STAT 471: Homework 3

Due: October 24, 2021 at 11:59pm

Contents

Instructions

Setup

Pull the latest version of this assignment from Github and set your working directory to `stat-471-fall-2021/homework/homework-3`. Consult the getting started guide if you need to brush up on `R` or `Git`.

Collaboration

The collaboration policy is as stated on the Syllabus:

> "Students are permitted to work together on homework assignments, but solutions must be written up and submitted individually. Students must disclose any sources of assistance they received; furthermore, they are prohibited from verbatim copying from any source and from consulting solutions to problems that may be available online and/or from past iterations of the course."

In accordance with this policy,

Please list anyone you discussed this homework with:

Please list what external references you consulted (e.g. articles, books, or websites):

Writeup

Use this document as a starting point for your writeup, adding your solutions after "**Solution**". Add your R code using code chunks and add your text answers using **bold text**. Consult the preparing reports guide for guidance on compilation, creation of figures and tables, and presentation quality.

Programming

The `tidyverse` paradigm for data wrangling, manipulation, and visualization is strongly encouraged, but points will not be deducted for using base `R`.

Grading

The point value for each problem sub-part is indicated. Additionally, the presentation quality of the solution for each problem (as exemplified by the guidelines in Section 3 of the preparing reports guide will be evaluated on a per-problem basis (e.g. in this homework, there are three problems). There are 100 points possible on this homework, 85 of which are for correctness and 15 of which are for presentation.

Submission

Compile your writeup to PDF and submit to Gradescope.

We'll need to use the following R packages:

```
library(kableExtra)   # for printing tables
library(cowplot)      # for side by side plots
library(glmnetUtils)  # for running ridge and lasso
library(ISLR2)        # necessary for College data
library(pROC)         # for ROC curves
library(tidyverse)
```

We'll also need the plot_glmnet function from Unit 3 Lecture 3:

```
# install.packages("scales")              # dependency of plot_glmnet
source("../../functions/plot_glmnet.R")
```

1 Framingham Heart Study

Heart disease is the leading cause of the death in United States, accounting for one out of four deaths. It is important to identify risk factors for this disease. Many studies have indicated that high blood pressure, high cholesterol, age, gender, race are among the major risk factors.

Starting from the late 1940s, National Heart, Lung and Blood Institute (NHLBI) launched its famous Framingham Heart Study. By now subjects of three generations together with other people have been monitored and followed in the study. Over thousands research papers have been published using these longitudinal data sets.

Using a piece of the data gathered at the beginning of the study, we illustrate how to identify risk factors of heart disease and how to predict this disease.

The data contain the following eight variables for each individual:

Variable	Description
`HD`	Indicator of having heart disease or not
`AGE`	Age
`SEX`	Gender
`SBP`	Systolic blood pressure
`DBP`	Diastolic blood pressure
`CHOL`	Cholesterol level
`FRW`	age and gender adjusted weight
`CIG`	Self-reported number of cigarettes smoked each week

1.1 Data import and exploration

i. Import the data from `stat-471-fall-2021/data/Framingham.dat` into a tibble called `hd_data`, specifying all columns to be integers except `SEX`, which should be a factor. Rename `Heart Disease?` to `HD`, and remove any rows containing `NA` values using `na.omit()`.

```
hd_data = read_csv("../../data/Framingham.dat", col_types = "iifiiiii") %>%
  rename("HD" = "Heart Disease?") %>% na.omit()
```

ii. What is the number of people in this data? What percentage of them have heart disease?

```
num_people = nrow(hd_data)
num_hd = hd_data %>% filter(HD == 1) %>% nrow()
sprintf("The number of people in the data: %i", num_people)
```

```
## [1] "The number of people in the data: 1393"
```

```r
sprintf("The percent of them having heart disease: %f%%", num_hd/num_people*100)
```

```
## [1] "The percent of them having heart disease: 22.038765%"
```

Thus, the number of observations, i.e., people, in the data is 1393 (since there are 1393 rows), and about 22.04% of then have heart disease.

iii. Split `hd_data` into training (80%) and test (20%) sets, using the rows in `train_samples` below for training. Store these in tibbles called `hd_train` and `hd_test`, respectively.

```r
set.seed(5) # seed set for reproducibility (DO NOT CHANGE)
n = nrow(hd_data)
train_samples = sample(1:n, round(0.8*n))
# split hd_data into training and test sets
hd_train = hd_data[train_samples,]
hd_test = hd_data[-train_samples,]
```

iv. Display the age distribution in `hd_train` with a plot. What is the median age?

```r
hd_train %>%
  ggplot(aes(x = AGE)) +
  # create histogram for age distribution
  geom_histogram(binwidth = 1) +
  labs(x = "Age", y = "Count") +
  # add vertical line to indicate median age
  geom_vline(xintercept = median(hd_train$AGE), color = "red",
             linetype = "dashed") +
  theme_bw() + theme(legend.position = "none")
```

