HW2 (125 points)

STAT 131A, Fall 2021

Due Friday, October 22

Question 1. Below I have copied over a function for running a permutation test from the book (and lab) that you can use for the rest of this homework.

```r
permutation.test <- function(group1,group2, FUN, repetitions){
  makePermutedStats<-function(){
      sampled <- sample(1:length(c(group1,group2)), size=length(group1),replace=FALSE)
      return(FUN(c(group1,group2)[sampled], c(group1,group2)[-sampled]))
  }
  stat.obs <- FUN(group1,  group2)
  stat.permute <-replicate(repetitions,makePermutedStats())
  p.value <- sum(stat.permute >= stat.obs) / repetitions
  return(list(p.value=p.value,observedStat=stat.obs,permutedStats=stat.permute))
}
```

a. (10 points) Inside `permutation.test` there is a function defined called `makePermutedStats`. I claim that this function simulates a single permutation of the data and returns the statistic defined by `FUN` (given by the user). Explain in detail how this function works and why this is a correct implementation for calculating the statistic of a single permutation of the data.

My answer is:
In 'makePermutedStats' function, funtion 'sample' is used to generate random samples without replacement from vector 'c(group1,group2)', with the same size of 'group1'. Then the generated samples in the pool i.e. 'c(group1,group2)' are passed to 'Fun' as the first group, while the rest data in the pool are passed to 'Fun' as the second group. Finally, 'Fun' will calculate the permuted statistic and return its value.

b. (5 points) Now explain the rest of the function, i.e. what the 4 lines after the definition of `makePermutedStats` do.

My answer is:
The first line is to pass the observation statistic calculated by 'Fun' to the 'stat.obs'.
The second line is to replicate the 'makePermutedStats' function for 'repetitions' times.
The third line is to calculate the p-value, which is the proportion of the permutation statistics equal or greater than the observation statistic.
The last line is to package all the results into a list of length 3, containing p-value, the observation statistic and the permutation statistics' vector.

Question 2. Evaluating whether a test gives valid p-values In this problem, we will go through a simple example of how you can use simulation from known distributions to determine whether a hypothesis test will perform well. We will only compare very simple settings, but this gives an idea of how you can use simulation to explore the performance of a test.

A Type I error is where you wrongly reject the null hypothesis when in fact it is true. We have said in class that a hypothesis test is a valid test if it correctly controls Type I error for a given level, i.e. if you perform a hypothesis test at level 0.05 *when in fact the null hypothesis is true*, then you will incorrectly reject the null hypothesis 5% of the time.

a. (15 points) Estimate the Type I error of the t-test when the data of the two groups is normal. Specifically, repeat the following simulation 10,000 times using the `replicate` function:

1) Simulate two groups of data each with 20 observations from a normal distribution. For the first group, let the standard deviation be 2.5, and for the second group the standard deviation be 5; both groups have mean 10.

2) Calculate the p-value of the t-test on this simulated data. Based on these simulations, report the type I error of the t-test.

```
# Enter code here for function that creates a single simulation of the data
# and returns the p-value for that single simulation:
set.seed(123)
f<-function(){
  group1 <- rnorm(20, 10, 2.5)
  group2 <- rnorm(20, 10, 5)
  return(t.test(group1, group2)$p.value)
}
# Enter code here that uses replicate to repeat this 10,000 times
replicate.stats <- replicate(10000, f())
# Now use that output to estimate the Type I error.
typeOneErr <- sum(replicate.stats < 0.05)/10000
typeOneErr
```

```
## [1] 0.0505
```

My answer is Type I error is 0.0505.

b. (10 points) Repeat the same simulation as above, only now make the data in both of the groups be generated from F distribution with parameters `df1=3` and `df2=24` (like in HW1); again make each group have 20 observations. What do you observe?

```
# Enter code here for simulation of the t-test from the F
# Reuse the code from above as applicable
set.seed(123)
f<-function(){
  group1 <- rf(20, df1 = 3, df2 = 24)
  group2 <- rf(20, df1 = 3, df2 = 24)
  return(t.test(group1, group2)$p.value)
}
# Enter code here that uses replicate to repeat this 10,000 times
replicate.stats <- replicate(10000, f())
# Now use that output to estimate the Type I error.
typeOneErr <- sum(replicate.stats < 0.05)/10000
typeOneErr
```

```
## [1] 0.044
```

My answer Type I error is 0.044, which is less than 0.05, meaning you will wrongly reject the null hypothesis when it is true at a probability less than 5%.

c. (15 points) We can also consider another type of error: when you do *not* reject the null, when in fact the two groups are different. This is called a Type II error and we can consider the probability of a test making a Type II error. An equivalent notion is that of the *power* of a test. The power is defined as the probability you will correctly reject the null when it is not true, i.e. power=1-P(Type II error).

