BIOS 7659 Homework 1

Charlie Carpenter

17 September, 2018

1

Case Study 3

We are interested in establishing patient clusters that express similar genes. With design 1 we have the issue of having two different types of comparisons to make: patients within an array and patients across arrays. This design would force patients to competitively hybridize on the array which would be counterproductive to establishing clusters of patients. Also, if something goes wrong with one array we would be losing two samples instead of two. Design two gives us the ability to use the reference as an estimate of background variability while keeping patient comparisons consistent (all are across arrays). Design 3 would also keep this consistency in comparisons while saving on costs spent on reference samples, but without giving us the benefit of the reference.

Case Study 4

Design 1 doesn't give us any replicates of time points, meaning if a GeneChip becomes damaged we lose all of our information about that time point. Also, obviously, we have no technical replicates of time points. Design 2 we have direct comparisons to time 0 for times 1 - 6, but across-array comparisons for all other time points. Design two would be better for a question along the lines of, "What genes stay consistent along the time course?" Design three, the loop design, is best for this question. The design has replicates of each time point, allows for direct and across-array comparisons of all time points, and would directly address the question of interest. This design also would give more statistical power to find genes expressing similar profiles throughout the time course. One draw back of the loop design is that is does require a more complicated analysis.

2

```
## Fold change of 2 on log_2 scale is effect size of 1
## sd estimate = 0.25 (MA section 3.8)

## using pwr.t.test for many different power levels

pwr.lvls <- c(0.8, 0.95)

## d = delta/sd
pwr.n <- sapply(pwr.lvls, function(x){
  pwr.t.test(d = 4, sig.level = 0.0001, power = x, type = "two.sample", alternative =
               "two.sided")$n

}
)

## Rounding up our sample sizes
ceiling(pwr.n)
```

```
## [1] 7 8
```

```
## True powers for these sample sizes
true.pwr <- sapply(c(7,8), function(x){
  pwr.t.test(d = 4, sig.level = 0.0001, n = x, type = "two.sample", alternative =
               "two.sided")$power
}
)
```

The standard deviation of kidney gene expression in inbred strains of mice on Affymetrix GeneChips is estimated to be 0.25. This has been a consistent finding in many past studies. The null hypothesis for each of the 20,000 probe sets is that there is no difference in true mean gene expression between the treated and untreated mice, with the alternative hypothesis being that there is a difference in the true mean gene expression. An alpha level of 0.0001 for the two-sided test on each of the 20,000 probe sets gives us an estimated 2 false discoveries. With 7 mice per group we can achieve approximately 90% power, and with 8

mice per group we can achieve over 95% power. These calculations are summarized in the table below.

N	True_Power	Cost
14	0.8928827	$14000
16	0.9711072	$16000

3

```
## Reading in data sets
array_data <- read.table("~/Documents/Classes_Fall_2018/BIOS 7659/HW1/arraydata.txt", row.names = 1, hea

sd_values <- read.table("~/Documents/Classes_Fall_2018/BIOS 7659/HW1/sdvalues.txt")

pvalues <- read.table("~/Documents/Classes_Fall_2018/BIOS 7659/HW1/pvalues.txt")
```

a)

```
##
(pwr.n.2 <- sapply(pwr.lvls, function(x){
  pwr.t.test(d = 1/.5, sig.level = 0.001, power = x, type = "two.sample",
             alternative= "two.sided")$n

}
)) %>%
  ceiling
```

```
## [1] 12 15
```

We would need 12 and 15 mice per group in order to achieve 80% and 95% power, respectively.

π_0 is the expected proportion of false null hypotheses. Here we are assuming that the null is true for all 20,000 t-tests we run, which is an unreal assumption. With the FDR approach we assume that some small percent of null hypotheses are false. So, if we to assume 5% of null hypotheses were false (common) π_0 would be 0.95.

b)

```
(FDR.n <- sapply(pwr.lvls, function(x){
  ssize::power.t.test.FDR(sd = .5, delta = 1, FDR.level = 0.05, pi0 = .95,
                          power = x, type = "two.sample",
                          alternative = "two.sided")$n
}
)) %>%
  ceiling
```

```
## [1] 11 14
```

Here we only need 11 and 14 mice per group in order to achieve 80% and 95% power, respectively.

c)