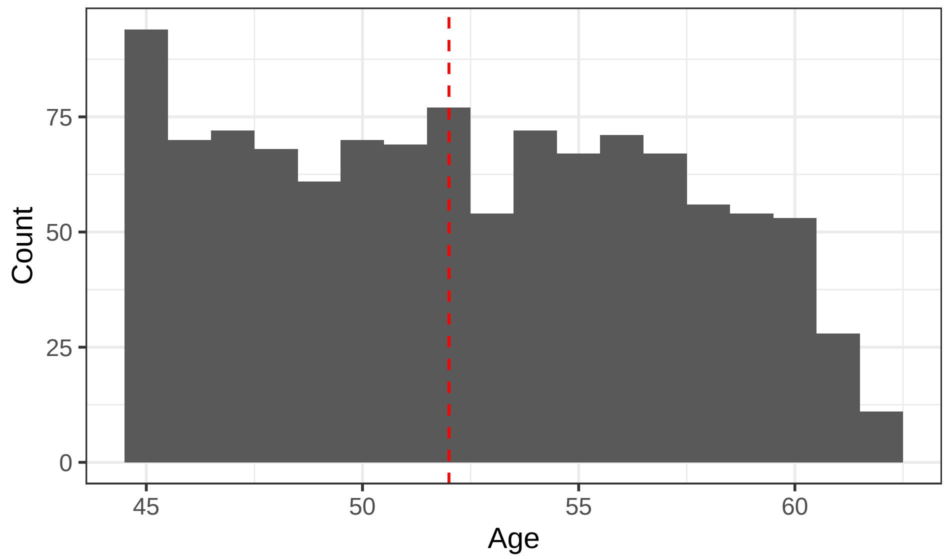

Figure 1: This is a histogram illustrating the age distribution for the data. The dashed red line represents the median age.

```r
# determine median age
median_age = hd_train %>% summarize(median(AGE)) %>% pull()
sprintf("The median age: %i", median_age)
```

```
## [1] "The median age: 52"
```

As seen in Figure 1, the age distribution ranges between 45 and 62. The median age is 52. The distribution is only slightly skewed to the right, and we can observe from Figure 1 that there is a large count of observations/people that are 45 years of age.

v. Use a plot to explore the relationship between heart disease and systolic blood pressure in `hd_train`. What does this plot suggest?

```r
hd_train %>%
  ggplot(aes(x = factor(HD), y = SBP, fill = factor(HD))) +
  # create boxplots to explore relationship
  geom_boxplot() +
  labs(x = "Heart Disease",
       y = "Systolic blood pressure") +
  theme_bw() + theme(legend.position = "none")
```

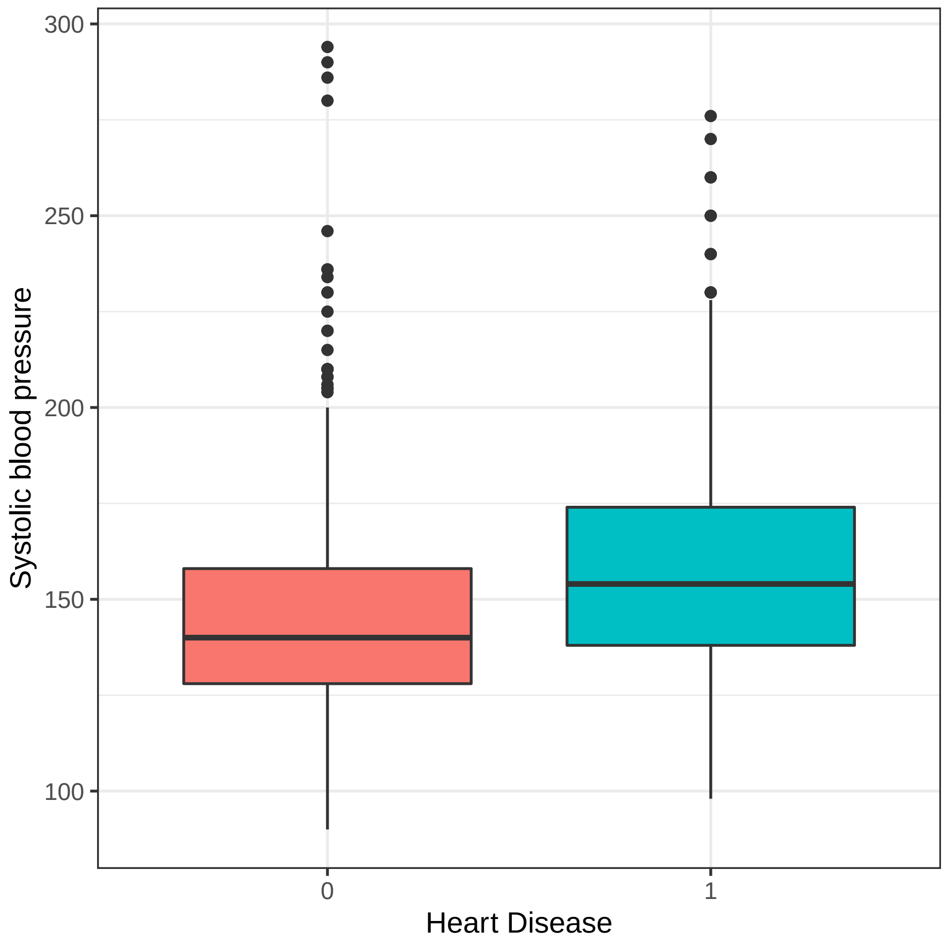

Figure 2: These are boxplots exploring the relationship between heart disease and systolic blood pressure in the training data. We can observe that heart disease tends to be associated with higher systolic blood pressure.

```r
hd_train %>%
  ggplot(aes(x = SBP, y = HD)) +
  geom_jitter(height = 0.1) + # jitter points for greater visibility
  geom_point() +
  labs(x = "Systolic blood pressure",
      y = "Prob(heart disease = 1)") +
  theme_bw()
```