We are going to repeat the simulation from part (a), only now focusing on calculating the power of the t-test. To do this, I now want you do the simulations from (a) with the mean in group 2 greater than the mean in group 1 (so that the null hypothesis is not true). Specifically, we are going to set the mean for group 2 be x units bigger than that of group 1, where x is going to be 1,2,3,4, and 5.

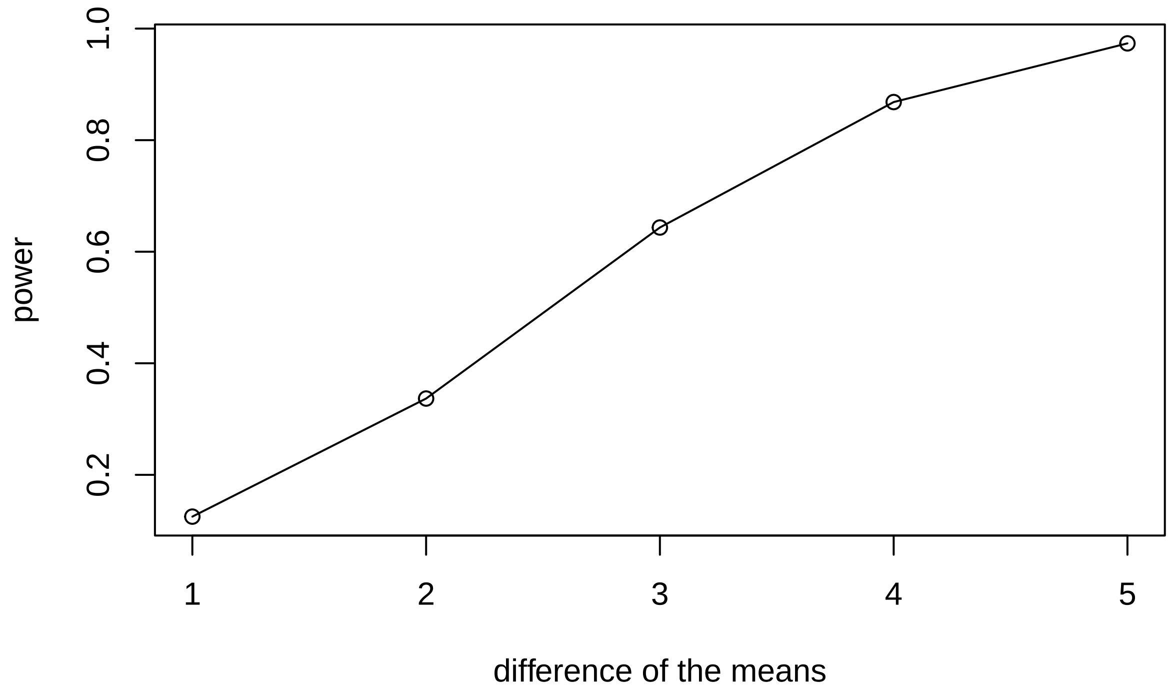
This requires you to redo the calculations in (a) for five different values. Instead of doing this manually, I want you to write a for-loop over the values of x. In other words, for each of x, your for-loop should calculate the power, i.e. the probability that the test *will* reject the null, under the same set up as part (a) but with the mean in group 2 greater than the mean in group 1.

```r
# Enter code here for calculating the power for different changes in the mean
# Reuse the code from a as applicable
set.seed(123)
f<-function(x){
  group1 <- rnorm(20, 10, 2.5)
  group2 <- rnorm(20, 10 + x, 5)
  return(t.test(group1, group2)$p.value)
}
typeTwoErr <- c()
for (x in 1:5) {
  replicate.stats <- replicate(10000, f(x))
  typeTwoErr <- c(typeTwoErr, sum(replicate.stats >= 0.05)/10000)
}
typeTwoErr
```

```
## [1] 0.8749 0.6632 0.3566 0.1317 0.0265
```

Demonstrate the results of your power calculations by plotting the power as a function of the difference in the means (i.e. power on y-axis, difference of the means on the x-axis).

```r
# Enter code here for calculating the power for different changes in the mean
# Reuse the code from a as applicable
power <- 1- typeTwoErr
plot(1:5, power, ylab = "power", xlab = "difference of the means")
lines(1:5, power)
```



Interpret what these results mean.

