_a_at" "1415677_at"   "1415678_at"   "1415679_at"

#Shorten sample names.

sampleNames(D)=
           paste("GC",substr(sampleNames(D),8,9),sep="")

sampleNames(D)
[1] "GC07" "GC08" "GC09" "GC10" "GC11" "GC12"
[7] "GC13" "GC14" "GC15" "GC16" "GC17" "GC18"
[13] "GC19" "GC20" "GC21" "GC22" "GC23" "GC24"
[19] "GC25" "GC26" "GC27" "GC28" "GC29" "GC30"
```

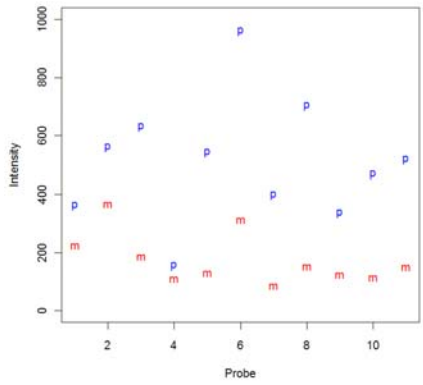
19

```
#Examine PM data for 10th probeset and a subset of chips.
pm(D, gn[10])[,11:20]

             GC17 GC18 GC19 GC20 GC21 GC22 GC23 GC24 GC25 GC26
1415679_at1   406  385  327  384  361  333  347  455  406  418
1415679_at2   717  619  424  593  495  405  508  711  585  634
1415679_at3   515  564  583  675  573  416  447  523  701  687
1415679_at4   201  156  145  170  146  102  153  222  150  178
1415679_at5   475  447  315  468  474  248  406  525  478  509
1415679_at6   798  808  766  854 1017  477  721  777 1060  947
1415679_at7   393  446  296  353  382  252  358  418  447  404
1415679_at8   586  579  545  605  582  505  552  634  699  665
1415679_at9   237  206  216  258  280  144  198  282  295  328
1415679_at10  384  391  329  533  392  337  378  435  470  483
1415679_at11  551  551  432  546  533  407  489  551  593  591
```

20

```
#Examine PM and MM data for first GeneChip.
cbind(pm(D, gn[10])[,1],mm(D, gn[10])[,1])

             [,1] [,2]
1415679_at1   362  224
1415679_at2   561  366
1415679_at3   630  184
1415679_at4   155  108
1415679_at5   542  128
1415679_at6   959  311
1415679_at7   397   84
1415679_at8   702  151
1415679_at9   336  123
1415679_at10  467  112
1415679_at11  518  148

plot(rep(1:11,2),c(pm(D, gn[10])[,1],mm(D, gn[10])[,1]),
   pch=rep(c("p","m"),each=11),col=(rep(c(4,2),each=11)),
   ylim=c(0,1000),xlab="Probe",ylab="Intensity")
```

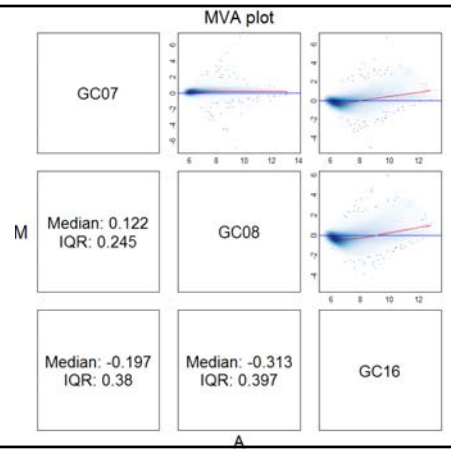
21



22

```
#Find the proportion of MM that exceed PM.

mean(pm(D)<mm(D))

[1] 0.3181409


#Plot PM difference (M) vs. PM average (A)
#for some GeneChip pairs.

library(geneplotter)
MAplot(D[,c(1,2,10)], pairs = TRUE,
       plot.method = "smoothScatter")
```

23



24