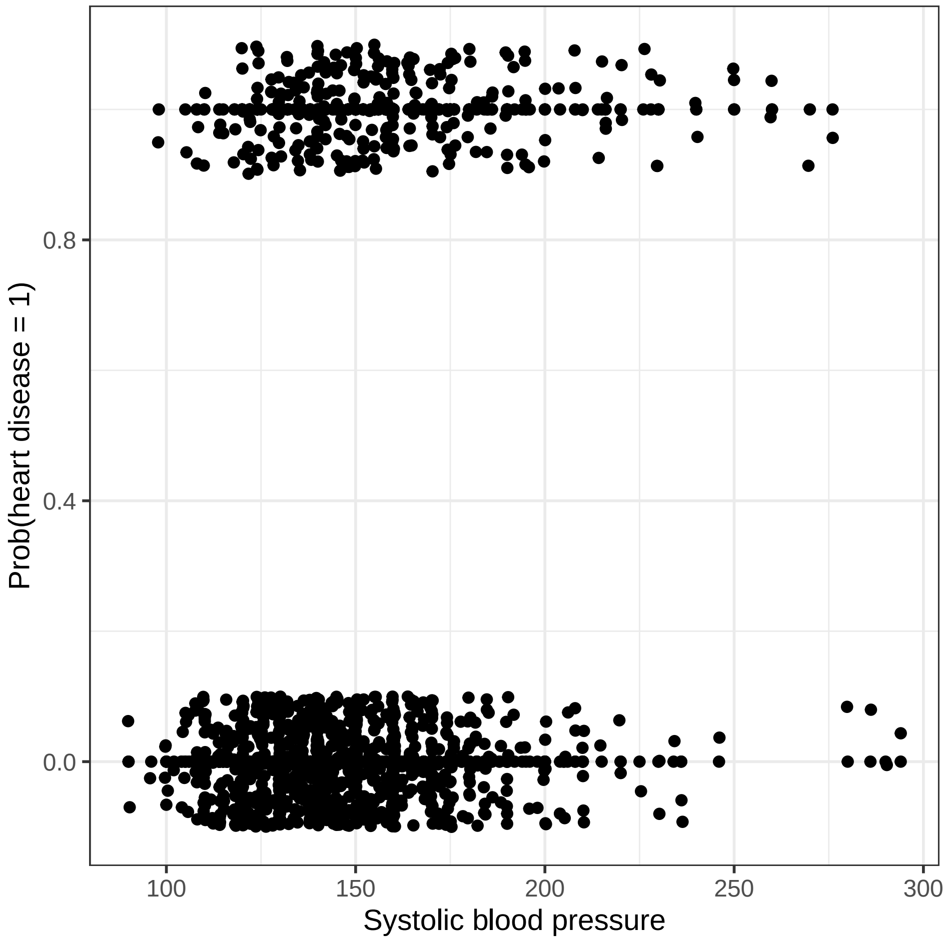

Figure 3: This is a scatterplot further exploring the relationship between heart disease and systolic blood pressure in the training data. Given that heart disease is a binary variable, we jitter the points in the plot.

Figure 2 suggests that heart disease seems to be associated with higher systolic blood pressure, as the box (representing the interquartile range) for observations/people with heart disease is higher in the plot than that for observations/people not having heart disease. That is, those with heart disease tend to have higher systolic blood pressure than those who do not have heart disease. However, note that we can only comment on the association (not causation). In particular, the median systolic blood pressure for those without heart disease is 140, while the median systolic blood pressure for those with heart disease is 154.

We can further see the relationship (as described) in Figure 3, as many of the observations without heart disease dominate at a lower systolic blood pressure reading. However, the boxplots in Figure 2 more clearly portrayed the relationship between heart disease and systolic blood pressure. From the jittered scatterplot in Figure 3, there perhaps does not appear to be a strong relationship between systolic blood pressure and heart disease, but it does seem that those with higher systolic blood pressure tend to have heart disease, as seen in the top of the plot (though it is less clear visually than it is in the boxplots).

1.2 Univariate logistic regression

In this part, we will study the relationship of heart disease with systolic blood pressure using univariate logistic regression.

