Modern Statistical Methods Assignment

Nick Head

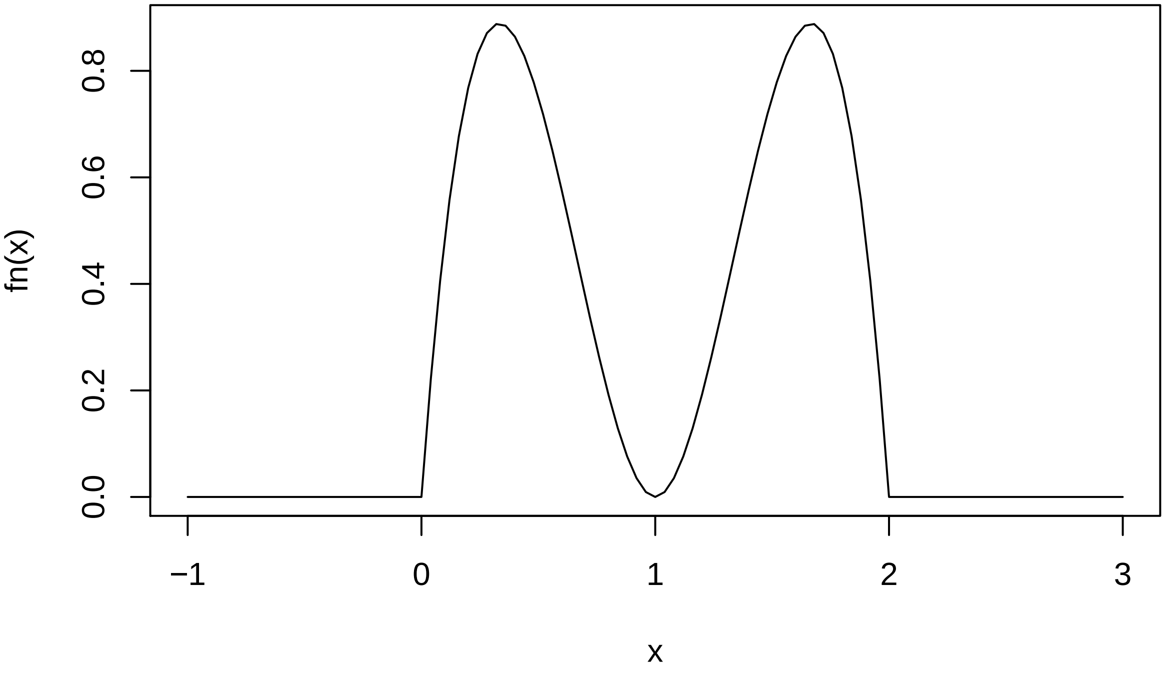
15 April 2015

Question 1

$$f(x) = \begin{cases} 6x(1-x)^2 & \text{if } 0 < x < 1 \\ 6(2-x)(1-x)^2 & \text{if } 1 \leq x < 2 \\ 0 & \text{otherwise} \end{cases}$$

a)

