1. **Completely randomized study** - Sample code for parts 1a, 1b, 1e is listed at the end of the solution set.

(a) There are 924 possible randomizations (ways to choose 6 treatment units out of 12). 250 randomizations yield absolute test statistics 2.0183 greater than or equal to the observed -2.0183. This yields \( p = .2705 \).

(b) Simulation results will vary. I got .286.

(c) The aim here was to have you recognize and evaluate the role of Monte Carlo variability in the simulations. The standard error of the estimated p-value is \( \sqrt{p(1-p)/1000} \); with \( p = .2705 \) the standard error is approx .014. Thus a simulation should be within .03 of the correct answer.

(d) The two-sample t-test (the s.d. in each sample is almost identical) yields \( t = -1.009 \) which yields \( p = .3368 \) for \( t \) with 10 d.f. Note the data are already logarithms – sorry for any confusion. Clearly the two methods are different. The randomization inference is exact and is based on the randomization distribution for these precise 12 children. The t-test is a superpopulation inference relating to the means of the populations from which the two samples are drawn – it requires an assumption of a normal distribution in the population. Though there’s nothing here that proves it, the two approaches agree in large samples.

(e) The attached code shows I did this. A key point is that the definition of the p-value in my sample program does not work when the treatment effect is not zero. (Here’s a quick way to think about it. If the treatment effect is very large, say 1000, then any randomization will produce a large test statistic (except the observed randomization which will only use the observed data). Thus for really large treatment effect the p-value should be quite small yet all randomized test statistics will have values larger than the observed in absolute value.) It turns out tha a 95% CI is (-6.9, 2.5) and a 90% CI is (-5.5, 1.7).

2. **Randomization with pairing** - Sample code for 2a and 2b is provided below.

(a) There are only \( 2^6 = 64 \) possible randomizations now with p-value = .375.

(b) Monte Carlo simulation is .361 (in my case).

(c) Paired t-test yields \( t = -0.9956 \) and \( p = .3652 \).

(d) If there is information used to make the pairs that is correlated with the outcome, then pairing should be valuable. In this case, the larger p-value and the lack of correlation of the outcomes across pairs both suggest that pairing was not useful here. (Perhaps because it was just made up pairs!!)

3. **Theory**

(a) There are several ways to write the test statistic. One natural way is \( \hat{\tau} = \frac{1}{N_T} \sum_i W_i Y_i(1) - \frac{1}{N_C} \sum_i (1 - W_i) Y_i(0) \) with \( N_T = N_C = 6 \).

(b) Under complete randomization only \( W_i \)'s are random. Then \( E(\hat{\tau}) = \frac{1}{N_T} \sum_i E(W_i)Y_i(1) - \frac{1}{N_C} \sum_i (1 - E(W_i)) Y_i(0) \). Moreover under randomization it is clear that \( E(W_i) = Pr(W_i = 1) = 6/12 = 1/2 \) which yields \( E(\hat{\tau}) = \frac{1}{2N_T} \sum_i Y_i(1) - \frac{1}{2N_C} \sum_i Y_i(0) = \frac{1}{12} \sum_i (Y_i(1) - Y_i(0)) = \tau \) (the finite population causal estimand). Thus our test statistic is an unbiased estimate of the average treatment effect.

(c) Note that the test statistic is the same in this case and \( E(W_i) = 1/2 \) for each unit again. Thus same proof works. Many of you came up with an alternative proof by considering there were only two possible randomizations for each pair.

4. **Projects**

Thanks for the proposals. That was helpful. I’ve made comments on each and am happy to discuss.

