Problem Set 6, Winter 2024

Michael Ghattas

START:

```
knitr::opts_chunk$set(echo = TRUE)

# Setup Environment
library(tidyverse)
```

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.4     v readr     2.1.5
## v forcats   1.0.0     v stringr   1.5.1
## v ggplot2   3.5.1     v tibble    3.2.1
## v lubridate 1.9.3     v tidyr     1.3.1
## v purrr     1.0.2
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors
```

```
library(mlbench)
library(glmnet)
```

```
## Loading required package: Matrix
##
## Attaching package: 'Matrix'
##
## The following objects are masked from 'package:tidyr':
##
##     expand, pack, unpack
##
## Loaded glmnet 4.1-8
```

CONTEXT: Pew Research Center data The data in "pew_data.RData" comes from the Pew Research Center, an organization that conducts nationally-representative public opinion polls on a variety of political and social topics. Dr. Durso constructed this data set from the 2017 Pew Research Center Science and NewsSurvey, downloaded from https://www.journalism.org/datasets/2018/ on 4/16/2019.

There are 224 variables in this data set, but only a subset will be used in this problem set. For this problem set, the outcome of interest will be the LIFE variable, which was presented to respondents like so: "In general, would you say life in America today is better, worse or about the same as it was 50 years ago for people like you?"

Possible responses included: 1 = Better today 2 = Worse today 3 = About the same as it was 50 years ago -1 = Refused

Question 1 - Data processing - 5 points

You will use the Pew data set again for these questions, but you will not conduct a full complete cases analysis this time. Instead, you'll conduct a "complete outcomes" analysis, where only observations containing a valid response to the outcome variable are included. The data for this question will be stored in a new data set called "pew2". You will need to have your directory set to where the data set is on your computer, so be sure to do that before running the code chunk below.

```
load("pew_data.RData")
# Select only relevant variables for analysis
pew2 <- dplyr::select(dat, PPINCIMP, PPGENDER, PPETHM, IDEO, PPEDUCAT, LIFE)
```

You will again binarize the LIFE outcome such that a value of 1 means that the participant responded that life in America is worse today than 50 years ago (LIFE=2) and a value of 0 contains all other resposnes (LIFE=1 or LIFE=3). Check to see how people respodned to the LIFE variable before you recode this and answer the question below the code chunk.

```
# Display LIFE variable's attributes and response frequencies
attributes(pew2$LIFE)$label
```

```
## [1] "In general, would you say life in America today is better, worse or about the same as it was 50
```

```
attributes(pew2$LIFE)$labels
```

```
##                                Refused                             Better today
##                                     -1                                        1
##                            Worse today About the same as it was 50 years ago
##                                      2                                        3
```

```
table(pew2$LIFE, exclude = NULL)
```

```
##
##   -1    1    2    3
##   18 1596 1900  510
```

A) How many people's response was "Refused", "Not asked", or "NA" for the LIFE variable? Your answer here: 18. This corresponds to the -1 value in the table output, which represents the "Refused" responses.

Use the code chunk below to drop any observations with a response of "Refused", "Not asked", or "NA" for the LIFE variable. Save the resulting data set as pew.life and answer the question below the code chunk

```
# Drop observations where LIFE is -1 (Refused) or NA
pew.life <- pew2 %>% filter(LIFE != -1 & !is.na(LIFE))

# Count the rows in the complete outcomes data set
nrow(pew.life)
```

```
## [1] 4006
```

B) How many rows remain in your data set once you've dropped all observations where the response to the LIFE variable was "Refused", "Not asked", or NA? Your answer here: 4006. This is the number of rows remaining in pew.life after filtering out "Refused" and NA responses in the LIFE variable.

Re-code the LIFE variable such that "Worse today" is equal to one and "Better today"/"About the same" are equal to 0. Be sure to display the frequencies of the recoded variable.