1.2.1 Logistic regression building blocks

Let's take a look under the hood of logistic regression using a very small subset of the data.

i. Define and print a new data frame called `hd_train_subset` containing `HD` and `SBP` for the individuals in `hd_train` who smoke (exactly) 40 cigarettes per week and have a cholesterol of at least 260.

```
hd_train_subset = hd_train %>%
  # filter to those who smoke 40 cig/wk and have cholesterol at least 260
  filter(CIG == 40 & CHOL >= 260) %>%
  select(HD, SBP)
hd_train_subset
```

```
## # A tibble: 5 x 2
##      HD   SBP
##   <int> <int>
## 1     1   190
## 2     0   142
## 3     1   150
## 4     0   130
## 5     1   130
```

ii. Write down the logistic regression likelihood function using the observations in `hd_train_subset`. **The logistic regression likelihood function using the observations in `hd_train_subset` is defined as the following product:** $\frac{e^{\beta_0+\beta_1\cdot 190}}{1+e^{\beta_0+\beta_1\cdot 190}} \cdot \frac{1}{1+e^{\beta_0+\beta_1\cdot 142}} \cdot \frac{e^{\beta_0+\beta_1\cdot 150}}{1+e^{\beta_0+\beta_1\cdot 150}} \cdot \frac{1}{1+e^{\beta_0+\beta_1\cdot 130}} \cdot \frac{e^{\beta_0+\beta_1\cdot 130}}{1+e^{\beta_0+\beta_1\cdot 130}}$.

iii. Find the MLE based on this subset using `glm()`. Given a value of `SBP`, what is the estimated probability $\mathbb{P}[\text{HD} = 1|\text{SBP}]$?

```
glm_fit_subset = glm(HD ~ SBP, family = "binomial", data = hd_train_subset)
summary(glm_fit_subset)
```

```
##
## Call:
## glm(formula = HD ~ SBP, family = "binomial", data = hd_train_subset)
##
## Deviance Residuals:
##      1       2       3       4       5
##  0.204  -1.318   0.821  -0.950   1.423
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept) -10.1427    12.8838   -0.79     0.43
## SBP           0.0737     0.0925    0.80     0.43
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 6.7301  on 4  degrees of freedom
## Residual deviance: 5.3801  on 3  degrees of freedom
## AIC: 9.38
##
## Number of Fisher Scoring iterations: 5
```

```r
coef(glm_fit_subset)
```

```
## (Intercept)         SBP
##    -10.1427      0.0737
```

As can be seen in the summary of the fit above (and directly based on the coefficients pulled above), the value of β_1 that maximizes the likelihood is 0.0737. And the intercept term is -10.1427. The product obtained under the hood (based on the optimization problem and plugging in these values for β_0 and β_1) is $\frac{e^{-10.1427+0.0737\cdot 190}}{1+e^{-10.1427+0.0737\cdot 190}} \cdot \frac{1}{1+e^{-10.1427+0.0737\cdot 142}} \cdot \frac{e^{-10.1427+0.0737\cdot 150}}{1+e^{-10.1427+0.0737\cdot 150}} \cdot \frac{1}{1+e^{-10.1427+0.0737\cdot 130}} \cdot \frac{e^{-10.1427+0.0737\cdot 130}}{1+e^{-10.1427+0.0737\cdot 130}}$**.**

Given a value of `SBP`, the estimated probability $\mathbb{P}[\textbf{HD} = 1|\textbf{SBP}]$ is $\frac{e^{-10.1427+0.0737\cdot SBP}}{1+e^{-10.1427+0.0737\cdot SBP}}$**.**

iv. Briefly explain how the fitted coefficients in part iii were obtained from the formula in part ii. **The fitted coefficients in part iii were obtained from the formula in part ii by determining the coefficients that maximize the value of the formula in part ii. That is, the coefficients are those that maximize the logistic regression likelihood function, thus serving as the solution to such an optimization problem.**

v. To illustrate this, fix the intercept at its fitted value and define the likelihood as a function of β_1. Then, plot this likelihood in the range $[0, 0.1]$, adding a vertical line at the fitted value of β_1. What do we see in this plot? [Hints: Define the likelihood as a function in R via `likelihood = function(beta_1)(???)`. Use `stat_function()` to plot it.]

```r
# fix intercept at fitted value
beta_0 = coef(glm_fit_subset)["(Intercept)"]
# define likelihood function
likelihood = function(beta_1)(
  (exp(beta_0 + beta_1*190)/(1 + exp(beta_0 + beta_1*190)))*
    (1/(1 + exp(beta_0 + beta_1*142)))*
    (exp(beta_0 + beta_1*150)/(1 + exp(beta_0 + beta_1*150)))*
    (1/(1 + exp(beta_0 + beta_1*130)))*
    (exp(beta_0 + beta_1*130)/(1 + exp(beta_0 + beta_1*130))))

fitted_value = glm_fit_subset$coefficients["SBP"]
beta_1 = seq(0, 0.1, by = 0.01)
tibble = tibble(x = beta_1)
tibble %>% ggplot(aes(x = beta_1)) +
  # plot likelihood function
  stat_function(fun = likelihood) +
  # add a vertical line at the fitted value of beta_1
  geom_vline(xintercept = fitted_value) +
  labs(x = "Beta 1 (i.e., coefficient for SBP)", y = "Likelihood") +
  theme_bw()
```