```
#Perform RMA normalization.

r=rma(D)
Background correcting
Normalizing
Calculating Expression

r
ExpressionSet (storageMode: lockedEnvironment)
assayData: 45101 features, 24 samples
  element names: exprs
protocolData
  sampleNames: GC07 GC08 ... GC30 (24 total)
  varLabels: ScanDate
  varMetadata: labelDescription
phenoData
  sampleNames: GC07 GC08 ... GC30 (24 total)
  varLabels: sample
  varMetadata: labelDescription
featureData: none
experimentData: use 'experimentData(object)'
Annotation: mouse4302
```



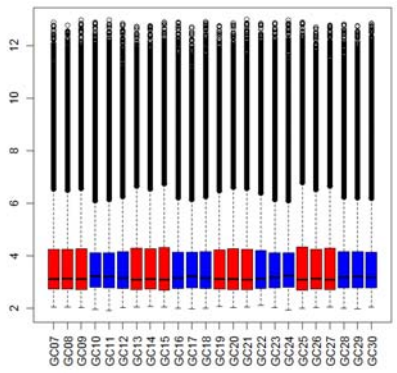
```
#Create a data matrix with a row for each probeset
#and a column for each GeneChip.

d=exprs(r)

dim(d)
[1] 45101    24
```



```
head(d)
                GC07     GC08     GC09     GC10     GC11     GC12     GC13
1415670_at   5.814663 6.181183 6.361403 5.027124 4.540326 5.194070 5.883083
1415671_at   8.143719 8.285620 8.484586 7.642196 7.544798 7.883108 8.326789
1415672_at   7.859956 7.729129 7.674707 7.148700 7.333395 7.753031 8.118746
1415673_at   3.611563 3.691870 3.670835 3.868315 3.504496 3.817986 3.984547
1415674_a_at 5.784022 5.854410 6.051814 5.103153 5.171708 5.623301 6.273496
1415675_at   5.423964 5.474299 5.497647 4.691470 5.154246 4.791819 5.469426
                GC14     GC15     GC16     GC17     GC18     GC19     GC20
1415670_at   6.010483 5.858698 5.321736 4.549718 5.268545 5.606858 5.921809
1415671_at   8.114104 8.371491 8.204701 7.427503 8.096952 8.185465 8.251371
1415672_at   8.094428 8.537742 8.095201 7.362150 8.124529 8.138183 8.152899
1415673_at   3.668004 3.694028 3.714788 3.605294 3.677794 3.564698 3.660264
1415674_a_at 6.192669 6.134427 5.665400 5.417380 5.692752 6.041425 6.110461
1415675_at   5.482453 5.300695 4.932934 5.124335 5.095452 5.268749 5.542943
                GC21     GC22     GC23     GC24     GC25     GC26     GC27
1415670_at   6.179215 5.277738 5.241534 4.508605 6.597141 6.149816 6.447506
1415671_at   8.311479 8.103617 7.905654 7.423459 8.416838 8.373552 8.441539
1415672_at   8.332273 7.752824 8.145597 7.254621 8.446753 8.079930 8.212414
1415673_at   4.104580 3.579261 3.610670 3.537173 4.188005 3.760504 4.054429
1415674_a_at 5.834398 5.885129 5.692189 5.176930 6.599257 6.315471 6.249057
1415675_at   5.196933 5.540621 5.066987 4.901728 5.294258 5.387153 5.547708
                GC28     GC29     GC30
1415670_at   5.836999 5.942128 6.129787
1415671_at   8.067470 8.059384 8.197599
1415672_at   7.814227 7.858566 8.216948
1415673_at   3.761913 3.746422 3.912751
1415674_a_at 5.773049 6.072727 6.282062
1415675_at   5.338097 5.316445 5.126147
```



```
boxplot(as.data.frame(d),las=3,col=2*batch)
```




```
#To keep the example simple, we will go through
#an analysis for the Batch 1 data.