```
# Re-code LIFE variable: 1 for "Worse today", 0 for "Better today" and "About the same"
pew.life$worse <- ifelse(pew.life$LIFE == 2, 1, 0)

# Display the frequencies of the recoded outcome
table(pew.life$worse, exclude = NULL)
```

```
##
##    0    1
## 2106 1900
```

C) Per the table of your recoded variable ("worse"), does the number of 1s in this variable match the number of people who responded "worse today" in the original LIFE variable? (Hint: if no, check in with me about it): Your answer here: Yes, the number of 1s in the recoded "worse" variable is 1900, which matches the count of people who responded "Worse today" (LIFE = 2) in the original LIFE variable.

Finally, check that all six variables are of the appropriate type. Income should be a numeric- or integer-type variable, and gender, ethnicity, ideology, education categoryshould be factor-type variables. *We will treat the "worse" variable differently this time; rather than a factor variable, we will treat it as a numeric variable.* The reason for this is because we will be computing validation-set and test-set model deviances in a later question, and the function to do that requires a numeric 0/1 outcome. Check that you've done this correctly by using the str() function.

```
# Convert variables to the correct types
pew.life$income <- as.numeric(pew.life$PPINCIMP)      # Income as numeric
pew.life$gender <- as.factor(pew.life$PPGENDER)       # Gender as factor
pew.life$eth <- as.factor(pew.life$PPETHM)            # Ethnicity as factor
pew.life$ideo <- as.factor(pew.life$IDEO)             # Ideology as factor
pew.life$edu <- as.factor(pew.life$PPEDUCAT)          # Education as factor
pew.life$worse <- as.numeric(pew.life$worse)          # Outcome as numeric

# Display the structure to confirm types
str(pew.life)
```

```
## tibble [4,006 x 12] (S3: tbl_df/tbl/data.frame)
##  $ PPINCIMP: 'labelled' num [1:4006] 16 19 12 12 21 18 19 16 7 10 ...
##   ..- attr(*, "label")= chr "Household Income"
##   ..- attr(*, "format.spss")= chr "F2.0"
##   ..- attr(*, "labels")= Named num [1:23] -2 -1 1 2 3 4 5 6 7 8 ...
##   .. ..- attr(*, "names")= chr [1:23] "Not asked" "REFUSED" "Less than $5,000" "$5,000 to $7,499" ..
##  $ PPGENDER: 'labelled' num [1:4006] 1 2 1 1 1 1 2 2 2 2 ...
##   ..- attr(*, "label")= chr "Gender"
##   ..- attr(*, "format.spss")= chr "F2.0"
##   ..- attr(*, "labels")= Named num [1:4] -2 -1 1 2
##   .. ..- attr(*, "names")= chr [1:4] "Not asked" "REFUSED" "Male" "Female"
```

```
##  $ PPETHM  : 'labelled' num [1:4006] 1 2 4 4 1 5 1 5 1 1 ...
##   ..- attr(*, "label")= chr "Race / Ethnicity"
##   ..- attr(*, "format.spss")= chr "F2.0"
##   ..- attr(*, "labels")= Named num [1:7] -2 -1 1 2 3 4 5
##   .. ..- attr(*, "names")= chr [1:7] "Not asked" "REFUSED" "White, Non-Hispanic" "Black, Non-Hispanic
##  $ IDEO    : 'labelled' num [1:4006] 1 3 2 3 2 3 2 3 3 2 ...
##   ..- attr(*, "label")= chr "In general, would you describe your political views as..."
##   ..- attr(*, "format.spss")= chr "F4.0"
##   ..- attr(*, "labels")= Named num [1:6] -1 1 2 3 4 5
##   .. ..- attr(*, "names")= chr [1:6] "Refused" "Very conservative" "Conservative" "Moderate" ...
##  $ PPEDUCAT: 'labelled' num [1:4006] 4 4 2 1 3 3 4 4 2 4 ...
##   ..- attr(*, "label")= chr "Education (Categorical)"
##   ..- attr(*, "format.spss")= chr "F2.0"
##   ..- attr(*, "labels")= Named num [1:6] -2 -1 1 2 3 4
##   .. ..- attr(*, "names")= chr [1:6] "Not asked" "REFUSED" "Less than high school" "High school" ...
##  $ LIFE    : 'labelled' num [1:4006] 2 2 1 2 2 1 2 1 2 2 ...
##   ..- attr(*, "label")= chr "In general, would you say life in America today is better, worse or abou
##   ..- attr(*, "format.spss")= chr "F4.0"
##   ..- attr(*, "labels")= Named num [1:4] -1 1 2 3
##   .. ..- attr(*, "names")= chr [1:4] "Refused" "Better today" "Worse today" "About the same as it was
##  $ worse   : num [1:4006] 1 1 0 1 1 0 1 0 1 1 ...
##  $ income  : num [1:4006] 16 19 12 12 21 18 19 16 7 10 ...
##  $ gender  : Factor w/ 2 levels "1","2": 1 2 1 1 1 1 2 2 2 2 ...
##  $ eth     : Factor w/ 5 levels "1","2","3","4",..: 1 2 4 4 1 5 1 5 1 1 ...
##  $ ideo    : Factor w/ 6 levels "-1","1","2","3",..: 2 4 3 4 3 4 3 4 4 3 ...
##  $ edu     : Factor w/ 4 levels "1","2","3","4": 4 4 2 1 3 3 4 4 2 4 ...
```

Question 2 - Splitting the data into training, validation, and test tests - 5 points

We will use the train-validate-test procedure to assess model generalizability. The first step of the train-validate-test process is to split the data into training, validation, and test sets. To make this easier, first create a new data set that contains only the variables that will be used in the analysis:

worse income
gender
eth
ideo
edu

