Homework 3

1. In this problem, you will consider choosing the tuning parameters for both ridge regression and the lasso, using 10-fold cross-validation. First download the files "main.R", "plotfuns.R", and "bstar.Rdata". The first line of the file "main.R" has you install the package glmnet. Once you have done this (i.e., once you have installed this package), you can comment this line out.

   We begin with a true signal bstar. Although this is stored as a vector of length $p = 2500$, bstar really represents an image of dimension 50×50. You can plot it by calling

   ```
   plot.image(bstar).
   ```

   This image is truly sparse, in the sense that 2084 of its pixels have a value of 0, while 416 pixels have a value of 1. You can think of this image as a toy version of an MRI image that we are interested in collecting.

   Suppose that, because of the nature of the machine that collects the MRI image, it takes a long time to measure each pixel value individually, but it's faster to measure a linear combination of pixel values. We measure $n = 1300$ linear combinations, with the weights in the linear combination being random, in fact, independently distributed as $N(0, 1)$. These measurements are given by the entries of the vector

   ```
   X %*% bstar
   ```

   in our R code. Because the machine is not perfect, we don't get to observe this directly, but we see a noisy version of this. Hence, in terms of our R code, we observe

   ```
   y = X %*% bstar + rnorm(n,sd=5).
   ```

   Now the question is: can we model y as a linear combination of the columns of X to recover some coefficient vector that is close to bstar? Roughly speaking, the answer is yes. Key points here: although the number of measurements $n = 1300$ is smaller than the dimension $p = 2500$, the true vector bstar is sparse, and the weights in a linear combination are i.i.d normal. This is the idea behind the field of *compressed sensing*. Below, you can find several clips regarding the history, motivation and applications of compressed sensing:

   - Robust Compressed Sensing: How Undersampling Introduces Noise and What We Can Do About It (minutes 2-16). `https://www.youtube.com/watch?v=ThiAk_n-8HI`
   - Compressed Sensing: Recovery, Algorithms, and Analysis (first 4 minutes). `https://www.youtube.com/watch?v=mgCIKnMgBmk`
   - Compressive Sensing (minutes 5-16). `https://www.youtube.com/watch?v=RvMgVv-xZhQ`

   The file "main.R" is setup to perform ridge regression of y on X, and the lasso of y on X, with the tuning parameter for each method selected by cross-validation. You will fill in the missing pieces. It's helpful to read through the whole file to get a sense of what's to be accomplished. Try to understand all the parts, even if it doesn't seem related to what you have to fill in; this should be good practice for working with R in the future, etc.

a. Fill in the missing parts. There are 2 missing parts marked by # TODO. When you're getting started, it might be helpful to read the documentation for the glmnet function, which you will use to perform ridge regression and the lasso.

b. Plot the cross-validation error curves for each of ridge regression and the lasso. You can do this using the function plot.cv, as demonstrated by the code at the end. For both ridge regression and the lasso, what value of λ is chosen by the usual rule? What value is chosen by the one standard error rule? Which method, ridge regression or the lasso, has a smaller minimum cross-validation error?

c. Now run ridge regression and the lasso on the entire data set X,y, for the same tuning parameter values as you did before. Save the objects returned by glmnet as a.rid, a.las, respectively. Plot the coefficient images corresponding to the values of λ chosen by 10-fold CV, for each of ridge regression and the lasso. For this, you'll want to use the indices that you computed in parts (a) and (b), i1.rid, i1.las (usual rule), as well as the coefficients a.rid$beta, a.las$beta that you just computed. Which image looks better? What is the difference between the ridge regression images and the lasso images? Which do you think matches the true image bstar more closely?

d. Look at the squared error between the ridge regression and the lasso coefficients that you computed in (c) and the true coefficient vector bstar. What has the lowest squared error?

e. Look at the absolute error between the ridge regression and the lasso coefficients that you computed in (c) and the true coefficient vector bstar. What has the lowest absolute error?