POLS0012 Causal Analysis: Tutorial Exercise 4

In this question, we'll use a simulated dataset to illustrate randomisation inference, based on Table 3.3 in the Gerber and Green textbook, an example of an experiment created with block randomisation and 14 observations. We'll code the exact p-values by hand first, then using R's built-in code from the `ri` package. We'll also illustrate it with and without accounting for blocked randomisation. The dataset is called "a" and is in the file "t4_data.Rda". It contains the following variables for 14 observations:

- Y: Outcome variable
- *block*: Two blocks used for randomisation, 1 or 2
- *tr*: =1 if in the treatment group, 0 if in the control group

You first need to install and load R's `ri` package using the code `install.packages(‘‘ri’’)` and `library(‘‘ri’’)`

a) Calculate the ATE and its p-value from the experiment, ignoring the blocking (suppose we didn't realise this was a block-randomised experiment). Store the ATE as an object

   Code:

   ```
   reg1 <- lm(Y~tr,data=a)
   summary(reg1)
   ate <- reg1$coef[2]
   ```

   The ATE is -6.43 with a p-value of 0.052

b) Using the `genperms()` function, create a matrix of all possible permutations of treatment assignment, ignoring the blocking. How many possible permutations are there?

   Code:

   ```
   perms <- genperms(a$tr)
   ```

   There are 3432 permutations of the treatment vector, ignoring blocking.

c) Now, create the null distribution of ATEs using the permutations matrix:

   i) Create an empty vector called "ates" using `ates <- c()`.

   ii) Using a `for()` loop, fill in this vector with the ATE under each possible permutation of treatment
   Code Hint: The columns of the matrix from (b) give you each possible permutation of treatment

Code:

```
ates <- c()
for(i in 1:ncol(perms)){
  ates[i] <- mean(a$Y[perms[,i]==1]) - mean(a$Y[perms[,i]==0])
}
```

Try to understand what we did here. We took the Y vector, which under the sharp null contains all of the potential outcomes under treatment and control: the observed outcomes are the full set of potential outcomes, because it doesn't matter whether someone is assigned to treatment or control. We then used Y to calculate the ATE under every possible randomisation, telling us the null distribution of possible ATEs. This allows us to ask: under the null, how likely is our ATE to arise by chance alone, merely due to the quirks of randomisation.

d) Calculate the exact two-tailed p-value using the estimated ATE and the null distribution. Is it higher or lower than the p-value calculated the traditional way in (a)?
Hint: You need to calculate the proportion of the null distribution that is at least as extreme as the estimated ATE. It can help to start with a drawing

Code:

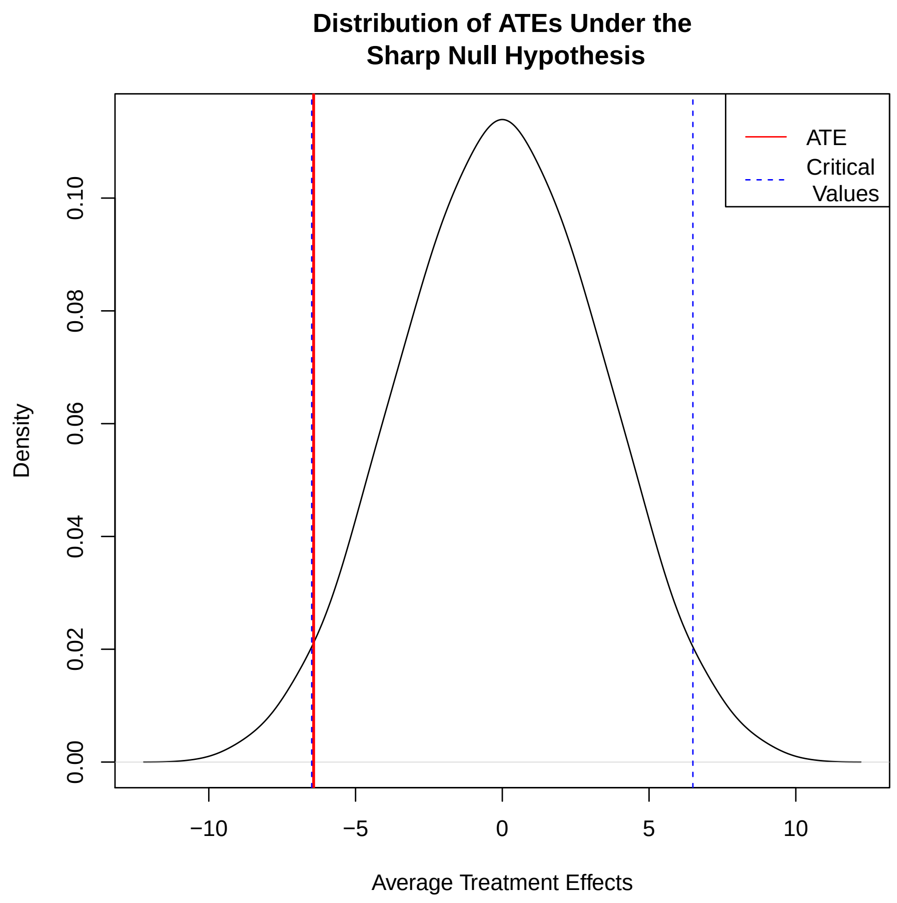
```
(length(ates[ates<=ate]) + length(ates[ates>=-ate])) / ncol(perms)
```