```r
# Create a new data set containing only the specified variables
pew.life2 <- pew.life %>% select(worse, income, gender, eth, ideo, edu)

# Display the structure of the new data set to confirm variable types
str(pew.life2)
```

```
## tibble [4,006 x 6] (S3: tbl_df/tbl/data.frame)
##  $ worse : num [1:4006] 1 1 0 1 1 0 1 0 1 1 ...
##  $ income: num [1:4006] 16 19 12 12 21 18 19 16 7 10 ...
##  $ gender: Factor w/ 2 levels "1","2": 1 2 1 1 1 1 2 2 2 2 ...
##  $ eth   : Factor w/ 5 levels "1","2","3","4",..: 1 2 4 4 1 5 1 5 1 1 ...
##  $ ideo  : Factor w/ 6 levels "-1","1","2","3",..: 2 4 3 4 3 4 3 4 4 3 ...
##  $ edu   : Factor w/ 4 levels "1","2","3","4": 4 4 2 1 3 3 4 4 2 4 ...
```

Saving the number of rows in this new data set will be useful, so run the following code chunk to do so.

```
# Save the number of rows in pew.life2 for further use
n <- nrow(pew.life2)
n
```

```
## [1] 4006
```

In the async material, the following line of code was provided to help create the split: tvt2 <- sample(rep(0:2,c(round(n.2),round(n.2),n-2round(n.2))),n)

To help you understand what's going on here before you use it, have a look at what's produced by what's in the inner rep() function by running the code chunk below.

```
Sixty.twenty.twenty <- rep(0:2,c(round(n*.2),round(n*.2),n-2*round(n*.2)))
table(Sixty.twenty.twenty)
```

```
## Sixty.twenty.twenty
##    0    1    2
##  801  801 2404
```

```
Seventy.fifteen.fifteen <- rep(0:2,c(round(n*.15),round(n*.15),n-2*round(n*.15)))
table(Seventy.fifteen.fifteen)
```

```
## Seventy.fifteen.fifteen
##    0    1    2
##  601  601 2804
```

```
Eighty.ten.ten <- rep(0:2,c(round(n*.10),round(n*.10),n-2*round(n*.10)))
table(Eighty.ten.ten)
```

```
## Eighty.ten.ten
##    0    1    2
##  401  401 3204
```

```
Ninety.five.five <- rep(0:2,c(round(n*.05),round(n*.05),n-2*round(n*.05)))
table(Ninety.five.five)
```

```
## Ninety.five.five
##    0    1    2
##  200  200 3606
```

A) Which value/s in these tables (0, 1, or 2) correspond to the portion of sample that will be assigned to the training set? Your answer here: The value 2 corresponds to the portion of the sample assigned to the training set in each scenario. In tvt2, a value of 2 represents the majority of the sample allocated to the training set.

B) Which value/s in these tables (0, 1, or 2) correspond to the portion of sample that will be assigned to the validation and test sets, respectively? Your answer here: The values 1 and 0 correspond to the validation and test sets, respectively.

Split your data set into training, validation, and test sets. Use the following proportions: 70% training, 15% validation, and 15% test. When splitting data into training/validation/test data sets, it's good practice to set a random seed to create a split that's reproducible (i.e., recoverable later). For this question, use the seed provided. To ensure that your answers match, be sure to run the set.seed() line *immediately* before your completed tvt2 line.

