Final. Due Thursday, December 16th at midnight.

Question 1: Profiling Linear Regression

The file `final_q1.R` contains five functions that compute the statistic R^2 for a linear regression. The inputs to these functions are the design matrix for the predictors `x` (without an intercept column) and the response vector `y`. These five functions only differ in how they go about computing the sum of squared errors explained by the predictors. The first function inverts `t(x)%*%x`. The second function solves `t(x)%*%x %*% b = t(x)%*%y` for `b`. The third function computes the eigen decomposition of `t(x)%*%x` and uses this to invert `t(x)%*%x`. The fourth function computes the full singular value decomposition for `x` and uses the left singular vectors. The fifth function computes the singular value decomposition for `x` (but does not compute the right singular vectors) and uses the left singular vectors.

Your task is to profile calls to all these functions in a block. Specifically, your job is to create a 5 by 20 matrix that has as row names that are the five function names and has each column representing the total time that is spent in the each function during one call of all five functions. The times in a given column are obtained by calling each of the five functions with `x` being a 1000 by 500 matrix of standard normal variates and `y` being a length 1000 vector of standard normal variates. Each column must be obtained using a different random `x` and `y`. You must use `Rprof` with `interval=0.01` to profile the calls and extract the necessary information from the output of `summaryRprof`. You cannot tabulate this information by reading the `summaryRprof` output, your code has to extract the necessary information. At the top of your solution file, set the seed as `1234567890` so that everyone in the class uses the same `x` matrices and `y` vectors. After creating your 5 by 20 matrix profiling matrix, produce a 5 by 6 matrix whose rows are summaries (the output of the `summary` function) of the 5 rows from your profiling matrix. Which function for computing R^2 takes the least time? From the brief descriptions above, why do you think that this function is the fastest?

Question 2: Gibbs Sampling for Bayesian Laplace Regression

In the class, we did iteratively reweighted least squares (IRWLS) for computing the regression coefficient estimate for median regression. The response y_i is assumed to come from a Laplace distribution with rate τ and conditional conditional median $\mathbf{x}_i'\boldsymbol{\beta}$. The density of each y_i is $f(y_i|\mathbf{x}_i, \boldsymbol{\beta}, \tau) = 0.5\tau \exp\left(-\tau|y_i - \mathbf{x}_i'\boldsymbol{\beta}|\right)$ for $i = 1, \ldots, n$ where $\mathbf{x}_i$ is a length p predictor vector whose first element is 1 and remaining elements are the $p-1$ predictor values for observation i. We will assume that the prior distribution for $(\boldsymbol{\beta}, \tau)$ factors as $f(\boldsymbol{\beta}, \tau) = f(\boldsymbol{\beta})f(\tau)$ so that $\boldsymbol{\beta}$ and τ are independent *a priori*. The particular priors we will use are $\tau \sim \text{Exponential}(1)$ and $\boldsymbol{\beta} \sim \mathcal{N}_p(\mathbf{0}, \mathbf{I})$

Instead of producing the MLE, we will produce Gibbs samples from the posterior distribution for $(\boldsymbol{\beta}, \tau)$, which is given (up to proportion) by

$$f(\boldsymbol{\beta}, \tau|\mathbf{y}, \mathbf{X}) \propto f(\boldsymbol{\beta})f(\tau)\prod_{i=1}^{n} f(y_i|\mathbf{x}_i, \boldsymbol{\beta}, \tau) = \frac{1}{(2\pi)^{\frac{p}{2}}}\exp(-0.5\boldsymbol{\beta}'\boldsymbol{\beta}) \times \exp(-\tau) \times \frac{\tau^n}{2^n}\exp\left(-\tau\sum_{i=1}^{n}|y_i - \mathbf{x}_i'\boldsymbol{\beta}|\right)$$

where $\mathbf{X}$ is a matrix whose i-th row is $\mathbf{x}_i'$.