d1=d[,batch==1]
diet=factor(diet[batch==1])
diet

[1] 1 1 1 2 2 2 3 3 3 4 4 4
Levels: 1 2 3 4
```



```
library(limma)

#R uses "set first to zero" constraint by default.

design=model.matrix(~diet)
design
   (Intercept) diet2 diet3 diet4
1            1     0     0     0
2            1     0     0     0
3            1     0     0     0
4            1     1     0     0
5            1     1     0     0
6            1     1     0     0
7            1     0     1     0
8            1     0     1     0
9            1     0     1     0
10           1     0     0     1
11           1     0     0     1
12           1     0     0     1
```

```
#In this case it may be more convenient to
#use a treatment mean parameterization.

design=model.matrix(~0+diet)
design

   diet1 diet2 diet3 diet4
1      1     0     0     0
2      1     0     0     0
3      1     0     0     0
4      0     1     0     0
5      0     1     0     0
6      0     1     0     0
7      0     0     1     0
8      0     0     1     0
9      0     0     1     0
10     0     0     0     1
11     0     0     0     1
12     0     0     0     1
```

31

```
#Name the columns of the design matrix.
#This is equivalent to naming the parameters
#in the regression vector.

colnames(design)=c("u1","u2","u3","u4")

#Fit linear model for each gene.

fit=lmFit(d1,design)

#Set up contrasts of interest.
#In this case, we consider all pairwise
#comparisons between treatment means.

contr.mat=makeContrasts(u1-u2,u1-u3,u1-u4,
                        u2-u3,u2-u4,u3-u4,
                        levels=design)
```

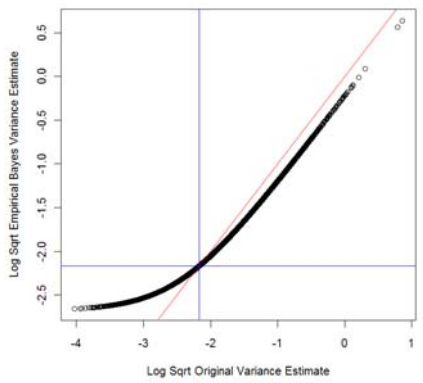
32

```
#Estimate contrasts.

fit2=contrasts.fit(fit,contr.mat)

#Conduct analysis based on Smyth's
#empirical Bayes approach.

fit3=eBayes(fit2)

#Estimates of parameters in the prior on
#the gene-specific error variances.

fit3$df.prior
[1] 4.477159

fit3$s2.prior
[1] 0.01307461
```

33

```
#Compare variance estimates before and after shrinkage.

plot(log(fit3$sigma),log(sqrt(fit3$s2.post)),
     xlab="Log Sqrt Original Variance Estimate",
     ylab="Log Sqrt Empirical Bayes Variance Estimate")
lines(c(-99,99),c(-99,99),col=2)
lsrs20=log(sqrt(fit3$s2.prior))
abline(h=lsrs20,col=4)
abline(v=lsrs20,col=4)
```

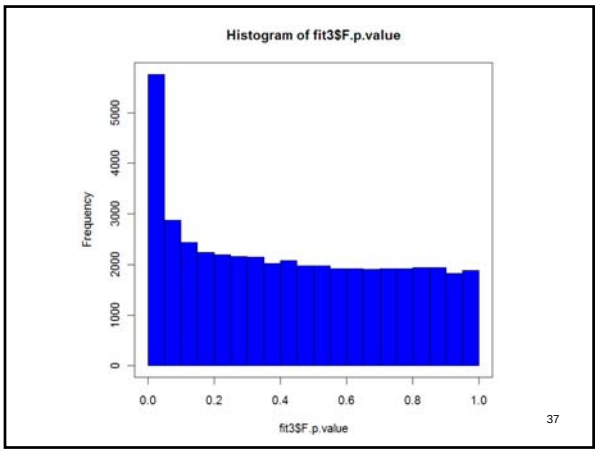
34



35