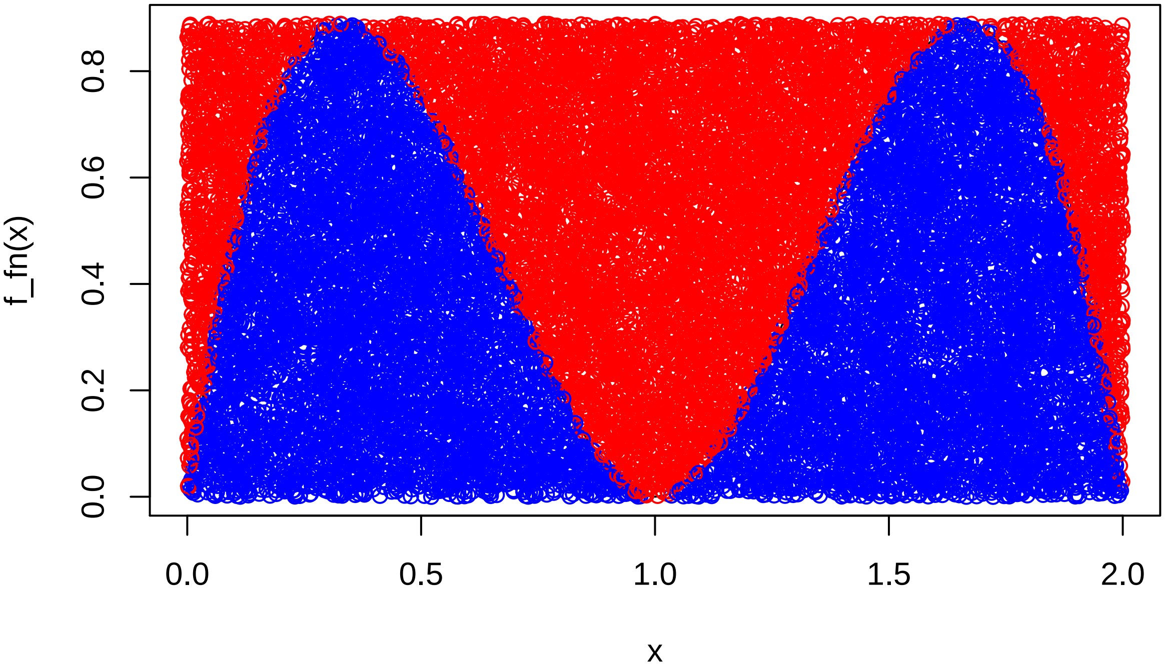
```
fn <- function(x)
  {
    ifelse( x > 0 & x < 1, 6*x*(1 - x)^2,
            ifelse( x >= 1 & x < 2, 6*(2-x)*(1 - x)^2,
                    0))
  }

curve(fn, from = -1, to = 3)
```



b)

See Appendix 1 for further implemention details of the Accept-Reject algorithm.

```
Nsim = 10000
set.seed(1234)
x = SimulateAcceptReject(Nsim,
                         f_fn = fn,
                         g_fn = function (y) {dunif(y, min = 0, max = 2)},
                         randg_fn = function (Nsim) {runif(Nsim, min = 0, max = 2)},
                         x_interval = c(0, 2))
```



c)

i)

$$E[log(X)] = -0.3074639$$

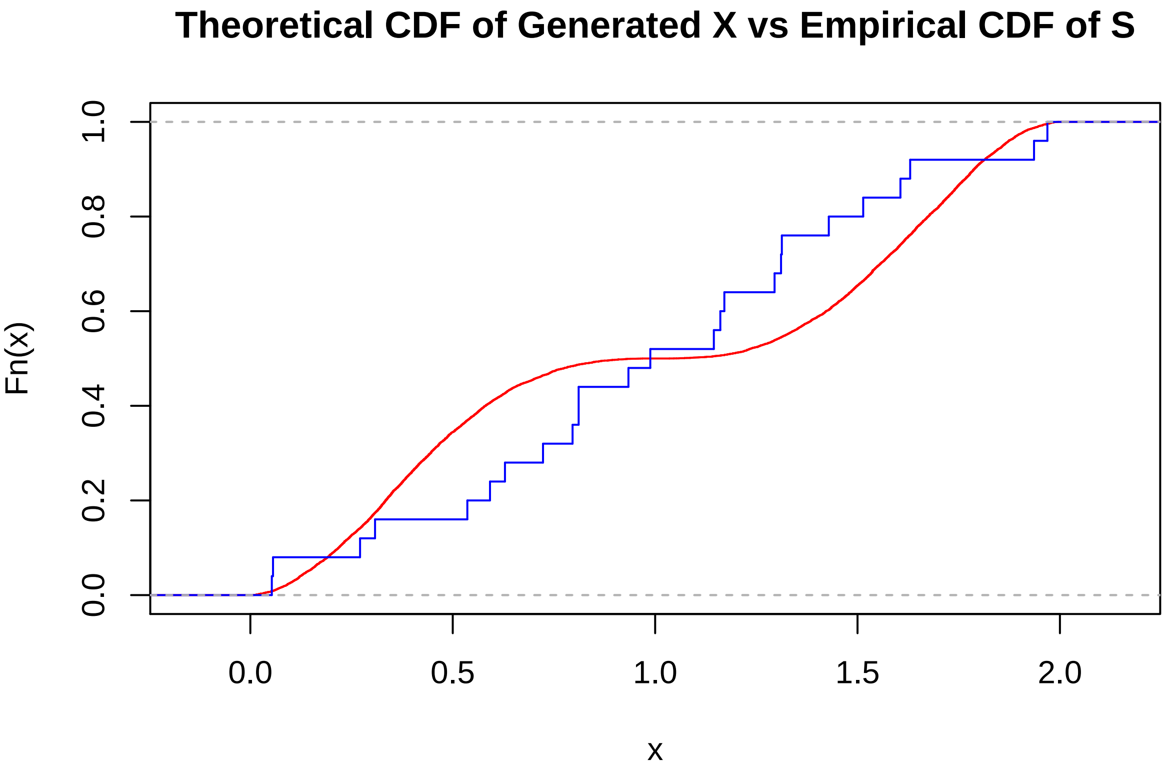
ii)