The full conditional for τ is easy to determine; $\tau|\boldsymbol{\beta}, \mathbf{y}, \mathbf{X} \sim \text{Gamma}\left(1 + n, 1 + \sum_{i=1}^{n}|y_i - \mathbf{x}_i'\boldsymbol{\beta}|\right)$ and so sampling τ from its full conditional is easy.

Unfortunately, the full conditional for $\boldsymbol{\beta}$ is difficult to sample from directly. We will introduce the same parameter expansion that we did for IRWLS. Let z_i be a random variable whose conditional distribution is given by

$$f(z_i|y_i, \mathbf{x}_i, \boldsymbol{\beta}, \tau) = \sqrt{\frac{1}{2\pi z_i^3}}\exp\left(-\frac{\tau^2(y_i - \mathbf{x}_i'\boldsymbol{\beta})^2}{2z_i}\left(z_i - \frac{1}{\tau|y_i - \mathbf{x}_i'\boldsymbol{\beta}|}\right)^2\right)$$

which is an Inverse-Gaussian density with mean $\nu = \frac{1}{\tau|y_i - \mathbf{x}_i'\boldsymbol{\beta}|}$ and shape $\lambda = 1$. This can be sampled from by using the `rinvGauss` function in the `SuppDists` package. The parameter inputs to this vectorized function are a vector of means `nu` and a vector of shapes `lambda`.

Using this parameter expansion, we get a multivariate normal full conditional for $\boldsymbol{\beta}$

$$\boldsymbol{\beta}|\mathbf{y}, \mathbf{X}, \mathbf{z}, \tau \sim \mathcal{N}_p\left(\left(\tau^2\mathbf{X}'\mathbf{Z}\mathbf{X} + \mathbf{I}\right)^{-1}\tau^2\mathbf{X}'\mathbf{Z}\mathbf{y}, \left(\tau^2\mathbf{X}'\mathbf{Z}\mathbf{X} + \mathbf{I}\right)^{-1}\right)$$

where $\mathbf{Z}$ is a diagonal matrix whose diagonal is formed by the $\mathbf{z}$ values. This can be sampled from using the `rmvnorm` function from the `mvtnorm` package. The parameter inputs to the function are a mean vector `mean` and variance-covariance matrix `sigma`.

In the archive `final_q2_files.zip` there is a main and headers directory structure as well as a `.RData` file. I have written a main function for the Gibbs sampling as well as an update header function that draws a new state given the current state. The update function calls a header function for drawing the next τ state given the current state. I have written this function. The update function also calls a header function for drawing the next $\boldsymbol{\beta}$ state given the current state. I have not written this function. It is your job to write this function. It should take in two inputs `data` (which is a list of the data `x` and `y`) and `state` (which is a list of the current `beta` and `tau`). The first thing that the function does is draw a vector `z` from the appropriate inverse Gaussian distributions for the parameter expansion. Then it should draw a `beta` vector from the appropriate multivariate normal and return this `beta` draw.

After writing your function, use the main function to get 10000 draws from the posterior of a Laplace regression using the data `q2_x` and `q2_y` that can be loaded from `final_q2_data.RData`. What are the posterior means of the $\boldsymbol{\beta}$ draws and the τ draws? Use the draws to make univariate 95 intervals for each element of $\boldsymbol{\beta}$. Which elements of $\boldsymbol{\beta}$ could you confidently say are non-zero in this Laplace regression?

Question 3: Givens Rotations for Eigen Decompostion

The method of doing Givens rotations for computing the eigen decomposition for a symmetric matrix X involves iteratively looping through the upper off-diagonal elements of the matrix X, determining if one is too large, and doing a Givens rotation if it is. X is replaced with the appropriately rotated X and the process continues until all off diagonal elements are small enough.

At each step, a matrix R for the rotation is determined and X is replaced by RXR'. The eigen-vector matrix U can be computed as the algorithm runs by starting with $U = I$ at the first step and updating U at subsequent steps by replacing it with UR.