```
# Set seed for reproducibility
set.seed(202205)

# Generate the 70-15-15 split for training, validation, and test sets
tvt2 <- sample(rep(0:2, c(round(n * 0.15), round(n * 0.15), n - 2 * round(n * 0.15))), n)

# Split the data based on tvt2 values
dat.train <- pew.life2[tvt2 == 2, ]
dat.valid <- pew.life2[tvt2 == 1, ]
dat.test <- pew.life2[tvt2 == 0, ]

# Display the row counts for each set
nrow(dat.train)
```

```
## [1] 2804
```

```
nrow(dat.valid)
```

```
## [1] 601
```

```
nrow(dat.test)
```

```
## [1] 601
```

C) How many rows are in the dat.train data set? Your answer here: 2804. This corresponds to the number of rows in the training set (dat.train) after using the 70-15-15 split.

D) How many rows are in the dat.valid data set? Your answer here: 601. This is the row count for the validation set (dat.valid) after splitting.

E) How many rows are in the dat.test data set? Your answer here: 601. This is the row count for the test set (dat.test) after splitting.

Question 3 - Fitting candidate models to the training data set and saving the results - 5 points

For this problem set, you'll assess the generalizability of the three models you fitted in Problem Set 5. Here are the four models you will test, all of which will use "worse" as the outcome: Model 1: Include income as a continuous predictor and gender as a categorical predictor. Model 2: In addition to the predictors in Model 1, include ethnicity and education as categorical predictors. Model 3: In addition to the predictors in Model 2, include the ideology variable.

The second step of the train-validate-test process is to "train" your models on the training set - that is, you will fit the three logistic regression models to the training set data to generate coefficient estimates for the predictors in each model. Fit these models in the code chunk below and save the model objects as model.1, model.2, and model.3 respectively.

```r
# Model 1: Include income and gender as predictors
model.1 <- glm(worse ~ income + gender, data = dat.train, family = 'binomial')
summary(model.1)
```

```
##
## Call:
## glm(formula = worse ~ income + gender, family = "binomial", data = dat.train)
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept)  0.431704   0.124185   3.476 0.000508 ***
## income      -0.051657   0.008419  -6.136 8.46e-10 ***
## gender2      0.237053   0.076569   3.096 0.001962 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 3877.6  on 2803  degrees of freedom
## Residual deviance: 3826.2  on 2801  degrees of freedom
## AIC: 3832.2
##
## Number of Fisher Scoring iterations: 4
```

```r
# Model 2: Add ethnicity (eth) and education (edu) as additional predictors
model.2 <- glm(worse ~ income + gender + eth + edu, data = dat.train, family = 'binomial')
summary(model.2)
```

```
##
## Call:
## glm(formula = worse ~ income + gender + eth + edu, family = "binomial",
##     data = dat.train)
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept)  0.433709   0.178042   2.436 0.014851 *
## income      -0.041280   0.009457  -4.365 1.27e-05 ***
## gender2      0.225011   0.077463   2.905 0.003675 **
## eth2        -0.456267   0.133670  -3.413 0.000642 ***
## eth3        -0.095325   0.203297  -0.469 0.639144
## eth4        -0.333801   0.130522  -2.557 0.010545 *
## eth5         0.031193   0.197778   0.158 0.874680
## edu2         0.069650   0.157547   0.442 0.658424
## edu3         0.253503   0.161011   1.574 0.115383
## edu4        -0.400116   0.165150  -2.423 0.015404 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 3877.6  on 2803  degrees of freedom
## Residual deviance: 3766.2  on 2794  degrees of freedom
## AIC: 3786.2
```

```
##
## Number of Fisher Scoring iterations: 4
```

```
# Model 3: Add ideology (ideo) as an additional predictor
model.3 <- glm(worse ~ income + gender + eth + edu + ideo, data = dat.train, family = 'binomial')
summary(model.3)
```