$$P(log(X) < 0.5) = 0.7794$$

iii)

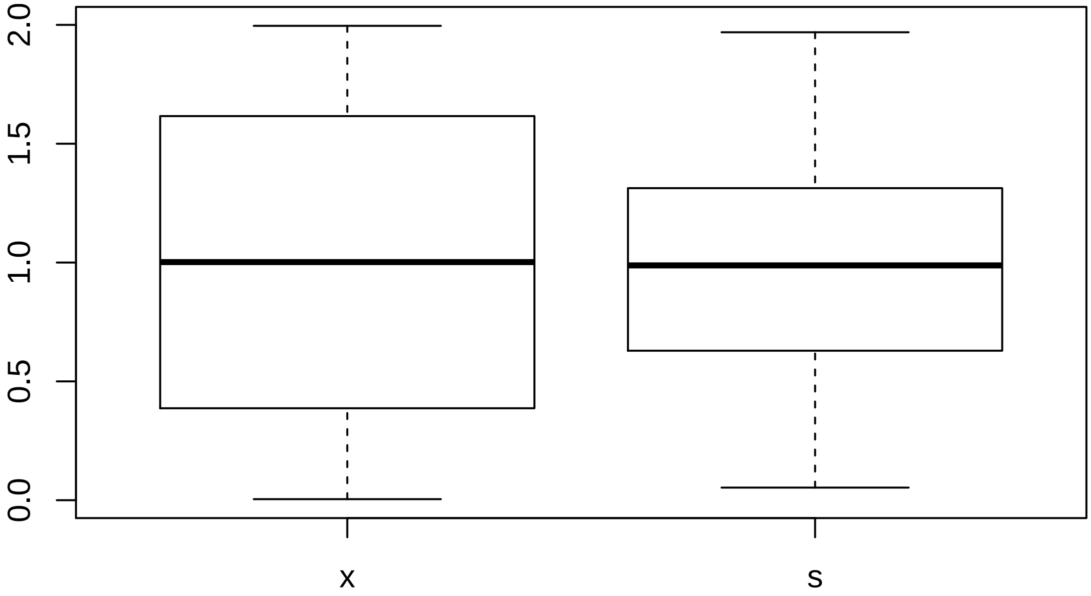
$$E[(log(X)|X > 0.5] = 0.2457718$$

d)