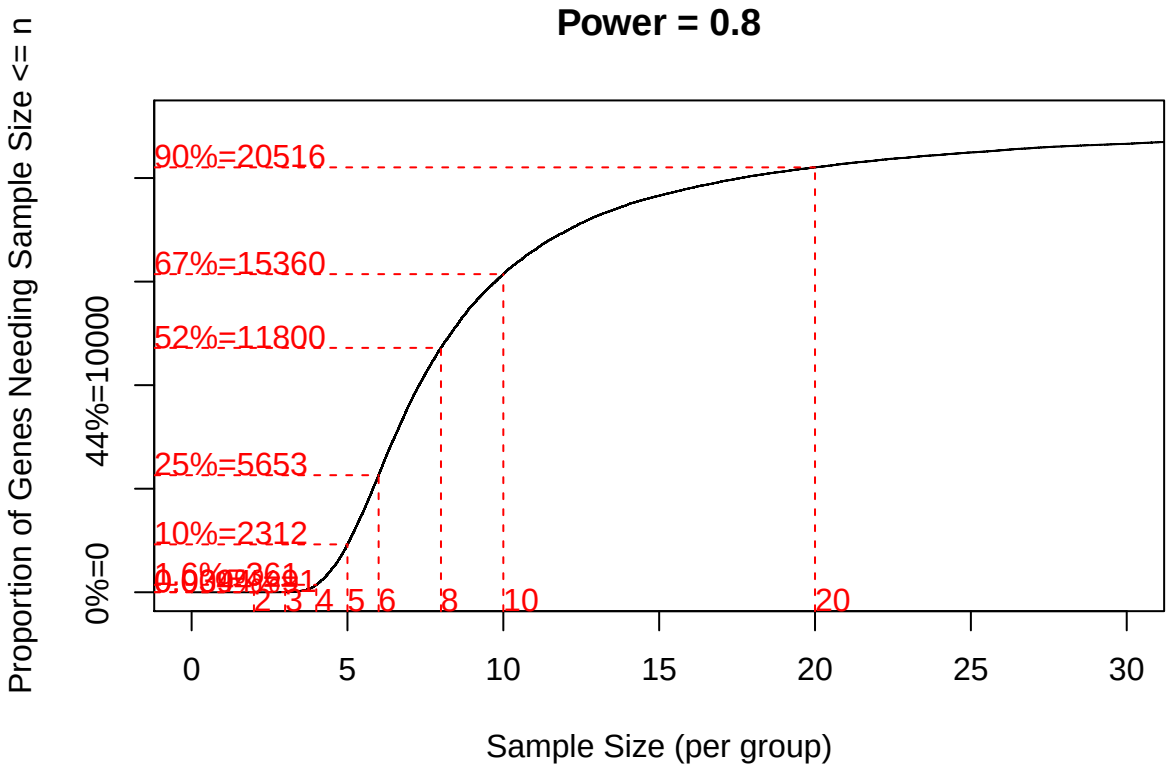
```
set.seed(5)
plot.pwr.8 <- ssize::ssize(sd = sd_values$V2, delta = 1,
                           sig.level = 0.001, power = 0.8)
```

```
## ..........................................................................................
## Warning in qt(sig.level/tside, nu, lower.tail = FALSE): NaNs produced
## ..........................................................................................
```