```
#Examine p-values for F-test whose null says
#that all contrasts in contr.mat are
#simultaneously zero.  In this case, that is
#equivalent to H_0:u1=u2=u3=u4.

hist(fit3$F.p.value,col=4)
box()
```

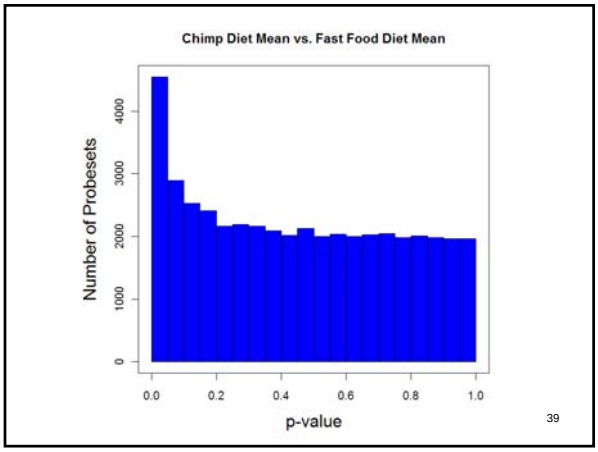
36



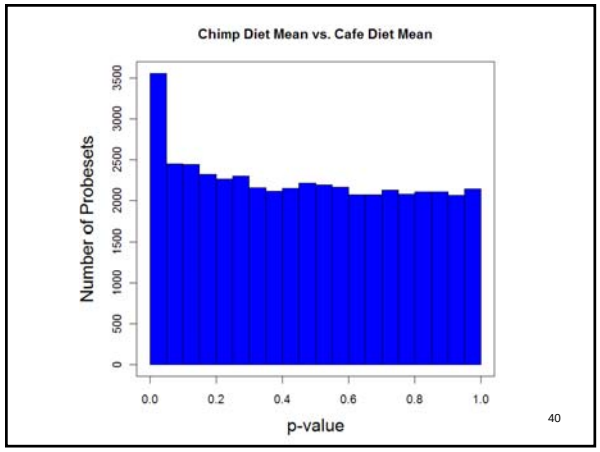
37

```
#Examine p-values for pairwise comparisons.

p=fit3$p.value
dim(p)
[1] 45101 6

hist(p[,1],col=4,cex.lab=1.5,
          xlab="p-value",
          ylab="Number of Probesets",
          main="Chimp Diet Mean vs. Fast Food Diet Mean")
box()

.
.
.