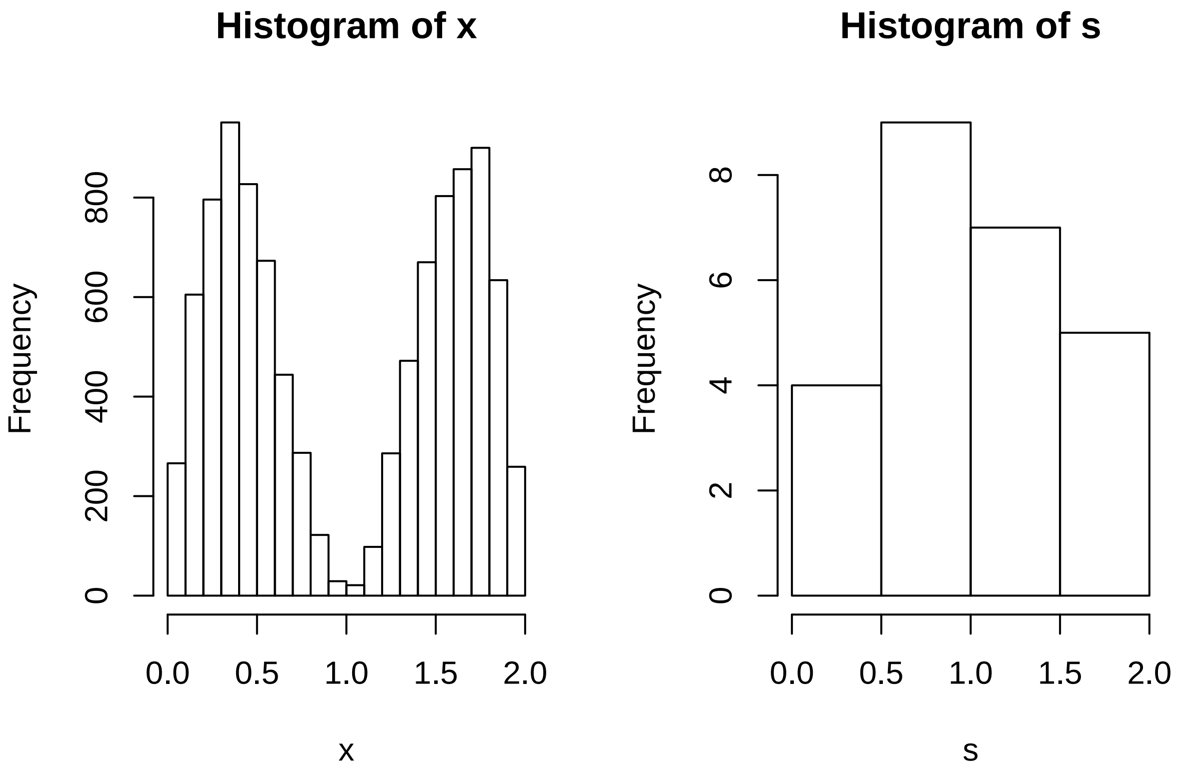

Theoretical CDF of Generated X vs Empirical CDF of S
Fn(x)
0.0
0.2
0.4
0.6
0.8
1.0
0.0
0.5
1.0
1.5
2.0
x

e)

	x	s
1	Min. :0.004455	Min. :0.0530
2	1st Qu.:0.386981	1st Qu.:0.6290
3	Median :1.001894	Median :0.9880
4	Mean :1.002557	Mean :0.9995
5	3rd Qu.:1.616175	3rd Qu.:1.3130
6	Max. :1.995972	Max. :1.9690



As can be seen from the summary statistics and boxplots, the means of both x and s appear to be almost identical, as are the minimum and maximum values, indicating that both distributions appear to have similar supports. The main difference appears to be in the size of the interquartile ranges and hence the variances. x appears to have a much larger variance than s although both do appear to be somewhat symmetric around the mean.