My answer is:
The power and difference of the means are positive correlation, which means when the difference of the means goes up, the power will go up too and the Type II error will decrease.

d. (10 points) Repeat the above power simulation in part (c), only now instead of different choices of x, we are going to set x to be 1 (i.e. group 2 to have a mean 1 unit bigger than the mean of group 1). Instead we are going to consider different possible sample sizes of each group.

You should write a for-loop that calculates the power of the test as you change the sample size of each of the groups to be 10,15,20,30,50.

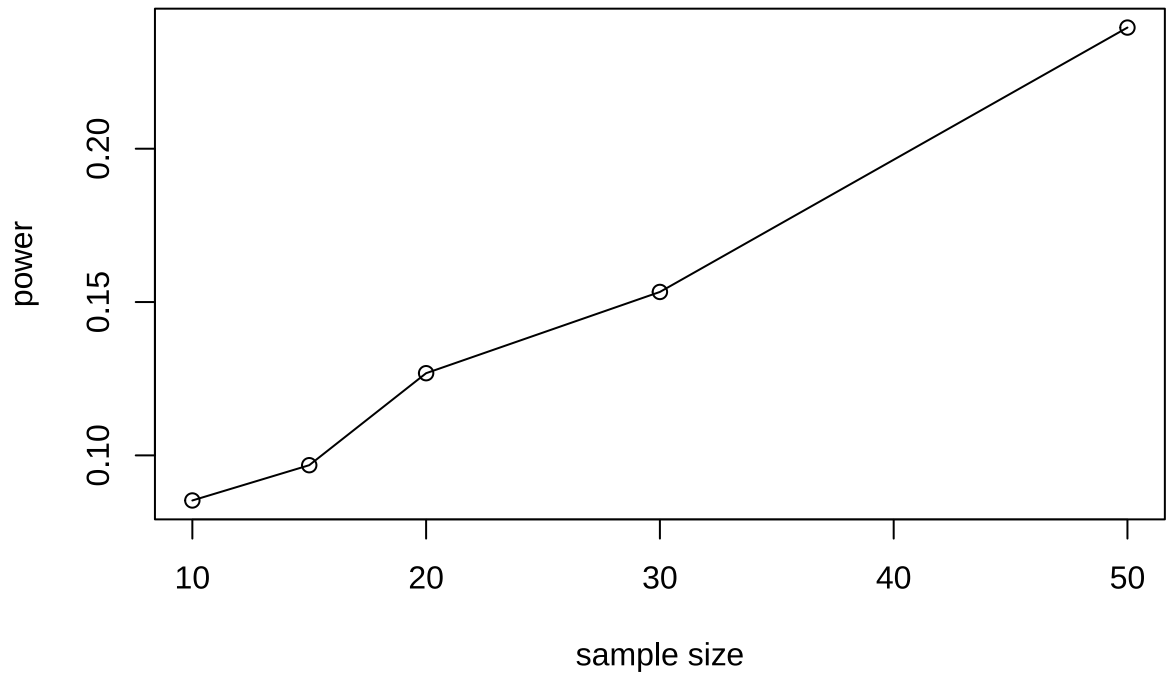
```r
# Enter code here for calculating the power for different sample sizes
# Reuse the code from (c) as applicable
set.seed(123)
f<-function(x, size) {
  group1 <- rnorm(size, 10, 2.5)
  group2 <- rnorm(size, 10 + x, 5)
  return(t.test(group1, group2)$p.value)
}
typeTwoErr <- c()
for (size in c(10, 15, 20, 30, 50)) {
  replicate.stats <- replicate(10000, f(1, size))
  typeTwoErr <- c(typeTwoErr, sum(replicate.stats >= 0.05)/10000)
}
```

```r
typeTwoErr
```

```
## [1] 0.9147 0.9032 0.8732 0.8467 0.7605
```

Similarly demonstrate the results by plotting the power as a function of the sample size and interpret the results.

```r
# Enter code here for calculating the power for different changes in the mean
# Reuse the code from (c) as applicable
power = 1 - typeTwoErr
plot(c(10, 15, 20, 30, 50), power, ylab = "power", xlab = "sample size")
lines(c(10, 15, 20, 30, 50), power)
```



My answer is:
The power and the sample size are positive correlation, which means when the the sample size goes up, the power will go up too and the Type II error will decrease.

e. (5 points) How could you compare the validity and power of the permutation test using the t-statistic as compared to the t-test? (Just use words to describe what you would do, without actually coding it)

My answer is:
1. The first step is to generate samples from the data pool.
2. The second step is to perform permutation test and t-test on the same sample, for which we need to replicate the calculation for t-statics for multiple times to do the permutation tests.

