Statistics 416
HW2
Solutions

1. *(20 points)*

Create and display a vector of the consecutive integers 1 though 12.

Create a vector of the numbers 9, 5, and 3.

Create the vector 1,2,3,4,5,6,9,5,3 and assign it to x.
Display x.

Multiply each entry in x by 2 and subtract 3. Assign the result to y.
Display y.

Display the second element of y.

Display all the elements in y except the 2^{nd}.

Display the first 3 elements of y.

Display all elements of y except the first 3.

Sort the elements of the vector y from smallest to largest.

Create a matrix containing the elements of y. Organize the elements in 3 rows. Fill the matrix with the elements going down the columns. Assign the result to m.
Display m.

Display the element of m in the 1^{st} row and 3^{rd} column.

Display the first row of m.

Display the first column of m.

Display the submatrix consisting of the first 2 rows and the 1^{st} and 3^{rd} columns of m.

Compare each entry in the first column of m to 0. If the entry is greater than 0, report TRUE. Otherwise report FALSE. Organize the results in a vector with entries corresponding to the entries in the first column of m.

Report the rows of m for which the entries in the first column of m are greater than 0.

Add an additional row (14, 15, 21) as the new last row of the matrix m.
Display m.

Sum the entries in each row of m.

Find the mean of the entries in each column of m.

Plot the entries in the first column (along the horizontal axis) against the corresponding entries in the second column (along the vertical axis).

Same as above except use solid circles rather than open circles to represent each point.

Same as above except color the points blue.

Connect the points with red lines.

2. *(10 points)*

a) The quality does not look good. The background is high. There are many streaks.

b) There are 48 sectors on the slide in a 12 row by 4 column format. Thus, 48 pins were probably used to print the slide.

c)

i. Mitochondrial translational initiation factor 2
ii. 52
iii. 290
iv. Channel 2 has a higher intensity than channel 1, and it looks as if the red signal is higher than the green signal based on the reported R/G ratios. Thus, channel 1 is G, and channel 2 is R.
v. Red is higher because the R/G ratio is greater than 1.
vi. 1152
vii. 210

d) 87 (mean intensity), 78 (median background)

e) 53533 has very uneven background and is quite splotchy. 53981 looks quite nice when compared to the other two slides that we examined.

3. *(10 points)* Area to perimeter = (spot area)*4*pi / perimeter^2. The area of a half circle is 0.5*pi*r^2. The perimeter is pi*r+2*r. Thus the area-to-perimeter measure would be

$0.5*pi*r^2*4*pi / (pi*r+2*r)^2 = 2*pi^2 / (pi+2)^2$ which is approximately 0.747.

4. *(10 points)*

```
d=read.table("http://www.public.iastate.edu/~dnett/microarray/pixelData.txt")
d=as.matrix(d)
sort(d[2:7,2:7])
```

```
[1] 17145 19270 23802 24026 24087 24833 25706 26478 26776 26992 27335 27517
[13] 28004 28360 28634 29184 29316 29427 29453 30082 30683 30951 30999 31682
[25] 32037 32092 32175 32228 32474 33187 33989 34416 35218 35236 36411 37174
```

Now we must find the 75th percentile of the pixel intensities because that is what Affymetrix uses as the intensity for a probe cell. The 75th percentile is the same as the 0.75 quantile of the 36 pixel intensities. According to our class notes, the 0.75 quantile is any number that is greater than or equal to at least 75% of the data points and less than or equal to at least 25% of the data points. Any number between the 27th and 28th pixel intensities (including the 27th and 28th values themselves) ordered from smallest to largest will satisfy this definition. Thus we need to put the pixel intensities in order and select any number between the 27th and 28th values. This can be done in R as follows.

```
sort(d[2:7,2:7])[27:28]
[1] 32175 32228
```

From this command we can easily see that the 27th and 28th values are 32175 and 32228. Thus the 0.75 quantile is any number between 32175 and 32228. I do not know which number Affymetrix uses, but one strategy would be to use the average of these two numbers.

This gives 32201.5 as the 0.75 quantile. Any answer in the interval [32175, 32228] is acceptable for this homework problem.

We could alternatively use the quantile function in R to get a 0.75 quantile.

```
quantile(d[2:7,2:7],0.75)
```

This command gives 32188.25 as a 0.75 quantile. This number is 25% of the way between 32175 and 32228.

5. *(10 points)* I wrote an R function for computing the trimmed mean. That function is provided below.

```
trimmed.mean=
function (x,low,up)
{
  low=quantile(x,low)
  up=quantile(x,up)
  mean(x[x>low&x<up])
}
```

The & sign takes two vectors consisting of logicals (TRUE and FALSE) and creates a new vector that is TRUE whenever the corresponding elements of both vectors are TRUE and FALSE otherwise. Thus, c(T,T,F,F)&c(F,T,T,T) evaluates to F,T,F,F. The & is known as the logical "and" operator. The condition in the square brackets in the function above is x>low&x<up. This will be TRUE only if x>low AND x<up. Using the R function, it would be easy to obtain a trimmed mean as follows.

trimmed.mean(c(34, 25, 22, 31, 47, 98, 26, 18, 19, 24, 34, 25, 27, 33, 31),.2,.9)

29

Answers may vary because there are multiple ways to define the 0.2 and 0.9 quantiles. Our definition says that at least 20% of the data must be less than or equal to the 0.2 quantile and at least 80% needs to be greater than or equal to the 0.2 quantile. R says that the 0.2 quantile is 23.6. All the data sorted from smallest to largest is

18 19 22 24 25 25 26 27 31 31 33 34 34 47 98.