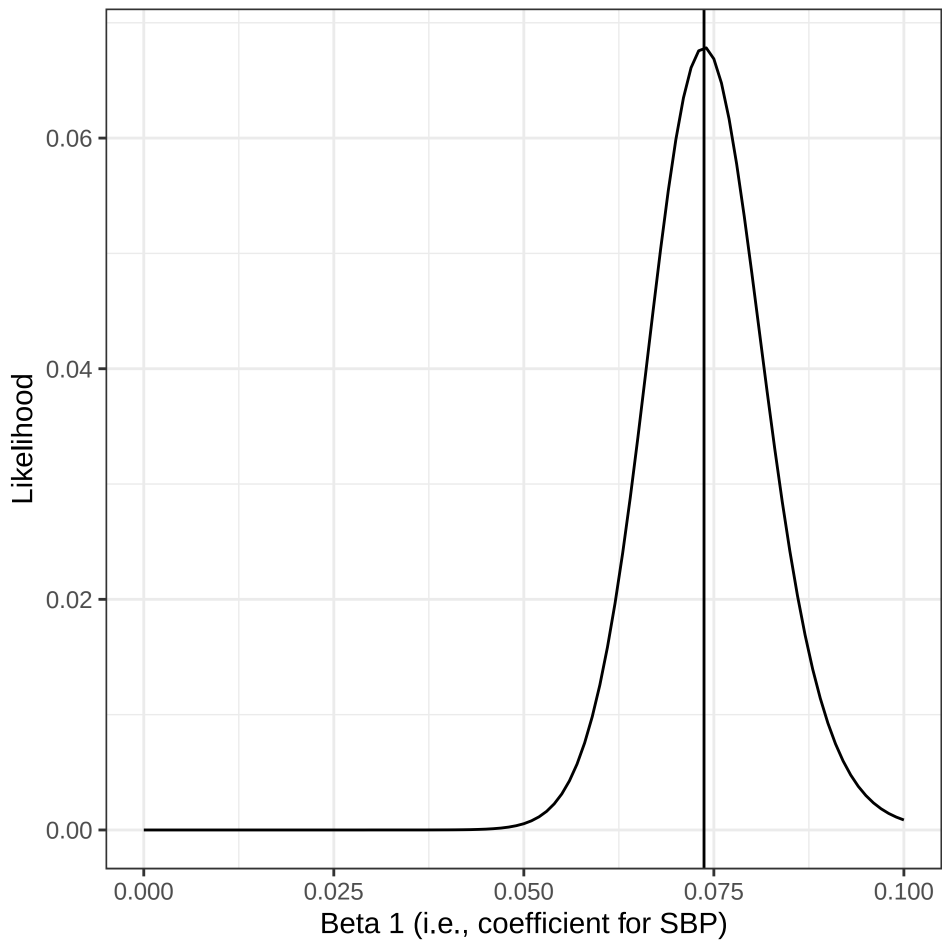

Figure 4: Defining the likelihood as a function of beta 1 (the coefficient for SBP, i.e., systolic blood pressure), this is a plot of this likelihood in the range between 0 and 0.1, with a vertical line at the fitted value of beta 1.

In Figure 4, we can see the likelihood function and observe that the vertical line is indicating the value of beta that is maximizing this likelihood. That is, from this plot, it is evident that the fitted value of β_1 corresponds to the value of β_1 at the peak of this function (i.e., the likelihood function). Thus, this plot highlights how the fitted value serves as an optimal solution given this likelihood function.

1.2.2 Univariate logistic regression on the full data

i. Run a univariate logistic regression of `HD` on `SBP` using the full training data `hd_train`. According to the estimated coefficient, how do the odds of heart disease change when `SBP` increases by 1?

```
glm_fit = glm(HD ~ SBP, family = "binomial", data = hd_train)
summary(glm_fit)
```

```
##
## Call:
## glm(formula = HD ~ SBP, family = "binomial", data = hd_train)
##
## Deviance Residuals:
##    Min      1Q  Median      3Q     Max
## -1.680  -0.717  -0.627  -0.521   2.116
```

```
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept) -3.75577    0.39321   -9.55  < 2e-16 ***
## SBP          0.01663    0.00251    6.62  3.6e-11 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1186.3  on 1113  degrees of freedom
## Residual deviance: 1140.9  on 1112  degrees of freedom
## AIC: 1145
##
## Number of Fisher Scoring iterations: 4
```

```
coef(glm_fit)
```

```
## (Intercept)         SBP
##     -3.7558      0.0166
```

As seen in the summary of the fit above as well as directly from the pulled fitted coefficients, the estimated coefficient for SBP (which is statistically significant) is 0.01663. Interpreting this, the log odds of heart disease increases by 0.01663 when `SBP` increases by 1. Moreover, the odds of heart disease increases by a multiplicative factor of $e^{0.01663} \approx 1.017$ when `SBP` increases by 1 (i.e., the odds increases by 1.7%).

ii. Plot the logistic regression fit along with a scatter plot of the data. Use `geom_jitter()` instead of `geom_point()` to better visualize the data. Based on the plot, roughly what is the estimated probability of heart disease for someone with `SBP` $= 100$?

```
hd_train %>%
  ggplot(aes(x = SBP, y = HD)) +
  # jitter points to better visualize the data
  geom_jitter(height = .05) +
  geom_smooth(method = "glm", formula = "y~x",
              method.args = list(family = "binomial"), se = FALSE) +
  labs(x = "Systolic blood pressure", y = "Prob(heart disease = 1)") +
  theme_bw()
```