**SAMPLE R CODE**

```r
Problem 1 -- parts a, b, e
#
# compute test statistic
#
y <- c(0.06,1.72,2.19,7.32,7.53,7.62,8.62,1.48,8.93,9.57,2.65,7.30)
tobs <- (sum(y[1:6]) - sum(y[7:12])) / 6
#
# set up ytrt and yctl under sharp null of zero effect
#
ytrt <- c(0.06,1.72,2.19,7.32,7.53,7.62,8.62,1.48,8.93,9.57,2.65,7.30)
yctl <- c(0.06,1.72,2.19,7.32,7.53,7.62,8.62,1.48,8.93,9.57,2.65,7.30)
```
Problem 2, parts a and b

```r
# set up vector to hold all randomization result
#
out <- rep(0,924)
#
# loop thru all randomizations and compute test statistic
#
cnt <- 0
for (i1 in (1:7)) {
  for (i2 in ((i1+1):8)) {
    for (i3 in ((i2+1):9)) {
      for (i4 in ((i3+1):10)) {
        for (i5 in ((i4+1):11)) {
          for (i6 in ((i5+1):12)) {

            cnt <- cnt + 1
            x <- c(i1,i2,i3,i4,i5,i6)
            trtsum <- sum(ytrt[x])
            ctlsum <- sum(yctl[-x])
            out[cnt] <- trtsum/6 - ctlsum/6
          }
        }
      }
    }
  }
}

pval <- 2*min(sum(out <= tobs), sum(out >= tobs))/924
#
# simulation version using the R sample command to choose the sample
#
outsim <- rep(0,1000)
for (i in (1:1000)) {
  ysamp <- sample(y,6)
  trtsum <- sum(ysamp)
  outsim[i] <- trtsum/6 - (sum(y)-trtsum)/6
}
pval <- sum(abs(outsim) >= abs(tobs))/1000
#
# confidence interval - calculation from (a) for a range of "true" treatment effects k (from -10 to 10 in steps of .1 here)
# obtain pvalues for each k .... k is in 90% CI if pvalue > .10
#
y <- c(0.06,1.72,2.19,7.32,7.53,7.62,8.62,1.48,8.93,9.57,2.65,7.30)
tobs <- (sum(y[1:6]) - sum(y[7:12])) / 6
pvals <- rep(0,201)
k <- c(-100:100)/10
for (i in (1:length(k))) {
  ytrt <- c(0.06,1.72,2.19,7.32,7.53,7.62,8.62+k[i],1.48+k[i],8.93+k[i],9.57+k[i],2.65+k[i],7.30+k[i])
yctl <- c(0.06-k[i],1.72-k[i],2.19-k[i],7.32-k[i],7.53-k[i],7.62-k[i],8.62,1.48,8.93,9.57,2.65,7.30)
out <- rep(0,924)
cnt <- 0
for (i1 in (1:7)) {
  for (i2 in ((i1+1):8)) {
    for (i3 in ((i2+1):9)) {
      for (i4 in ((i3+1):10)) {
        for (i5 in ((i4+1):11)) {
          for (i6 in ((i5+1):12)) {

            cnt <- cnt + 1
            x <- c(i1,i2,i3,i4,i5,i6)
            trtsum <- sum(ytrt[x])
            ctlsum <- sum(yctl[-x])
            out[cnt] <- trtsum/6 - ctlsum/6
          }
        }
      }
    }
  }
}
pvals[i] <- pval
}
```
# set up y vector now as 6 x 2
#
y <- c(0.06,1.72,2.19,7.32,7.53,7.62,8.62,8.93,9.57,2.65,7.30)
dim(y) <- c(6,2)
tobs <- mean(y[,2]) - mean(y[,1])
#
# set up vector to hold all randomization result
#
out <- rep(0,64)
#
# loop thru all randomizations and compute test statistic
#
cnt <- 0
for (i1 in (1:2)) {
    for (i2 in (1:2)) {
        for (i3 in (1:2)) {
            for (i4 in (1:2)) {
                for (i5 in (1:2)) {
                    for (i6 in (1:2)) {
                        cnt <- cnt + 1
                        ctlsum <- sum(y) - trtsum
                        out[cnt] <- trtsum/6 - ctlsum/6
                    }
                }
            }
        }
    }
}
pval <- 2*min(sum(out <= tobs), sum(out >= tobs))/64
#
# simulation version using the R sample command to choose the sample
#
outsim <- rep(0,1000)
for (i in (1:1000)) {
    ix <- rep(1,6)
    ix <- ix + rbinom(6,1,0.5)
    ctlsum <- sum(y) - trtsum
    outsim[i] <- trtsum/6 - (sum(y)-trtsum)/6
}
pval <- sum(abs(outsim) >= abs(tobs))/1000