```
##
## Call:
## glm(formula = worse ~ income + gender + eth + edu + ideo, family = "binomial",
##     data = dat.train)
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept)  0.858361   0.293828   2.921  0.00349 **
## income      -0.041235   0.009498  -4.342 1.41e-05 ***
## gender2      0.245727   0.078142   3.145  0.00166 **
## eth2        -0.436706   0.134751  -3.241  0.00119 **
## eth3        -0.075245   0.204591  -0.368  0.71304
## eth4        -0.340541   0.131290  -2.594  0.00949 **
## eth5         0.043179   0.198686   0.217  0.82796
## edu2         0.073525   0.158246   0.465  0.64220
## edu3         0.279810   0.161946   1.728  0.08402 .
## edu4        -0.352346   0.166335  -2.118  0.03415 *
## ideo1       -0.345988   0.281175  -1.231  0.21851
## ideo2       -0.457487   0.257062  -1.780  0.07513 .
## ideo3       -0.426671   0.252744  -1.688  0.09138 .
## ideo4       -0.755395   0.266727  -2.832  0.00462 **
## ideo5       -0.380311   0.289505  -1.314  0.18896
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 3877.6  on 2803  degrees of freedom
## Residual deviance: 3753.1  on 2789  degrees of freedom
## AIC: 3783.1
##
## Number of Fisher Scoring iterations: 4
```

Question 4 - Computing validation deviances and choosing a model to advance - 5 points

The third step of this process is to take the models estimated using the training set data, run the validation set data through those models to obtain new predictions, then evaluate the result using a predetermined criterion. In this case, we'll use model deviance as the criterion. There is a useful function for this provided in the async material (5.2.1: backward_train_validate_test_5_2_1), which I've copied below for convenience:

valid.dev<-function(m.pred, dat.this){ pred.m<-predict(m.pred,dat.this, type="response") -2$sum(dat.thischd * log(pred.m) + (1 - dat.thischd)$log(1-pred.m)) }

Start by adapting this code to work for the current data set

```r
# Modify the valid.dev function for this data set using "worse" as the binary outcome variable
valid.dev <- function(m.pred, dat.this) {
  pred.m <- predict(m.pred, dat.this, type = "response")  # Generate predicted probabilities
  -2 * sum(dat.this$worse * log(pred.m) + (1 - dat.this$worse) * log(1 - pred.m))  # Compute model devi
}
```

Next, apply this function to your three candidate models to obtain the validation deviances. After doing so, answer the questions below.

```r
# Deviance for Model 1
dev.1 <- valid.dev(model.1, dat.valid)
dev.1
```

```
## [1] 815.5384
```

```r
# Deviance for Model 2
dev.2 <- valid.dev(model.2, dat.valid)
dev.2
```

```
## [1] 808.0623
```

```r
# Deviance for Model 3
dev.3 <- valid.dev(model.3, dat.valid)
dev.3
```

```
## [1] 799.8163
```

A) What is the validation deviance of Model 1? Your answer here: 815.5384

B) What is the validation deviance of Model 2? Your answer here: 808.0623

C) What is the validation deviance of Model 3? Your answer here: 799.8163

D) Based on the validation deviances you computed, which model do you choose based on the results you obtained? Your answer here: Model 3. It has the lowest validation deviance (799.8163), indicating the best fit on the validation set among the three models.

Question 5 - Evaluating the model chosen in the validation step using the test set data - 5 points

Now that you've chosen a candidate model based on its performance on the validation data set, you'll now do the final step in the process: compute the deviance of this model when applied to the test data set. Use the adapted deviance function to compute the deviances of the chosen model when applied to the test set.