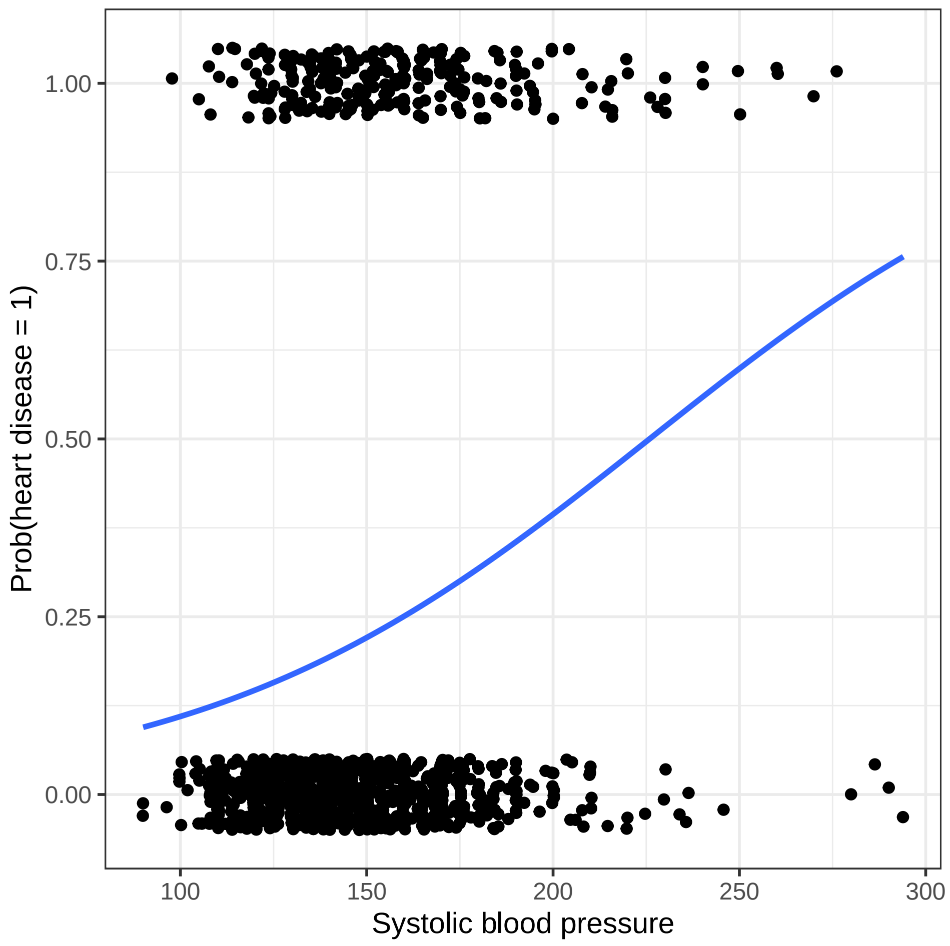

Figure 5: This is a plot of the logistic regression fit along with a scatter plot of the data. We jitter the points to better visualize the data.

Based on the plot in Figure 5, the estimated probability of heart disease for someone with `SBP` = 100 is roughly 0.11, i.e., 11%. (We can also note that the plotted logistic curve is not very steeply sloped in the sense that the S shape is not very pronounced.)

We can also apply the predict function to get the probability of heart disease for someone with `SBP` = 100.

```
prediction = predict(glm_fit, newdata = list(SBP = 100), type = "response")
sprintf("The probability of heart disease for someone with `SBP` = 100: %f",
        prediction)
```

```
## [1] "The probability of heart disease for someone with `SBP` = 100: 0.109770"
```

1.3 Multiple logistic regression

i. Run a multiple logistic regression of `HD` on all of the other variables in the data. Other things being equal, do the estimated coefficient suggest that males are more or less prone to heart disease? Other things being equal, what impact does an increase in `AGE` by 10 years have on the odds of heart disease (according to the estimated coefficients)?

```
glm_fit_multiple = glm(HD ~ ., family = "binomial", data = hd_train)
summary(glm_fit_multiple)
```

```
##
## Call:
## glm(formula = HD ~ ., family = "binomial", data = hd_train)
##
## Deviance Residuals:
##    Min      1Q  Median      3Q     Max
## -1.737  -0.726  -0.554  -0.320   2.458
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept) -8.46586    1.12415   -7.53    5e-14 ***
## AGE          0.06151    0.01659    3.71  0.00021 ***
## SEXFEMALE   -0.96813    0.17468   -5.54    3e-08 ***
## SBP          0.01568    0.00432    3.63  0.00029 ***
## DBP          0.00302    0.00837    0.36  0.71866
## CHOL         0.00439    0.00171    2.57  0.01007 *
## FRW          0.00618    0.00449    1.38  0.16911
## CIG          0.01080    0.00675    1.60  0.10953
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1186.3  on 1113  degrees of freedom
## Residual deviance: 1077.5  on 1106  degrees of freedom
## AIC: 1094
##
## Number of Fisher Scoring iterations: 4
```