3. The third step is to calculate the Type II error as well as the power for both tests. Generally speaking, if the test has larger power, it will have a lower Type II error rate, which is a better choice.

Question 3. Consider the data from HW1 containing information on predicting heart disease in patients. We have provided another copy of the data with this HW, to avoid having to find the data from last time, but it is the same dataset. Read the data in again,

```
heart<-read.csv("heartDisease.csv",header=TRUE)
head(heart)
```

```
##   age sex cp trestbps chol fbs restecg thalach exang oldpeak slope ca thal num
## 1  63   1  1      145  233   1       2     150     0     2.3     3  0    6   0
## 2  67   1  4      160  286   0       2     108     1     1.5     2  3    3   2
## 3  67   1  4      120  229   0       2     129     1     2.6     2  2    7   1
## 4  37   1  3      130  250   0       0     187     0     3.5     3  0    3   0
## 5  41   0  2      130  204   0       2     172     0     1.4     1  0    3   0
## 6  56   1  2      120  236   0       0     178     0     0.8     1  0    3   0
```

a. (5 points) Perform a t-test comparing serum cholestoral in mg/dl (`chol`) between those patients with chest-pain type any type of angina (i.e. `cp` either `typical angina` or `atypical angina`) and those patients with non-anginal pain. What do you conclude?

```
# code for running t-test
angina <- heart$chol[heart$cp %in% c(1, 2)]
nonanginal <- heart$chol[heart$cp == 3]
t.test(angina, nonanginal)
```

```
##
##  Welch Two Sample t-test
##
## data:  angina and nonanginal
## t = -0.32992, df = 144.25, p-value = 0.7419
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -18.91046  13.50048
## sample estimates:
## mean of x mean of y
##  242.6806  245.3855
```

My answer is: For t-test, we don't reject the null hypothesis that true difference in means is equal to 0, since the p-value is 0.74, which is higher than 0.05.

b. (5 points) Repeat the above, using a permutation test based on the t-statistic instead and give your conclusions.

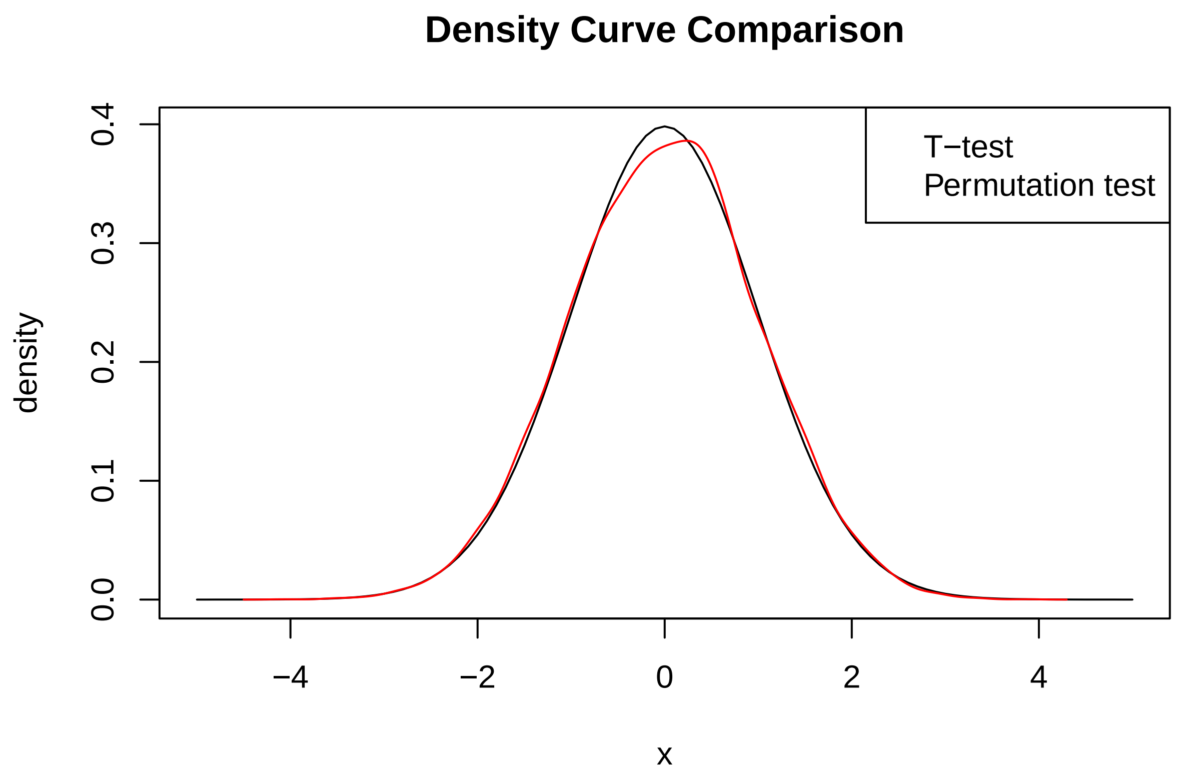
```
# code for running permutation test
stat.t <- function(group1, group2) {
  return(t.test(group1, group2)$statistic)
}
stat.permute <- permutation.test(angina, nonanginal,
                                 FUN = stat.t, repetitions = 10000)
p.value = sum(abs(stat.permute$permutedStats) > stat.permute$observedStat)/10000
p.value
```

```
## [1] 1
```