```r
# Compute the deviance of the chosen model (Model 3) on the test set
test.dev <- valid.dev(model.3, dat.test)
test.dev
```

```
## [1] 812.8986
```

A) What is the deviance of the chosen model when applied to the test set? Your answer here: The test set deviance of Model 3, our chosen model, is 812.8986.

CONTEXT - HOUSE VALUES IN BOSTON, CIRCA 1970 This dataset was obtained through the mlbench package, which contains a subset of data sets available through the UCI Machine Learning Repository. From the help file: Housing data for 506 census tracts of Boston from the 1970 census. The dataframe BostonHousing contains the original data by Harrison and Rubinfeld (1979). The original data are 506 observations on 14 variables.

Continuous variables: crim per capita crime rate by town zn proportion of residential land zoned for lots over 25,000 sq.ft
indus proportion of non-retail business acres per town nox nitric oxides concentration (parts per 10 million) rm average number of rooms per dwelling age proportion of owner-occupied units built prior to 1940 dis weighted distances to five Boston employment centres rad index of accessibility to radial highways tax full-value property-tax rate per USD 10,000 ptratio pupil-teacher ratio by town b 1000(B - 0.63)^2 where B is the proportion of blacks by town lstat percentage of lower status of the population medv median value of owner-occupied homes in USD 1000's

Categorical variables: chas Charles River dummy variable (= 1 if tract bounds river; 0 otherwise)

Question 6 - Cross-validated ridge regression - 10 points

The BostonHousing data is contained inside of an R package, so you'll load the data into memory a little differently than usual. Run the following code chunk, confirm that the data in loaded into memory, and ensure that your variables are of the proper type (they should be)

```
# Load the BostonHousing dataset from the mlbench package
data(BostonHousing)

# Display the structure of the dataset to confirm variable types
str(BostonHousing)
```

```
## 'data.frame':    506 obs. of  14 variables:
##  $ crim   : num  0.00632 0.02731 0.02729 0.03237 0.06905 ...
##  $ zn     : num  18 0 0 0 0 0 12.5 12.5 12.5 12.5 ...
##  $ indus  : num  2.31 7.07 7.07 2.18 2.18 2.18 7.87 7.87 7.87 7.87 ...
##  $ chas   : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1 ...
##  $ nox    : num  0.538 0.469 0.469 0.458 0.458 0.458 0.524 0.524 0.524 0.524 ...
##  $ rm     : num  6.58 6.42 7.18 7 7.15 ...
##  $ age    : num  65.2 78.9 61.1 45.8 54.2 58.7 66.6 96.1 100 85.9 ...
##  $ dis    : num  4.09 4.97 4.97 6.06 6.06 ...
##  $ rad    : num  1 2 2 3 3 3 5 5 5 5 ...
##  $ tax    : num  296 242 242 222 222 222 311 311 311 311 ...
##  $ ptratio: num  15.3 17.8 17.8 18.7 18.7 18.7 15.2 15.2 15.2 15.2 ...
##  $ b      : num  397 397 393 395 397 ...
##  $ lstat  : num  4.98 9.14 4.03 2.94 5.33 ...
##  $ medv   : num  24 21.6 34.7 33.4 36.2 28.7 22.9 27.1 16.5 18.9 ...
```

For this question, conduct a cross-validated ridge regression. Use medv as the outcome and all of the other variables in the data set as the predictors. *Do NOT split your data into training and test sets for this question;* conduct the analysis on the whole data set. First, conduct the cross-validated ridge regression. Be sure to use the set.seed() provided to make your analysis reproducible.

```r
set.seed(202211)

# Set up the design matrix (predictors) for ridge regression, omitting the intercept
X <- model.matrix(medv ~ ., data = BostonHousing)[,-1]
# Set up the response vector (outcome)
y <- BostonHousing$medv

# Conduct cross-validated ridge regression (alpha = 0 for ridge regression)
cvfit.house.ridge <- cv.glmnet(x = X, y = y, alpha = 0)
```

Next, display the value for lambda.min *and* the coefficients associated with it. Make sure these are visible in your knitted document.

```r
# Display lambda that minimizes cross-validation error
cvfit.house.ridge$lambda.min
```

```
## [1] 0.6777654
```

```r
# Display the coefficients associated with lambda.min
coef(cvfit.house.ridge, s = "lambda.min")
```

```
## 14 x 1 sparse Matrix of class "dgCMatrix"
##                        s1
## (Intercept)  28.001475824
## crim         -0.087572712
## zn            0.032681030
## indus        -0.038003639
## chas1         2.899781645
## nox         -11.913360479
## rm            4.011308385
## age          -0.003731470
## dis          -1.118874607
## rad           0.153730052
## tax          -0.005751054
## ptratio      -0.854984614
## b             0.009073740
## lstat        -0.472423800
```

Finally, display the value for lambda.1se *and* the coefficients associated with it. Again, make sure these are visible in your knitted document.