From the above summary of the multiple logistic regression fit, we can observe that the estimated coefficient for `SEXFEMALE` is -0.96813. Thus, as the coefficient is statistically significant, other things being equal, the estimated coefficient suggests that males are more prone to heart disease than females, as being female decreases the log odds of heart disease (since this coefficient is negative) and furthermore the odds and probability of heart disease for females. Other things being equal, according to the estimated coefficients, an increase in `AGE` by 10 years has an impact of increasing the odds of heart disease by a multiplicative factor of $e^{10 \cdot 0.06151} \approx 1.850$.

ii. Mary is a patient with the following readings: `AGE=50, SEX=FEMALE, SBP=110, DBP=80, CHOL=180, FRW=105, CIG=0`. According to the fitted model, what is the estimated probability Mary has heart disease?

```
mary = list(AGE = 50, SEX = "FEMALE", SBP = 110, DBP = 80,
            CHOL = 180, FRW = 105, CIG = 0)
prediction = predict(glm_fit_multiple, newdata = mary, type = "response")
sprintf("The estimated probability Mary has heart disease: %f", prediction)
```

```
## [1] "The estimated probability Mary has heart disease: 0.049562"
```

Thus, according to the fitted model, the estimated probability Mary has heart disease is 0.049562, or approximately 4.96%.

iii. What are the misclassification rate, false positive rate, and false negative rate of the logistic regression classifier (based on the probability threshold of 0.5) on `hd_test`? Print these in a nice table. Plot the ROC curve, and add a red point to the plot corresponding to the threshold of 0.5 (recalling that the true positive rate is one minus the false negative rate). What is the AUC? How does it compare to that of a classifier that guesses randomly?

```r
# compute fitted probabilities and add predictions to the tibble
fitted_probabilities = predict(glm_fit_multiple,
                               newdata = hd_test, type = "response")
predictions = as.numeric(fitted_probabilities > 0.5)
hd_test = hd_test %>% mutate(predicted_hd = predictions)

# calculate misclassification rate
misclassification_rate = hd_test %>%
  summarise(misclassification_rate = mean(HD != predicted_hd)) %>%
  pull(misclassification_rate)

# false positive rate = number of false positives / total actual negatives
false_positives = hd_test %>% select(HD, predicted_hd) %>%
  filter(HD == 0 & predicted_hd == 1) %>% nrow()
actual_negatives = hd_test %>% select(HD, predicted_hd) %>%
 filter(HD == 0) %>% nrow()
fp_rate = false_positives / actual_negatives

# false negative rate = number of false negatives / total actual positives
false_negatives = hd_test %>% select(HD, predicted_hd) %>%
  filter(HD == 1 & predicted_hd == 0) %>% nrow()
actual_positives = hd_test %>% select(HD, predicted_hd) %>%
  filter(HD == 1) %>% nrow()
fn_rate = false_negatives / actual_positives

# create table of these three metrics
model_metrics = tribble(
  ~metric, ~classifier_performance,
  #------/-----------------------/
  "Misclassification rate", misclassification_rate,
  "False positive rate", fp_rate,
  "False negative rate", fn_rate
  )

# print these metrics in nice table
model_metrics %>% kable(format = "latex", row.names = NA,
                        booktabs = TRUE,
                        digits = 2,
                        col.names = c("Metric",
                                      "Classifier performance"),
                        caption = "These are the three metrics
                        misclassification error, false positive rate,
                        and false negative rate for the logistic
                        regression classifier on the test data.") %>%
  kable_styling(position = "center") %>%
  kable_styling(latex_options = "HOLD_position")
```

Table 2: These are the three metrics misclassification error, false positive rate, and false negative rate for the logistic regression classifier on the test data.

Metric	Classifier performance
Misclassification rate	0.20
False positive rate	0.02
False negative rate	0.89

As can be seen in Table 2, for the logistic regression classifier (based on the probability threshold of 0.5) on `hd_test`, the misclassification rate is 20.072%, the false positive rate is 2.252%, and the false negative rate is 89.474%.

```
# plot the ROC curve for this classifier
roc_data = roc(hd_test %>% pull(HD), fitted_probabilities)
tibble(FPR = 1-roc_data$specificities,
       TPR = roc_data$sensitivities) %>%
  ggplot(aes(x = FPR, y = TPR)) +
  geom_line() +
  geom_abline(slope = 1, linetype = "dashed") +
  # add a red point to the plot corresponding to the threshold of 0.5
  geom_point(x = fp_rate, y = 1-fn_rate, colour = "red") +
  theme_bw()
```

