Biostatistics 615 Mastery Assignment #2 (10 pts)

Due by September 17th 2024 (Tuesday) 11:59pm. Use Gradescope (via Canvas) to submit an R file.

- Your submission should only contain one `R` file named `constrainedPolynomialRegression.R` that contains a function named `constrainedPolynomialRegression(p, y)`.
- Your code will be evaluated in Gradescope using 10 different test cases using an automated script. Full credit will be given if your code passes all test cases.
- You are allowed to submit multiple times before the deadline, but only the last submission will be graded. Automated feedback will be provided for each submission.
- You need to implement the function to work with arbitrary (valid) input values beyond the 10 cases tested. If you tweak your implementation so that your functions works specifically for the test cases, you will not receive any credit.
- You may test the function in the Google Colab page at https://bit.ly/615hw2xtra to test your code on a subset of test cases.
- Implement your function as efficient as you can. If your program does not finish after running for 2.0 seconds, you will lose the points for those test cases. Note that the official solution finishes within 0.3 seconds for any test case, so this should be a reasonable time limit.
- All the returned values should have at least 8 correct significant digits, so make sure that your implementation retains sufficient numerical precision.

Problem 1 - `constrainedPolynomialRegression.R` (10 pts)

Write an R function `constrainedPolynomialRegression.R` that contains a function `constrainedPolynomialRegression(p, y)`, which fits the following polynomial regression model with parameter constraints: for $i = 1, \ldots, n$,

$$Y_i = \beta_0 + \sum_{j=1}^{p} \beta_j X_i^j + \epsilon_i, \qquad \epsilon_i \sim \mathrm{N}(0, \sigma^2),$$

where $\beta_j = \beta_{j-1}/j$ for $j = 2, \ldots, p$, predictor $X_i = i/n$. The polynomial degree p and outcome Y_i's are all given as input parameters `p` and `y`, respectively.

Note that your `R` script file must be named as `constrainedPolynomialRegression.R`. The input arguments include a positive integer for p, and a numeric vector `y` containing a number of Y_i's. The output returns $\widehat{\beta}_k (k = 0, 1, \ldots, p)$ as a numeric vector. Your answer least an 8 digit precision. No error handling for malformed arguments is needed. Example runs are given below.

For example, when `p = 3` and `y = c(0.5, 1, 1.5)`, the four (length of $p+1$) coefficient values should be as follows (rounded to 8 digits):

```
0.21911269
0.78088731
0.39044365
0.13014788
```

The actual examples that will be used for grading will contain more complex examples, so be sure to retain maximum precision in your implementation, robustly across various types of input. At least 8 digit precision will be evaluated for correctness.

For this problem, it is important to minimize the number of arithmetic operations as much as possible to reduce relative errors. The order of arithmetic operations should be $O(p)$, not $O(p^2)$. Make sure that you are using as least number of arithmetic operations as possible.

Note that you are NOT allowed to use any functions outside the `base` package in your implementation. Use `help(...)` to check whether a function belongs to the `base` package or not.

No error handling for malformed arguments is needed.