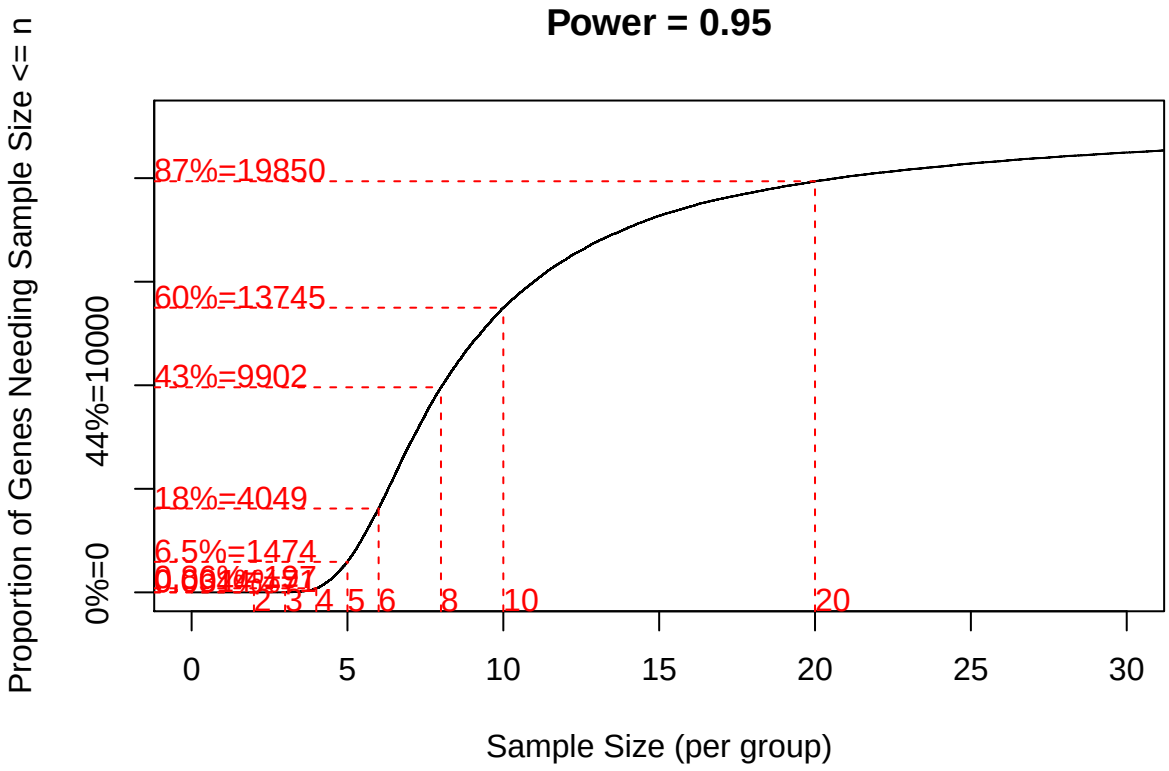
```
set.seed(5)
plot.pwr.95 <- ssize::ssize(sd = sd_values$V2, delta = 1,
                            sig.level = 0.001, power = 0.95)
```

```
## ..........................................................................................
## Warning in qt(sig.level/tside, nu, lower.tail = FALSE): NaNs produced
## ..........................................................................................
```

```
ssize::ssize.plot(plot.pwr.8, xlim = c(0,30), main = "Power = 0.8")
```




```
ssize::ssize.plot(plot.pwr.95, xlim = c(0,30), main = "Power = 0.95")
```



Based on this plot we would need 20 mice per group in order for 90% of our genes to have at least 80%

power, and we'd need 20 mice per group in order for 87% of our genes to have at least 95% power.

e)

```
set.seed(234)
data <- list(x=array_data,y=c(rep(1,4),rep(2,4)), geneid=row.names(array_data), genenames = row.names(a
samr.obj <-  samr(data, resp.type="Two class unpaired", nperms=100)
```