My answer is:
For this permutation test based on the t-statistic, we don't reject the null hypothesis that true difference in means is equal to 0, since the p-value is 1, which is higher than 0.05.

c. (10 points) Compare the null hypothesis of the t-test and the permutation test for this data by plotting the density curve for both null hypotheses on the same plot. For the permutation test, you should plot an kernel density estimate of the curve (not the histogram). Color the t-test black and the permutation test red and provide a legend. How do they compare?

```
# code for plotting the densities of the two null distributions
curve(dt(x, df = 144.25), xlim = c(-5, 5),
      main = "Density Curve Comparison", ylab = "density")
lines(density(stat.permute$permutedStats), col = "red")
legend("topright", legend = c("T-test", "Permutation test"),
       col = c("black", "red"))
```



My answer is:
The curve of permutation test and t-test looks very similar in this case, since the degree of freedom is quite large for t-test (144.25 from the previous output), the t-test curve which represents a t-distribution will approxiamate to the normal distribution.

d. (20 points) Of greater interest is the actual diagnosis (`num`). Recall, the diagnosis was encoded from 0-4, with 0 being no heart disease diagnosed. Use a permutation test based on the t-statistic to test the difference in the serum cholestoral in mg/dl `chol` between all of these diagnosis levels, i.e. all pairwise comparisons.

Steps that you will need:

- Create a matrix that gives all the pairwise combinations of the levels of `num` using the `combinations` function in the `gtools` package. You may need to install this package with `install.packages` if you are working on your own computer.
- Create a function that takes x, the pair of levels being compared, and calculates the results of the permutation test
- Run this function on each combination using the `apply` function.

I will go over the code that you will need next week. You should reuse this code and adapt it to this problem.

Now add code, based on adapting the lecture code, to do pair-wise permutation tests of all of the (5) levels of diagnosis and interpret your results

```r
# Code for doing permutation tests on all combinations
library(gtools)
combn <- combinations(5, 2, v = 0:4)
p.values <- apply(combn, 1, function(x) {
  group1 <- heart$chol[heart$num == x[1]]
  group2 <- heart$chol[heart$num == x[2]]
  stat.permute <- permutation.test(group1, group2, FUN = stat.t,
                       repetitions = 10000)
  p.value = sum(abs(stat.permute$permutedStats) > stat.permute$observedStat)/10000
  return(p.value)
})
p.values
```

```
##  [1] 1.0000 1.0000 1.0000 1.0000 1.0000 0.7970 1.0000 0.2554 0.7162 1.0000
```

My answer is:/ Judging from the p-values, we can see that the difference in the serum cholestoral between all of these diagnosis levels is not significant, so that we don't reject the null hypothesis that true difference in means is equal to 0.

e. (10 points) Use Bonferroni adjustments to correct for multiple testing of your above results. How did this affect your results?

```r
# code to use Bonferroni multiple testing correction
Bonf.p.values <- p.adjust(p.values, method = "bonferroni")
Bonf.p.values
```

```
##  [1] 1 1 1 1 1 1 1 1 1 1
```

My answer is:
After preforming the Bonferroni adjustments, some of the p-values increased which means we are more confident at the result. Still, we don't reject the null hypothesis, since 0.05/10 = 0.005, and there's no p-value less than 0.005.

f. (5 points) 125–200 mg/dL is a normal range of serum cholesteral, though it varies by person what is healthy. Given this, is the difference of means a good statistic, or would you propose a different statistic?

My answer is:
I would propose the statistic to be the serum cholesteral level higher than 200 mg/dL or less than 125 mg/dL, since they are out of the normal range of serum cholesteral and can cause a hazzard for related health problems. So instead of comparing the diffence of means of serum cholestera, I think to focus on the outliers will be a better idea since we want to avoid probable health issues.