Homework 2 Graded

Sheridan Grant

Must be uploaded to Canvas under "Homework 2 Graded" by
Tuesday, April 14 at 11:59am

Instructions

Format your code using the style shown on the course website. Any time I ask you to demonstrate something, show something, generate something, etc., you must provide the code that does so. The grader will be running your code and verifying that it solves the problems presented below.

Your code should not produce errors; **you will not receive credit for a part of a problem if your code produces an error for that part.** To check that your code doesn't produce errors before you submit, I recommend clicking the broom under the Environment tab to "clear objects from the workspace," running your entire homework file all at once, and checking over the output for errors or other unexpected behavior.

Finally, we will be giving [5pts] for code style and cleanliness. For any function you write, include a comment on the line above the function saying what the function expects as input and what it outputs. If you do this and the rest of your code is reasonably neat (blank line between parts of questions!!!) then this is an easy [5pts].

1 t-testing

For this question, you'll write a function that performs a 1-sample t-test.

(a) Write a function `tScore` that takes in two arguments—a vector of samples and the mean under the null hypothesis—and returns a t-score, i.e. the difference between the sample mean and null mean, scaled by the estimated standard deviation of the sample mean. [3pts]

(b) Now write a function `tTest1` that takes in three arguments—a vector of samples, the mean under the null hypothesis, and a character string that will either be `''greater''`, `''less''`, or `''equal''`—and returns the p-value for the 1-sample t-test. The character string represents the *null* hypothesis, i.e.

``‘‘greater’’`` means the null hypothesis is that the true mean is greater than or equal to the null mean ($H_0 : \mu \geq \mu_0$). You will need to use control flow (“if-else” statements) to make sure the function does the correct computation for each of the three character strings. [3pts]

(c) Ideally, your t-test function would provide the user with more information than just the p-value. Write a function `tTest1better` that is similar to `tTest1` except for the following:

- Takes in an extra argument, `alpha`, that is the “size” of the test: if the p-value is below this level, we reject the null hypothesis.
- Outputs a *list* (not a vector) with 3 named elements: `pVal`, the p-value; `tScore`, the t-score; and `reject`, a logical that tells the user whether or not the test rejects the null hypothesis at level `alpha`. Look up lists in R and how to name their elements; the main advantage of lists is that they can hold different data types, unlike vectors or matrices.
- Prints a message to the user that says “The null hypothesis is [rejected/not rejected] at level [alpha].” Your code should ensure that the correct phrase “rejected” or “not rejected” and the correct `alpha`-level are in the print statement. Google the `paste` function if you’ve forgotten about it.

[4pts]

2 Nested Loops

One reason loops are powerful is that they can be *nested*—one loop can go inside another.

(a) Write a function `matrixSum` that computes the sum of all elements in a matrix. You may assume the input is a matrix with numeric elements. You may *not* use the `sum` or `mean` functions—that is, you must use loops. I cannot stop you from checking that your function works by using `sum`, of course! [3pts]

(b) Write a function `triangleSum` that takes in three arguments: `X`, a matrix; `upper`, a logical; and `diagonal`, a logical. This function should compute the sum of all the elements either above the diagonal of `X` (if `upper == TRUE`) or below the diagonal of `X` (if `upper == FALSE`), either including the diagonal (if `diagonal == TRUE`) or not (if `diagonal == FALSE`). The diagonal of a matrix is all the elements with the same row/column index. Consider the following matrix:

$$A = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \\ 10 & 11 & 12 \\ 13 & 14 & 15 \end{bmatrix}$$

Its diagonal is $[1, 5, 9]$, `triangleSum(A, upper = T, diagonal = F)` should output `11`, and `triangleSum(A, upper = F, diagonal = T)` should output `109`. Note that the lower "triangle" in this case is really a trapezoid—**we still sum up everything below the diagonal**. Hints:

- If you have an "outer" loop and an "inner" loop, you can use the looping variable from the outer loop to define the vector that the inner loop loops through.
- You need *different* looping variables for the outer and inner loops. If you use `i` for both, your function won't work (make sure you understand why).
- Matrices need not be square, so for full credit make sure your function works for any-size matrices. You may write in the comment above the function that it is only intended for square matrices for [-2pts].
- This function sounds complicated, but it doesn't have to be if you break down the computation and your control flow cleverly.

[7pts]

3 Matrix Algebra

(a) A quadratic form is a polynomial of degree 2, i.e. there are only powers of zero (constants), one, and two. For example, $2x_1x_2x_3^2 + x_2^2x_3 + x_1x_3 + 4$ is a quadratic form in 3 variables. Every quadratic form in n variables can be written

$$\begin{bmatrix} 1 & x_1 & \dots & x_n \end{bmatrix} A \begin{bmatrix} 1 \\ x_1 \\ \dots \\ x_n \end{bmatrix}$$

with A a $(n+1) \times (n+1)$ square symmetric matrix.

Write a function, `quadForm`, that takes in a column vector `x` (of numerics) and a matrix `A` and outputs the quadratic form x^TAx, which is a single numeric. You will need to look up how to do matrix multiplication in R! [2pts]

(b) Suppose I tell you that $x_1 = 4$ and $x_2 = 7$, so that $x = [1, 4, 7]$. Find two different matrices, `A1` and `A2` that yield a quadratic form that evaluates to 147.

Write two lines of code defining variables `A1` and `A2` and two more lines of code using `quadForm` that show that your two matrices both work. [3pts]

(c) With matrix computations in R, you can solve linear algebra problems even if you've never taken a linear algebra class. A linear system of equations, is a set of equations that are linear in the variables $x_1, \ldots, x_n$ (you saw the $n = 2$ case early in high school, most likely). Consider the following linear system of equations:

$$\begin{aligned} x_1 + x_2 + x_3 &= 1 \\ x_1 + 2x_2 + x_3 &= 2 \\ x_1 + x_2 + 3x_3 &= 3 \end{aligned}$$

With $x = [x_1, x_2, x_3]$, write this system of equations in matrix form: $Ax = b$ (you do not have to show your work) and define, in R, a matrix `A` and vector `b` corresponding to this system of equations.

The solution to the system is then $x = A^{-1}b$, (assuming A^{-1} exists, which it does in this problem). Figure out how to compute a matrix inverse in R and compute the solution to the system of equations. Check that the solution you get actually works (you can also check by hand if you want). [5pts]