# Biostatistics 615 Mastery Assignment #4 (10 pts)

Due by October 1st 2024 (Tuesday) 11:59pm. Use Gradescope (via Canvas) to submit an R file.

- Your submission should only contain one `R` file named `fastRidgeRegression.R` that contains a function named `fastRidgeRegression(X, Y, lambda)`.
- 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/615hw3xtra 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 3.0 seconds, you will lose the points for those test cases. Note that the official solution finishes much faster for any test case, so this should be a reasonable time limit. Also, note that the running time includes the time for reading the input files. The time reported in the Google Colab test page will be much faster because the input files are already loaded in the memory.

## Problem 1 - `fastRidgeRegression.R` (10 pts)

Consider a multiple linear regression:

$$\boldsymbol{y} = \boldsymbol{X}\boldsymbol{\beta} + \boldsymbol{\epsilon},$$

where $\boldsymbol{y}$ is an $n \times 1$ vector of response variables, $\boldsymbol{X}$ is an $n \times p$ design matrix; $\boldsymbol{\beta}$ is a $p \times 1$ vector of regression coefficients and $\boldsymbol{\epsilon}$ is an $n \times 1$ vector of random errors. We assume that $\mathrm{E}(\boldsymbol{\epsilon}) = \boldsymbol{0}_n$ and $\mathrm{Var}(\boldsymbol{\epsilon}) = \sigma^2 \mathrm{I}_n$, where $\boldsymbol{0}_n$ is an $n \times 1$ vector of zeros and $\boldsymbol{I}_n$ is an $n \times n$ identity matrix. One regularization approach to estimating $\boldsymbol{\beta}$ is the ridge regression method, which minimizes the penalized residual sum of squares:

$$(\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta})^T(\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta}) + \lambda \boldsymbol{\beta}^T\boldsymbol{\beta},$$

where $\lambda > 0$ is a pre-specified tuning parameter. The closed form of the ridge regression estimator is given by

$$\hat{\boldsymbol{\beta}} = (\boldsymbol{X}^T\boldsymbol{X} + \lambda \boldsymbol{I}_p)^{-1}\boldsymbol{X}^T\boldsymbol{y}.$$

Suppose the true regression coefficients $\boldsymbol{\beta}$ only take integer values, then rounding each element of $\hat{\boldsymbol{\beta}}$ to the nearest integer produces a better estimate.

Your task is to write a R function `fastRidgeRegression(X,Y,lambda)` inside the function `fastRidgeRegression.R`. The function computes the integer regression coefficient estimates given $\boldsymbol{X}$, $\boldsymbol{y}$ and $\lambda$.

The matrices $\boldsymbol{X}$ and $\boldsymbol{y}$ are stored in the binary files and examples provided in the Google Colab page at https://bit.ly/615hw3xtra.

The three input arguments are (1) a matrix for $\boldsymbol{X}$, (2) a matrix for $\boldsymbol{y}$, and (3) a tuning parameter $\lambda$. The function returns a data frame containing two variables `index` and `beta`. The variable `index` contains the (1-based) indices of the nonzero regression estimates. The variable `beta` contains corresponding **nonzero integer value** regression estimates for each index appears in the same row. If all the regression coefficient estimates are zero, then there will be no row in the returned data frame. Example output of running the test code is given below:

```
> X = readRDS('test.1.X.rds')
> dim(X)
[1] 100 10
> Y = readRDS('test.1.Y.rds')
> dim(Y)
[1] 100   1
> rst = fastRidgeRegression(X,Y,0.1)
> print(rst,row.names=FALSE)
 index beta
     1  -21
     2  -12
     3   12
     4   19
     5   11
     6  -14
     7  -20
     8   10
     9   10
    10  -16
> rst = fastRidgeRegression(X,Y,1000)
> print(rst,row.names=FALSE)
 index beta
     1   -1
     2   -1
     6   -1
     7   -1
    10   -1
```

The example input files "`test.1.X.rds`" and "`test.1.Y.rds`" are provided within the file `hw3_xtra_examples.tar.gz` accessible in https://bit.ly/615hw3xtra.

The actual examples that will be used for grading will contain more complex examples, so be sure to implement your function to work with arbitrary valid input matrices.

For this problem, it is important to make sure that the implementation works efficiently for arbitrary dimension of $n \times p$ matrix $\boldsymbol{X}$, regardless whether $n > p$ or $n < p$. When $n < p$, the solution based on the Cholesky decomposition may be inefficient Because $X^T X$ will be a $p \times p$ matrix. When $n << p$, you need to transform the formula to avoid the direct computation of $X^T X$, and use $XX^T$ instead.

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.