6 Control Structures Homework

Data

As with Homework 5, all the numeric values you need, other than `0.05`, `0`, `1`, `2` and `3` are defined below:

```
Year=c(1936, 1946, 1951, 1963, 1975, 1997, 2006)
CaloriesPerRecipeMean <- c(2123.8, 2122.3, 2089.9, 2250.0, 2234.2, 2249.6, 3051.9)
CaloriesPerRecipeSD <- c(1050.0, 1002.3, 1009.6, 1078.6, 1089.2, 1094.8, 1496.2)
CaloriesPerServingMean <- c(268.1, 271.1, 280.9, 294.7, 285.6, 288.6, 384.4)
CaloriesPerServingSD <- c(124.8, 124.2, 116.2, 117.7, 118.3, 122.0, 168.3)
ServingsPerRecipeMean <- c(12.9, 12.9, 13.0, 12.7, 12.4, 12.4, 12.7)
ServingsPerRecipeSD <- c(13.3, 13.3, 14.5, 14.6, 14.3, 14.3, 13.0)
sample.size <- 18
tenth.increment <- 0.10
hundredth.increment <- 0.100
idx.1936 <- 1
idx.2006 <- length(CaloriesPerRecipeMean)
idxs36_07 <- c(idx.1936,idx.2006)
alpha=0.05


CookingTooMuch.dat <- data.frame(
  Year=Year,
  CaloriesPerRecipeMean = CaloriesPerRecipeMean,
  CaloriesPerRecipeSD = CaloriesPerRecipeSD,
  CaloriesPerServingMean = CaloriesPerServingMean,
  CaloriesPerServingSD = CaloriesPerServingSD,
  ServingsPerRecipeMean = ServingsPerRecipeMean,
  ServingsPerRecipeSD = ServingsPerRecipeSD
)
```

Similar restrictions from Homework 4 and 5 apply. In this homework you wil be expected to show some proficiency coding decision and iteration constructs. Use `if` statements where you might otherwise have used binary array indices, and use `for/do` loops where you otherwise have used vector operations.

Exercise 1

Recreate the table from Homework 5, Exercise 1. However, for this table, include only the unique and non-trivial (difference not equal 0) pairs. The table you create here will have the same columns as the previous exercise (`Mean1`, `Mean2`, `SD1`, `SD2`, `CohenD`). It should look something like:

Year1	Year2	Mean1	Mean2	SD1	SD2	CohenD
1946	1936	2122.3	2123.8	1002.3	1050.0	d
1951	1936	2089.9	2123.8	1009.6	1050.0	d
1963	1936	2250.0	2123.8	1078.6	1050.0	d
...	...	...	...	...	...	...
1951	1946	2089.9	2122.3	1009.6	1002.3	d
1963	1946	2250.0	2122.3	1078.6	1002.3	d
...	...	...	...	...	...	...

Year1	Year2	Mean1	Mean2	SD1	SD2	CohenD
2006	1997	3051.9	2249.6	1496.2	1094.8	d

Write two nested loops, the outer loop iterating over $i = 1, \ldots, k-1$ and the inner loop over $j = i+1, \ldots, k$. Set individual table elements by indexing `CaloriesPerRecipeMean`,`CaloriesPerRecipeSD` and `Year`.

Some suggestions: You can start with table filled with 0 or NA values, with the number of rows equal to the number of unique combinations among the treatments. You could also start with empty vectors (`Mean1 <- c()`), concatenate values (`Mean1 <- c(Mean1, current.M1)`) as you iterate over i and j, then create a table with these vectors. A third option might be to create two index vectors as you iterate over i and j and use these vectors to index means, standard deviations and year. In any case, this exercise requires the use of two nested loops.

Show your code, and print your table.

Exercise 2.

In Homework 4 and 5, we calculated sums of likelihood $L(x; \mu\sigma)$ over series of x. This provides an approximate integral, using the Newton-Cotes formula

$$\int_{x_0}^{x_n} f(x)dx \approx \sum_{j=0}^{n} w_j f(x_j)$$

with $x_j = x_0 + h \times j$, n is the number of x_j in the series, step size h and weight $w_j = h$.

We will improve this approximation by iterating over successive approximations of $f = L(x; 0, 1)$ over series of x with increasingly smaller step sizes, using your likelihood function from the previous homework.

Part a.

Let $i = 0$. Calculate L_0 by summing over $L(\mathbf{X_0})$, where X_0 is a series from $x_0 = -1, \ldots, x_n = 1$ incremented by $h = 0.1$. Multiply this sum by $w_i = h$ for an approximate $\int_{x_0}^{x_n} L(x)dx$. Use $\mu = 0$ and $\sigma = 1$ for this exercise.

Part b.

Let $i = 1$. Create a second series X_1 by setting $h_1 = h_0/2$. Compute L_1 from this series as in part a.

Part c.