When considering the (i, j) (where $i < j$) off diagonal element of X, the rotation matrix R is given by the following values. $R_{k,k} = 1$ for $k \neq i, j$, $R_{k,\ell} = 0$ for $k \neq \ell$ and $k, \ell \notin \{i, j\}$. This sets all of the values for R as they would be in the identity matrix except for the (i, i), (j, j), (i, j), and (j, i) elements. Let X_{local} and R_{local} be the matrices

$$X_{\text{local}} = \begin{pmatrix} X_{i,i} & X_{i,j} \\ X_{j,i} & X_{j,j} \end{pmatrix} \quad \text{and} \quad R_{\text{local}} = \begin{pmatrix} R_{i,i} & R_{i,j} \\ R_{j,i} & R_{j,j} \end{pmatrix} = \begin{pmatrix} c & s \\ -s & c \end{pmatrix}$$

where $s^2 + c^2 = 1$. The multiplication RXR' is to result in a matrix A whose off diagonal elements are 0.

$$\begin{aligned} A &= R_{\text{local}} X_{\text{local}} R'_{\text{local}} \\ &= \begin{pmatrix} c & s \\ -s & c \end{pmatrix} \begin{pmatrix} X_{i,i} & X_{i,j} \\ X_{j,i} & X_{j,j} \end{pmatrix} \begin{pmatrix} c & -s \\ s & c \end{pmatrix} \\ &= \begin{pmatrix} cX_{i,i} + sX_{j,i} & cX_{i,j} + sX_{j,j} \\ -sX_{i,i} + cX_{j,i} & -sX_{i,j} + cX_{j,j} \end{pmatrix} \begin{pmatrix} c & -s \\ s & c \end{pmatrix} \\ &= \begin{pmatrix} c^2X_{i,i} + scX_{j,i} + scX_{i,j} + s^2X_{j,j} & -scX_{i,i} - s^2X_{j,i} + c^2X_{i,j} + scX_{j,j} \\ -scX_{i,i} + c^2X_{j,i} - s^2X_{i,j} + scX_{j,j} & -s^2X_{i,i} - scX_{j,i} - scX_{i,j} + c^2X_{j,j} \end{pmatrix} \\ &= \begin{pmatrix} A_{1,1} & 0 \\ 0 & A_{2,2} \end{pmatrix} \end{aligned}$$

Using the fact that X is symmetric and so $X_{i,j} = X_{j,i}$, we get identical off diagonal equations of

$$sc(X_{i,i} - X_{j,j}) = (c^2 - s^2)X_{j,i} \quad \text{or} \quad \frac{2sc}{(c^2 - s^2)} = \frac{2X_{j,i}}{X_{i,i} - X_{j,j}}$$

Setting $s = \sin(\theta)$ and $c = \cos(\theta)$ and using double angle formulas, we get

$$\tan(2\theta) = \frac{2X_{j,i}}{X_{i,i} - X_{j,j}}$$

If $X_{i,i} = X_{j,j}$ (or up to machine precision), then we should set $\theta = \pi/4$. Otherwise, we should set $\theta = \frac{1}{2}\tan^{-1}\left(\frac{2X_{j,i}}{X_{i,i} - X_{j,j}}\right)$. This then provides the definition for R_{local} and thus R.

The file `final_q3.R` contains the skeleton of a function that is to compute the eigen decomposition of an input matrix X. It is your job to complete this function. Write tests for this function that determines whether its output is the same (up to some reasonable error, do not forget that the eigen vectors can be negated and return the same eigen decomposition) as the output from `eigen` for random 2 by 2, 3 by 3, 4 by 4, and 5 by 5 symmetric matrices generated, for example, by the code `z = matrix(rnorm(d^d),d,d); X = z + t(z)` where `d` is the desired dimension. Before generating each matrix you would use for testing, set your seed at `1234567890`.