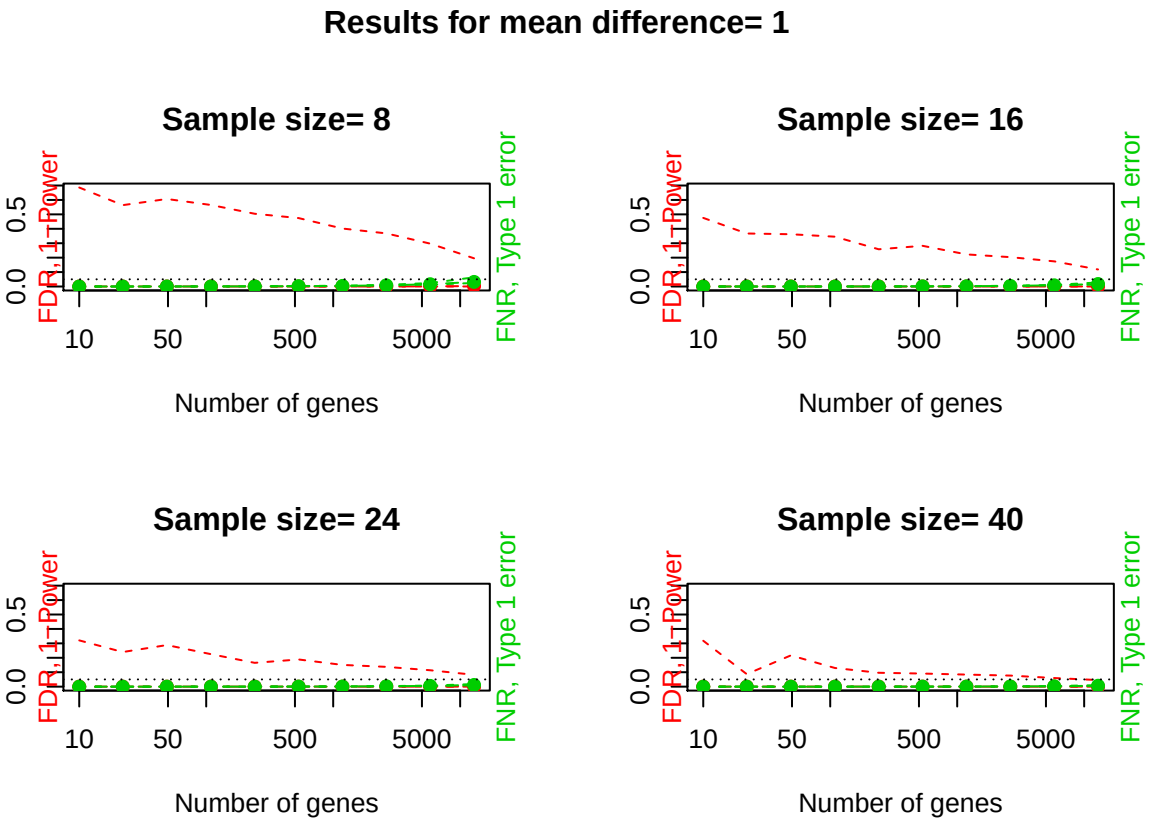
```
## perm= 1
## perm= 2
## perm= 3
## perm= 4
## perm= 5
## perm= 6
## perm= 7
## perm= 8
## perm= 9
## perm= 10
## perm= 11
## perm= 12
## perm= 13
## perm= 14
## perm= 15
## perm= 16
## perm= 17
## perm= 18
## perm= 19
## perm= 20
## perm= 21
## perm= 22
## perm= 23
## perm= 24
## perm= 25
```

```
## perm= 26
## perm= 27
## perm= 28
## perm= 29
## perm= 30
## perm= 31
## perm= 32
## perm= 33
## perm= 34
## perm= 35
## perm= 36
## perm= 37
## perm= 38
## perm= 39
## perm= 40
## perm= 41
## perm= 42
## perm= 43
## perm= 44
## perm= 45
## perm= 46
## perm= 47
## perm= 48
## perm= 49
## perm= 50
## perm= 51
## perm= 52
## perm= 53
## perm= 54
## perm= 55
## perm= 56
## perm= 57
## perm= 58
```

```
## perm= 59
## perm= 60
## perm= 61
## perm= 62
## perm= 63
## perm= 64
## perm= 65
## perm= 66
## perm= 67
## perm= 68
## perm= 69
## perm= 70
## perm= 71
## perm= 72
## perm= 73
## perm= 74
## perm= 75
## perm= 76
## perm= 77
## perm= 78
## perm= 79
## perm= 80
## perm= 81
## perm= 82
## perm= 83
## perm= 84
## perm= 85
## perm= 86
## perm= 87
## perm= 88
## perm= 89
## perm= 90
## perm= 91
```

```
## perm= 92
## perm= 93
## perm= 94
## perm= 95
## perm= 96
## perm= 97
## perm= 98
## perm= 99
## perm= 100
```

```
samr.assess.samplesize(samr.obj = samr.obj, data = data, dif = 1) %>%
samr.assess.samplesize.plot()
```




```
## NULL
```

The plots displayed show the estimaed FDR and FNR vs the numer of genes being tested for different (per group) sample sizes. The dashed lines represent the range from the 10^{th} to 90^{th} percentiles of the permutation distributions.

We will be testing over 20,000 genes so these graphs aren't very informative for our design, but we

might be able to infer appropriate sample sizes from them. Since our FDR (which equals 1 - power under the assumptions of this method) stays under 0.05 for the entire span of the plot for a sample size of 8 and the "confidence interval" is shrinking the entire way, we should be achieve over 95% power with 8 mice per group according to this method.

f)

The *pwr.t.test* function uses standard parametric methods for evaluating the power of t-tests. We will be assuming the standard deviation is the same for all genes, and we will also assume that the null hypothesis is true for all t-tests. This is the most constrained of all these methods

The FDR approach we assume that some small percent of null hypotheses for the t-tests are false and adjusts the p-values accordingly. This is a less conservative approach than the classical assumption that all null hypotheses are true.

ssize (realistically) does not assume that the standard deviation is the same for all genes. Instead it pools the standard deviations from pilot data and gives us a cumulative power curve based on the percent of genes that hit the desired power. This method is based off of t-tests as well and uses a Bonferonni correction for each test. This is the most conservative of all these methods.

PowerAtlas would have estimated the proportion of true positive, true negative, and false positive signals we'd see in based on p-values from pilot data.

samr gives estimates of false discovery and false negative rates based off of permutation distribution of gene 'scores' in pilot data. This is the least constrained of all these methods.