Compute $\delta = |L_i - L_{i-1}|$. If $\delta < 0.0001$, your sequence of iterations has converged on a solution for $\int_{x_0}^{x_n} L(x)dx$. Finish with Part d. Otherwise, increment i, let $h_i = h_{i-1}/2$. Create the next series X_i and compute the next L_i.

Hint: code this first as a for loop of a small number of i until you know your code will converge toward a solution. I've found 10 is sufficient.

Part d

Report L_i, i, n and h. Compare your final L_i to

```
pnorm(1, lower.tail = TRUE)-pnorm(-1, lower.tail = TRUE)
```

```
## [1] 0.6826895
```

Is your L within 0.0001 of this value?

You might find it interesting to produce staircase plots for the first 2-4 iterations (plot L_i vs X_i on one graph). You might also find it interesting to plot δ or L versus i or h. You can create vectors to hold the intermediate steps. How many iterations might it take to get within 0.000001 of the expected value from R?

Exercise 3

Part a.

Write a function to compute mean, standard deviation, skewness and kurtosis from a single vector of numeric values. You can use the built-in mean function, but must use one (and only one) for loop to compute the rest. Be sure to include a check for missing values.

See https://www.itl.nist.gov/div898/handbook/eda/section3/eda35b.htm for formula for skewness and kurtosis. This reference gives several definitions for both skewness and kurtosis, you only need to implement one formula for each. Note that for computing skewness and kurtosis, standard deviation is computed using N as a divisor, not $N - 1$.

Your function should return a list with `Mean`, `SD`, `Skewness` and `Kurtosis`. If you use IML, you will need to implement this as a subroutie and use call by reference; include these variables in parameter list.

Part b.

Use this to compute skewness and kurtosis for Calories per Serving and Servings per Recipe, 1936 or 2006 from the Joy of Cooking data set. You will need to read in the file provided for lecture for this exercise, but you won't need to upload this file to D2L.

Part c.

If you wish, compare your function results with the `skewness` and `kurtosis` in the `moments` package.

Exercise 4

Write a function to compute Euclids algorithm for GCD.

Given positive integers a and b, the greatest common denominator can be found by

$$GCD_{a,b} = \begin{cases} a = b & a \\ a > b & GCD(a-b, b) \\ a < b & GCD(a, b-a) \end{cases}$$

Part a

Implement this using a `while` loop.

Part b

Implement this using recursion.

Test both implementations with $GCD(18, 48)$, $GCD(19, 48)$ and $GCD(20, 48)$

If you choose SAS for this exercise, you may need to demonstrate if IML or the macro languages supports this kind of recursion.

Exercise 5

Repeat the analysis from Exercise 5, Homework 4, but this time implement calculating MSB and MSW using iteration, not array functions.

Part a

Starting with `CaloriesPerServingMean` and `CaloriesPerServingSD`, compute the summations below using loops:

$$MSB = \frac{n_1(x_1 - \bar{x})^2 + n_2(x_2 - \bar{x})^2 + \cdots + n_k(x_k - \bar{x})^2}{k - 1} = \frac{\sum_i n_i(x_i - \bar{x})^2}{k - 1}$$

and

$$MSW = \frac{(n_1 - 1)s_1^2 + (n_2 - 1)s_2^2 + \ldots (n_k - 1)s_k^2}{N - k} = \frac{\sum_i (n_i - 1)s_i^2}{N - k}$$

Part b

Calculate F-ratio and a p for this F, using the F distribution with $k - 1$ and $N - k$ degrees of freedom. Use $\alpha = 0.05$.

Part c.

Print MSW, MSB, F and p and compare these values to the values from Homework 4.

Exercise 6

Modify your Tukey HSD function from previous Homework to accept three parameters, `s`, `n` and optional `k` and `alpha`. Check for the following conditions:

- If `s` and `n` are length 1, assume `s` is pooled standard deviation; use `k` as the number of comparisons and compute degrees of freedom as $(n - 1)k$.

- if `s` is length 1 and length `n` > 1,assume `s` is pooled standard deviation, compute a harmonic mean for `n`, and use $N - k$ as d.f. Use length `n` as the number of means, ignore the `k` parameter.
- if `s` is length > 1 and length n $= 1$, compute a pooled standard deviation and use `n` as a common sample size. Use $(n - 1)k$ for d.f., ignore the `k` parameter.
- if length `s` = length n, compute a pooled standard deviation and harmonic mean, ignore the `k` parameter.
- if both length `s` > 1 and length `n` > 1 but length `s` is not equal to length `n`, then return NA.

Test your function with the following sets of parameters:

```
tukey.hsd(128.5,18,7)
tukey.hsd(128.5,c(rep(18,7)))
tukey.hsd(128.5,c(17,18,17,18,17,18,17))
tukey.hsd(CaloriesPerServingSD,18)
tukey.hsd(CaloriesPerServingSD,c(rep(18,7)))
tukey.hsd(CaloriesPerServingSD,c(rep(18,6)))
```

The results should be nearly identical (to round error) for four of the six. Another should be slightly larger, and one should return NA.