```r
# Display lambda that is one standard error away from lambda.min
cvfit.house.ridge$lambda.1se
```

```
## [1] 4.781501
```

```r
# Display the coefficients associated with lambda.1se
coef(cvfit.house.ridge, s = "lambda.1se")
```

```
## 14 x 1 sparse Matrix of class "dgCMatrix"
```

```
##                       s1
## (Intercept) 20.637793052
## crim        -0.066138824
## zn           0.019686880
## indus       -0.070144117
## chas1        2.691468060
## nox         -4.963324354
## rm           3.483462851
## age         -0.008046426
## dis         -0.454147171
## rad          0.023146146
## tax         -0.002826675
## ptratio     -0.643074027
## b            0.007326104
## lstat       -0.329596043
```

Question 7 - Cross-validated lasso regression - 10 points

For this question, you will use the same outcome (medv) and the same predictors in the as in the last question, but you will instead conduct a cross-validated lasso regression. *Do NOT split your data into training and test sets for this question*; conduct the analysis on the whole data set. First, conduct the cross-validated lasso regression. Be sure to use the set.seed() provided to make your analysis reproducible.

```
# Prepare data for lasso regression (already formatted as X and y in previous steps)
# X is the matrix of predictors, and y is the outcome (medv)

set.seed(202211) # Ensures reproducibility

# Conduct cross-validated lasso regression (alpha = 1 for lasso regression)
cvfit.house.lasso <- cv.glmnet(x = X, y = y, alpha = 1)
```

Next, display the value for lambda.min and the coefficients associated with it. Make sure these are visible in your knitted document.

```
# Display lambda that minimizes cross-validation error
cvfit.house.lasso$lambda.min
```

```
## [1] 0.009170489
```

```
# Display the coefficients associated with lambda.min
coef(cvfit.house.lasso, s = "lambda.min")
```

```
## 14 x 1 sparse Matrix of class "dgCMatrix"
##                          s1
## (Intercept)  35.707318035
## crim         -0.104719398
## zn            0.044527873
## indus         0.007458193
## chas1         2.698073139
## nox         -17.125645162
## rm            3.829379107
## age           .
```

```
## dis          -1.454866978
## rad           0.285276396
## tax          -0.011295472
## ptratio      -0.942664365
## b             0.009211899
## lstat        -0.523075319
```

Finally, display the value for lambda.1se and the coefficients associated with it. Again, make sure these are visible in your knitted document.

```
# Display lambda that is one standard error away from lambda.min
cvfit.house.lasso$lambda.1se
```

```
## [1] 0.4158705
```

```
# Display the coefficients associated with lambda.1se
coef(cvfit.house.lasso, s = "lambda.1se")
```

```
## 14 x 1 sparse Matrix of class "dgCMatrix"
##                        s1
## (Intercept) 16.268762379
## crim        -0.019439916
## zn           .
## indus        .
## chas1        1.791658205
## nox         -2.085387037
## rm           4.259590140
## age          .
## dis         -0.237030303
## rad          .
## tax          .
## ptratio     -0.771520796
## b            0.006407868
## lstat       -0.517661470
```

Question 3 - 5 points

An important difference between ridge regression and lasso regression is that predictors can be dropped from a model in lasso but not in ridge. The number of predictors set to zero (if any) in lasso depends on the extent of the coefficient shrinkage at a given lambda. Answer the two questions below about the results of your cross-validated lasso models.

1) Among the set of coefficients associated with *lambda.min* in the cross-validated lasso regression, which predictors were set to zero? Please list them. Your answer here: age.

2) Among the set of coefficients associated with *lambda.1se* in the cross-validated lasso regression, which predictors were set to zero? Please list them. Your answer here: zn, indus, age, rad, tax.

3) Which of these - lambda.min or lambda.1se - had more coefficients set to zero? Your answer here: lambda.1se had more coefficients set to zero compared to lambda.min, indicating a simpler model.

END.