The histograms however appear to tell a different story. While x clearly appears bimodal, s clearly only appears to have one mode.

f)

```
ks = ks.test(x, s)
ks
```

```
##
##  Two-sample Kolmogorov-Smirnov test
##
## data:  x and s
## D = 0.2143, p-value = 0.2022
## alternative hypothesis: two-sided
```

The Kolmogorov-Smirnoff tested conducted above indicates that we cannot reject the null hypothesis that the two datasets come from different distributions (at the 5% level, with a p-value of 0.2022221). The statistic D_n is calculated as the largest absolute deviation between the empirical distribution and the theoretical one against which we are comparing it. For this run the D_n statistic is calculated as 0.2143. This can be seen visually in the chart in section d) as roughly the largest deviation between the theoretical CDF of x and the empirical CDF of s.

This does however contradict some of our earlier findings. Even though some of the statistics such as mean and support of the distributions do indeed appear to be similar, by looking at the histograms the bimodal nature of the Camel distribution is clearly missing in the s distribution. Without this key feature of the empirical dataset, we should be quite cynical of the results of this non-parametric test.

Question 2

a)

```
twain = c(0.225, 0.262, 0.217, 0.240, 0.230, 0.229, 0.235, 0.217)

snod = c(0.209, 0.205, 0.196, 0.210, 0.202, 0.207, 0.224, 0.223, 0.220, 0.201)

ttest = t.test(twain, snod)
```

The t-test gives a p-value of: 0.0031561

b)

The following test does a randomized permutation. The code is detailed in Appendix 2

```
pvalue = RandomPermutationTest(twain, snod)
```

The random permutation test gives a p-value of: 0.002

This can be compared with an exact randomization test:

```
library(exactRankTests)
exact = perm.test(x = twain, y = snod, exact = TRUE)
```

The exact randomization test gives a p-value of: 0.0011198

c)

The following test uses a bootstrap approach. The code is detailed in Appendix 3

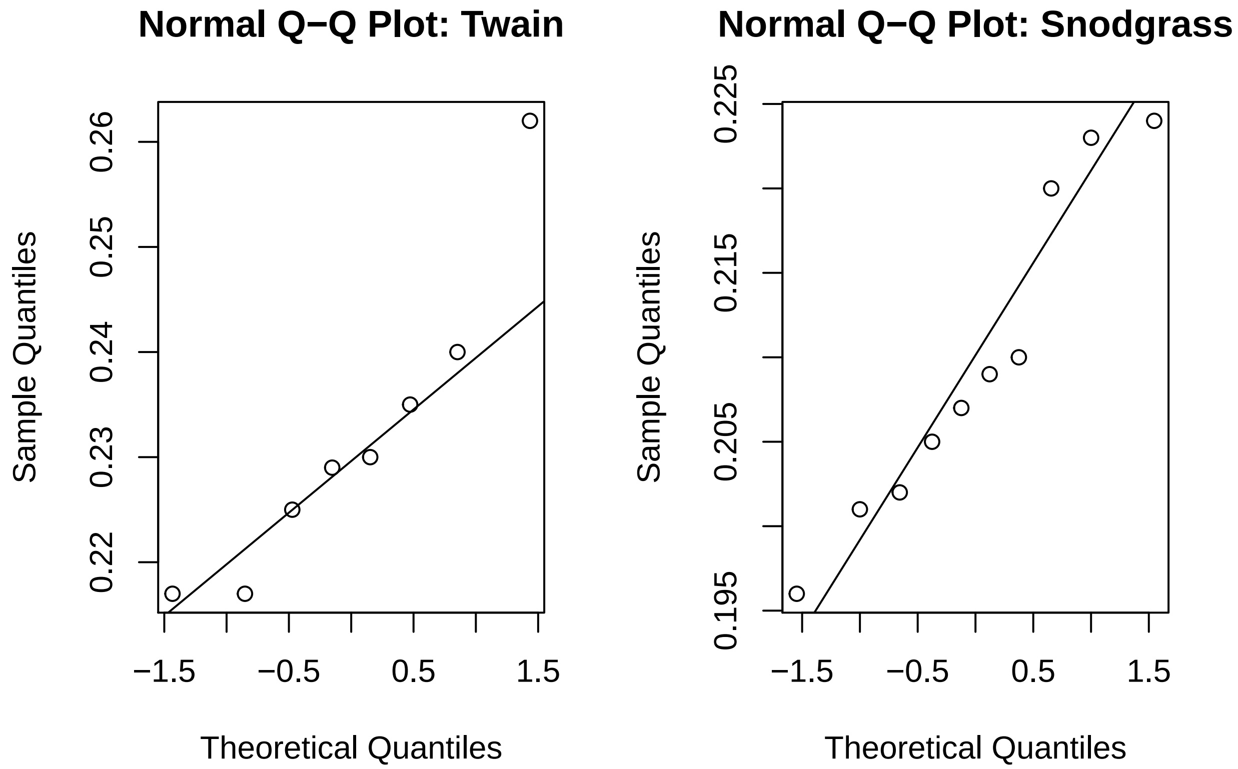
```
boot.p = BootstrapTTest(twain, snod)
```