hist(p[,6],col=4,cex.lab=1.5,
          xlab="p-value",
          ylab="Number of Probesets",
          main="Cafe Diet Mean vs. Pellet Diet Mean")
box()
```

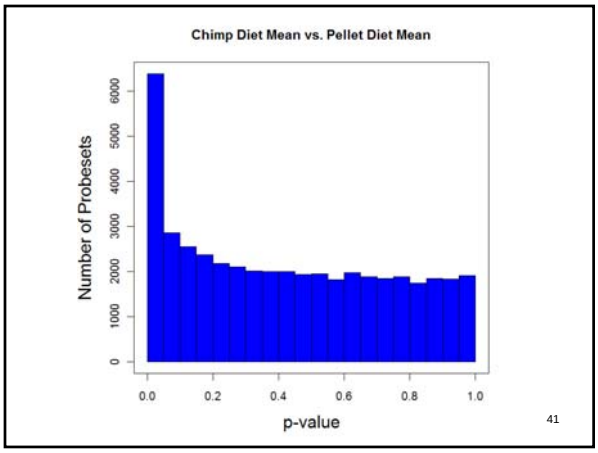
38



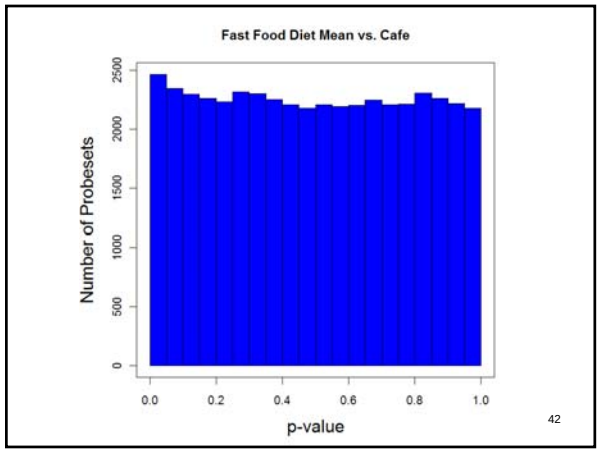
39



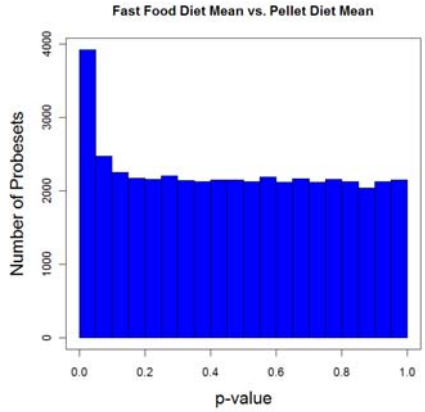
40



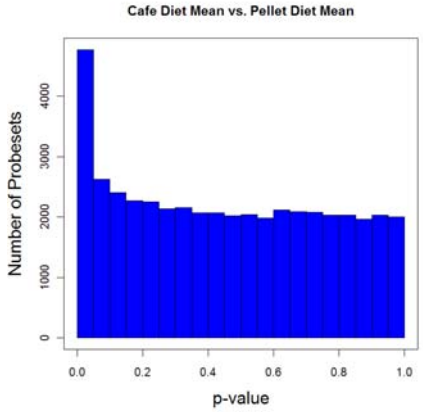
41



42



43



44

```
#Load some multiple testing functions.

source("http://www.public.iastate.edu/~
        dnett/microarray/multtest.txt")

#Estimate proportion of true nulls
#for each pairwise comparison.

apply(p,2,estimate.m0)/nrow(d1)
  u1 - u2   u1 - u3   u1 - u4   u2 - u3   u2 - u4   u3 - u4
0.8682734 0.9418860 0.8249632 0.9839767 0.9472074 0.9017337

#Separately for each column of p,
#convert p-values to q-values.

qvalues=apply(p,2,jabes.q)
```

45

```
#For each pairwise comparison, find
#number of genes declared significant
#at an estimated FDR level of 0.05.

apply(qvalues<=0.05,2,sum)

u1 - u2  u1 - u3  u1 - u4  u2 - u3  u2 - u4  u3 - u4
    113       44     1658        1      371      612
```

46

```
#Find probeset IDs of some significant genes.

row.names(d1)[qvalues[,2]<=0.05]

 [1] "1416029_at"   "1416184_s_at" "1417219_s_at"
 [4] "1417761_at"   "1417883_at"   "1418474_at"
 [7] "1418654_at"   "1418833_at"   "1419146_a_at"
[10] "1420731_a_at" "1421258_a_at" "1421259_at"
[13] "1422257_s_at" "1423078_a_at" "1425252_a_at"
[16] "1425300_at"   "1425303_at"   "1425645_s_at"
[19] "1426159_x_at" "1427747_a_at" "1427883_a_at"
[22] "1428004_at"   "1428093_at"   "1429272_a_at"
[25] "1429735_at"   "1435455_at"   "1436504_x_at"
[28] "1436902_x_at" "1437185_s_at" "1438322_x_at"
[31] "1438377_x_at" "1438711_at"   "1439566_at"
[34] "1442026_at"   "1448619_at"   "1448898_at"
[37] "1449375_at"   "1451122_at"   "1451787_at"
[40] "1453080_at"   "1453345_at"   "1454161_s_at"
[43] "1455065_x_at" "1455439_a_at"
```

47