The exact p-value is 0.061. It is higher than the p-value estimated in (a) - quite a common occurrence.

e) To visualise what is happening, plot the null distribution as a density or histogram, adding vertical lines for the estimated ATE and the critical values for the exact hypothesis test
Code Hint: you can add vertical lines to a plot using `abline(v=)` and you can use the `quantile()` function to find the critical values

Code:

```
plot(density(ates),
     xlab="Average Treatment Effects",
     main="Distribution of ATEs Under the\n Sharp Null Hypothesis")
abline(v=ate,col="red",lwd=2)
abline(v=quantile(ates,0.025),col="blue",lty=2)
abline(v=quantile(ates,0.975),col="blue",lty=2)
legend("topright",c("ATE","Critical\n Values"),lty=c(1,2),col=c("red","blue"),ncol=1)
```



f) Now we'll carry out the inference automatically using R's built-in functions from the `ri` library. The results should be identical to your hand-coded estimate in (d):

  i) Declare the potential outcomes under the sharp null using the `genouts()` function. It takes three arguments: the observed outcome variable, the treatment variable, and the null hypothesis specified as `ate=`

  ii) Generate the null distribution using the `gendist()` function. It takes two arguments: the potential outcomes from step (i) and the matrix of permutations from question (b)

  iii) Calculate the p-value and plot the results using the `dispdist()` function. It takes two arguments: the distribution from step (ii) and the estimated ATE

Code:

```
Y_null <- genouts(a$Y,a$tr,ate=0)
distout <- gendist(Y_null,perms)
dispdist(distout,ate)
```

g) Finally, let's re-estimate everything, this time accounting for the blocking. First, calculate the ATE and its p-value from the experiment, accounting for the blocking. Is it different to (a)?

Hint: Remember that you need to control for block membership

Code:

```
reg2 <- lm(Y~tr + factor(block),data=a)
summary(reg2)
ate <- reg2$coef[2]
```

The ATE is unchanged but the p-value has fallen a lot compared to (a), to 0.0014 as we would expect: blocking allows us to estimate with greater precision

h) Using the `genperms()` function, create a matrix of all possible permutations of treatment assignment, accounting for the blocking. How many possible permutations are there, compared to part (b)?

Code Hint: You need to add the argument `blockvar=` to the `genperms()` function

Code:

```
perms <- genperms(a$tr,blockvar=a$block)
```

There are now 1400 possible permutations compared to the 3432 in (b). This shows how blocking reduces the number of possible randomisations

i) Repeat (f), this time using the permutations matrix from question (h). Comment on how the null distribution has changed, compared to (f)

Code:

```
Y_null <- genouts(a$Y,a$tr,ate=0)
distout <- gendist(Y_null,perms)
dispdist(distout,ate)
```

The null distribution is narrower, with fewer extreme values compared to (f). Blocking is useful because it rules out 'crazy' randomisations that result in extreme ATEs. This exercise demonstrates it directly.