The bootstrapping approach gives us a p-value of: 0.001

d)

Using all three tests (t-test, permutation and bootstrapped t-test) the p-values are between 0.003 and 0.001, therefore giving very strong support (at the $\alpha = 0.01$ level) to reject the null hypothesis of equality of means. Based off this we are therefore led to believe that the two sets of letters are indeed written by two different people.

Even though the three sets of tests all lead to the same conclusions we must still be wary of whether the tests are appropriate for this analysis. With a standard t-test although it is robust to non-normality, it is still worth checking if the data is somewhat normal considering we are dealing with proportions which generally would be definitely non-normal. For this reason we can conduct a QQ test of the Twain and Snodgrass datasets:



The QQ tests do not indicate any significant departures from normality therefore we have no reason to distrust the conclusions from the t-test. A further reason to potentially distrust the t-test would be due to the small sample sizes. For this reason we are satisfied that both the permutation and bootstrap tests also give us the same asymptotic results.

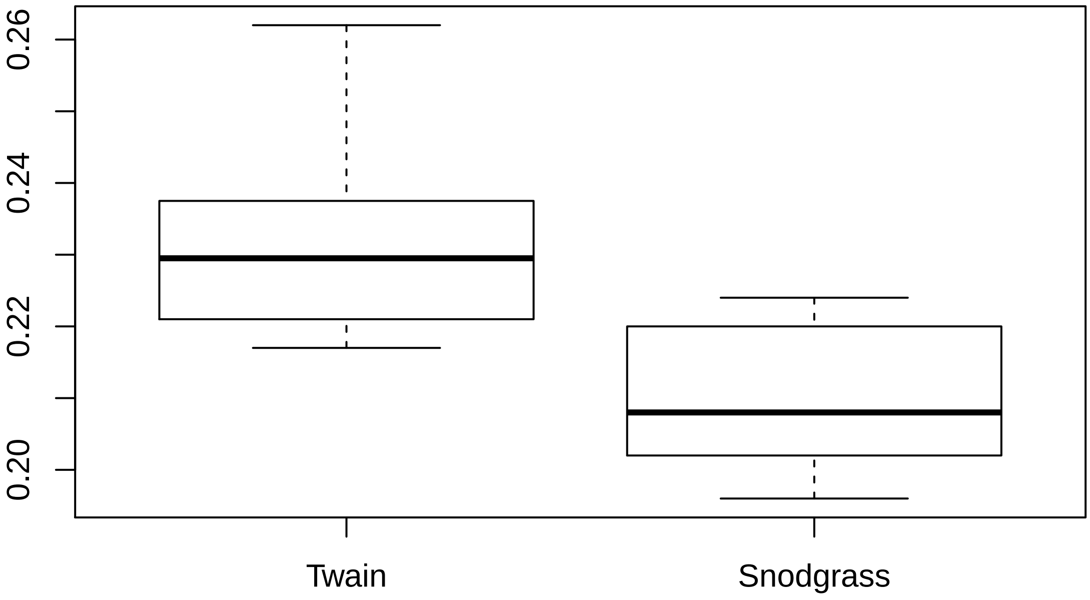
Discuss appropriateness of t-test considering non-normality of data

CI's?