Note that 3 out of 15 (20%) of the observations are less than or equal to 23.6. Also note that 12 out of 15 (80%) of the observation are greater than or equal to 23.6. Thus 23.6 satisfies our definition of the 0.2 quantile.

According to our definition, 47 must be the 0.9 quantile of the data set. 14/15=93.3333% of the observations are less than or equal to 47. Also 2/15=13.3333% of the observations are greater than or equal to 47. These percentages must be at least 90% and 10%, respectively. No other number satisfies our requirements for the 0.9 quantile. The trimmed mean is the mean of the numbers strictly between the 0.2 and 0.9 quantiles. These number are

24 25 25 26 27 31 31 33 34 34

and their mean is 29.

R gives 41.8 as the 0.9 quantile which doesn't satisfy our definition, but we will end up with the same trimmed mean if we use the 0.9 quantile according to R.

So acceptable answers are 29 or 29.55556=mean(c(25, 25, 26, 27, 31, 31, 33, 34, 34)) if 24 is taken as the 0.2 quantile.

6. I wrote R functions to do the relevant calculations. Many of you have little of no programming experience, so I did not expect you to write functions like these. However, if you study these functions you should be able to learn some basic programming skills.

(a) *(15 points)*

```
affy.tbw=function(x)
{
  m=median(x)
  mad=median(abs(x-m))
  t=(x-m)/(5*mad+0.0001)
  b=(1-t^2)^2
  b[abs(t)>=1]=0
  tbw=sum(b*x)/sum(b)
  tbw
}
```

```
ideal.mismatch=
function (pm,mm)
{
  im=mm
  index=(pm<mm)
  if(sum(index)>0){
    M=affy.tbw(log(pm/mm,2))
    if(M>.03){
      im[index]=pm[index]/2^M
    }
    else{
      im[index]=pm[index]/2^(0.03/(1+((0.03-M)/10)))
    }
  }
  im
}
```

```
> pm=c(2234,3945,4232,2645,1787,1100,2345)
> mm=c(5895,2238,1887,300,1800,154,156)
> ideal.mismatch(pm,mm)
[1]  810.0751 2238.0000 1887.0000  300.0000  647.9875  154.0000
156.0000
```

(b) *(5 points)*

```
probe.value=
function (pm,mm)
{
  im=ideal.mismatch(pm,mm)
  v=pmax(pm-im,2^(-20))
  log(v,2)
}
```

```
> probe.value(pm,mm)
[1] 10.475657 10.737247 11.195372 11.195372 10.153568  9.885696 11.096056
```

7.

(a) *(10 points)*

I expected most of you to use a series of command like the following.

```
d=read.table("http://www.public.iastate.edu/~dnett/microarray/pmval.txt")
d=as.matrix(d)
x=sweep(d,1,apply(d,1,median))
x=sweep(x,2,apply(x,2,median))
x=sweep(x,1,apply(x,1,median))
x=sweep(x,2,apply(x,2,median))
x=sweep(x,1,apply(x,1,median))
x=sweep(x,2,apply(x,2,median))
x=sweep(x,1,apply(x,1,median))
x=sweep(x,2,apply(x,2,median))
```

```
x=sweep(x,1,apply(x,1,median))
x=sweep(x,2,apply(x,2,median))
x=sweep(x,1,apply(x,1,median))
x=sweep(x,2,apply(x,2,median))
x=sweep(x,1,apply(x,1,median))
x=sweep(x,2,apply(x,2,median))
x=sweep(x,1,apply(x,1,median))
x=sweep(x,2,apply(x,2,median))
x=sweep(x,1,apply(x,1,median))
x=sweep(x,2,apply(x,2,median))
x=sweep(x,1,apply(x,1,median))
x=sweep(x,2,apply(x,2,median))
apply(d-x,1,mean)
 [1]  9.685233 11.603983 10.610233 10.687733 10.970233  9.570233 10.483983
 [8] 10.055233 10.647733 11.147694
```

It is also possible to use a built-in R function called medpolish.

mp=medpolish(d)
1 : 15.17
2 : 13.785
Final: 13.7025

fitted.values=d-mp$residuals
apply(fitted.values,1,mean)

```
       1        2        3        4        5        6        7
 9.68216 11.60466 10.62466 10.68716 10.96966  9.56966 10.48216
       8        9       10
10.05716 10.64716 11.13966
```

Answers will vary depending on many steps you used. For example *mp=medpolish(d,eps=0.000000000000000001,maxiter=100)* will cause R to iterate 45 times for this dataset. The answer will change as follows

fitted.values=d-mp$residuals
apply(fitted.values,1,mean)

```
        1         2         3         4         5         6         7
 9.685227 11.603977 10.610227 10.687727 10.970227  9.570227 10.483977
        8         9        10
10.055227 10.647727 11.147727
```

7. (b) *(10 points)*

```
y=apply(d-x,1,mean)
t.test(y[1:5],y[6:10],var.equal=T)

        Two Sample t-test

data:  y[1:5] and y[6:10]
t = 0.8063, df = 8, p-value = 0.4434
alternative hypothesis: true difference in means is not equal to 0
```

```
95 percent confidence interval:
 -0.6147502  1.2757658
sample estimates:
mean of x mean of y
 10.71148  10.38097
```

The output shows no convincing evidence of a treatment effect (t-stat=0.8063, df= 8, p-value=0.4434). The difference between the two observed means could have easily arisen by chance.