Assumption of equal variance

Difference of means vs studentized t

Permutation test is more exact however is less applicable, as one can only test the null hypothesis of identical distributions. The bootstrap approach is less precise (accuracy is only guaranteed as the sample size tends to infinity) however it is more versatile for instance in this case being able to test equal means with unequal variances.


0.20
0.22
0.24
0.26
Twain
Snodgrass

Appendix 1: Accept-Reject Code Listing

```
SimulateAcceptReject <- function(Nsim, f_fn, g_fn, randg_fn, x_interval = c(0,
    1), plot = TRUE) {
    # M is found by finding the maximum of f(x)/g(x) over [0,1]
    M = optimize(f = function(x) {
        f_fn(x)/g_fn(x)
    }, interval = x_interval, maximum = TRUE)$objective

    if (plot == TRUE) {
        # graphing logic
        ylim <- c(0, M * g_fn(randg_fn(1)))
        xlim <- x_interval
        curve(f_fn(x), from = x_interval[1], to = x_interval[2], xlim = xlim,
            ylim = ylim)
        par(new = T)
    }

    x = NULL
    successCount = 0
    while (successCount < Nsim) {
        x_star = randg_fn(1)
        y_star = runif(1, max = M * g_fn(x_star))  # y* drawn from proposal function g
        # accept x* if y* <= f(x*)
        if (y_star <= f_fn(x_star)) {
            successCount = successCount + 1
            x[successCount] = x_star

            if (plot == TRUE) {
                # plot the accepted points
                points(x_star, y_star, xlab = NA, ylab = NA, xaxt = "n", yaxt = "n",
                  xlim = xlim, ylim = ylim, col = "blue")
            }
        } else {
            if (plot == TRUE) {
                # plot the rejected points
                points(x_star, y_star, xlab = NA, ylab = NA, xaxt = "n", yaxt = "n",
                  xlim = xlim, ylim = ylim, col = "red")
            }
        }
    }
    return(x)
}
```

Appendix 2: Random Permutation Test Code Listing

```r
RandomPermutationTest <- function(a, b) {
    # Combine the two datasets into a single dataset i.e., under the null
    # hypothesis, there is no difference between the two groups
    combined = c(a, b)

    # Observed difference
    t.observed = t.test(a, b)$statistic

    number_of_permutations = 1000

    t.random = NULL
    for (i in 1:number_of_permutations) {
        # Sample from the combined dataset without replacement
        shuffled = sample(combined, length(combined))

        a.random = shuffled[1:length(a)]
        b.random = shuffled[(length(a) + 1):length(combined)]

        # Null (permuated) difference
        t.random[i] = t.test(a.random, b.random)$statistic
    }

    # P-value is the fraction of how many times the permuted difference is equal
    # or more extreme than the observed difference

    pvalue = sum(abs(t.random) >= abs(t.observed))/number_of_permutations
}
```

Appendix 3: Bootstrap T Test Code Listing

```
BootstrapTTest <- function(z, y) {
    B = 1000

    boot.t = numeric(B)

    x = c(z, y)

    n = length(z)
    m = length(y)

    t.obs = t.test(z, y)$statistic

    for (i in 1:B) {
        shuffled = sample(x, n + m, replace = TRUE)

        zstar = shuffled[1:n]
        ystar = shuffled[(n + 1):(n + m)]

        boot.t[i] = t.test(zstar, ystar)$statistic
    }

    return(sum(boot.t > t.obs)/B)
}
```