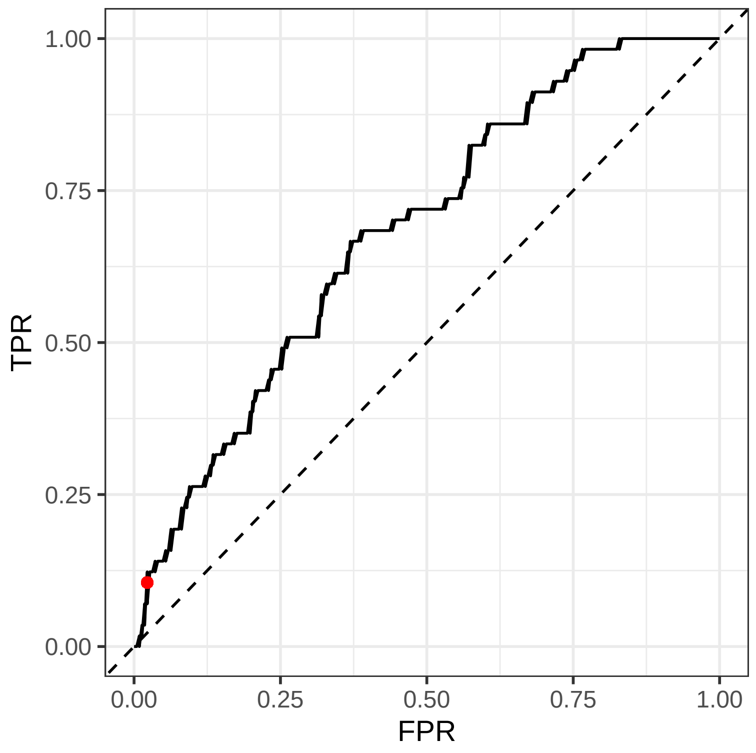

Figure 6: This is a plot of the ROC curve with the true positive rate on the y-axis and the false positive rate on the x-axis. There is a red point added to the plot corresponding to the threshold of 0.5, which appears towards the bottom left corner.

```
# determine the AUC
auc = roc_data$auc
sprintf("The area under the curve: %f", auc)
```

```
## [1] "The area under the curve: 0.681208"
```

Based on the ROC curve in Figure 6, the AUC is 0.681208. Thus comparing it to that of a classifier that guesses randomly, it performs better. A classifier that guesses randomly has an area under the curve of 0.5, so any value of AUC between 0.5 and 1 performs better than a classifier that merely guesses randomly. Since the AUC here is 0.681208, the classifier performs better than one that simply guesses randomly.

2 College Applications

Next, we will examine the `College` dataset from the `ISLR` package. According to the documentation, these data contain "statistics for a large number of US Colleges from the 1995 issue of US News and World Report." The goal will be to predict the acceptance rate.

Next, let us make a few small adjustments to the data:

```
college_data = ISLR2::College %>%
  bind_cols(Name = rownames(ISLR2::College)) %>% # add college names
  relocate(Name) %>%                             # put name column first
  mutate(Accept = Accept/Apps) %>%               # redefine `Accept`
  select(-Private,-Apps) %>%                     # remove `Private` and `Apps`
  as_tibble()                                    # cast to tibble
```

Now, let's take a look at the data and its documentation:

```
college_data                                  # take a look at the data
```

```
## # A tibble: 777 x 17
##    Name       Accept Enroll Top10perc Top25perc F.Undergrad P.Undergrad Outstate
##    <chr>       <dbl>  <dbl>     <dbl>     <dbl>       <dbl>       <dbl>    <dbl>
##  1 Abilene C~  0.742    721        23        52        2885         537     7440
##  2 Adelphi U~  0.880    512        16        29        2683        1227    12280
##  3 Adrian Co~  0.768    336        22        50        1036          99    11250
##  4 Agnes Sco~  0.837    137        60        89         510          63    12960
##  5 Alaska Pa~  0.756     55        16        44         249         869     7560
##  6 Albertson~  0.816    158        38        62         678          41    13500
##  7 Albertus ~  0.963    103        17        45         416         230    13290
##  8 Albion Co~  0.906    489        37        68        1594          32    13868
##  9 Albright ~  0.808    227        30        63         973         306    15595
## 10 Alderson-~  0.856    172        21        44         799          78    10468
## # ... with 767 more rows, and 9 more variables: Room.Board <dbl>, Books <dbl>,
## #   Personal <dbl>, PhD <dbl>, Terminal <dbl>, S.F.Ratio <dbl>,
## #   perc.alumni <dbl>, Expend <dbl>, Grad.Rate <dbl>
```

```
?College                                   # read the documentation
```

Note that `Accept` is now the acceptance *rate*, and will serve as our response variable. We will use the 15 variables aside from `Name` and `Accept` as our features.

Let's define the 80%/20% train/test partition:

```
set.seed(471) # seed set for reproducibility (DO NOT CHANGE)
n = nrow(college_data)
```

```
train_samples = sample(1:n, round(0.8*n))
college_train = college_data %>% filter(row_number() %in% train_samples)
college_test = college_data %>% filter(!(row_number() %in% train_samples))
```

In what follows, we will do some exploratory data analysis and build some predictive models on the training data `college_train`.

2.1 Exploratory data analysis

Please use the training data `college_train` to answer the following EDA questions.

i. Create a histogram of `Accept`, with a vertical line at the median value. What is this median value? Which college has the smallest acceptance rate in the training data, and what is this rate? How does this acceptance rate (recall the data are from 1995) compare to the acceptance rate for the same university in 2020? Look up the latter figure on Google.

```
college_train %>%
  ggplot(aes(x = Accept)) +
  geom_histogram() +
  labs(x = "Acceptance rate", y = "Count") +
  # add vertical line at the median value for Accept
  geom_vline(xintercept = median(college_train$Accept), color = "red") +
  theme_bw()
```

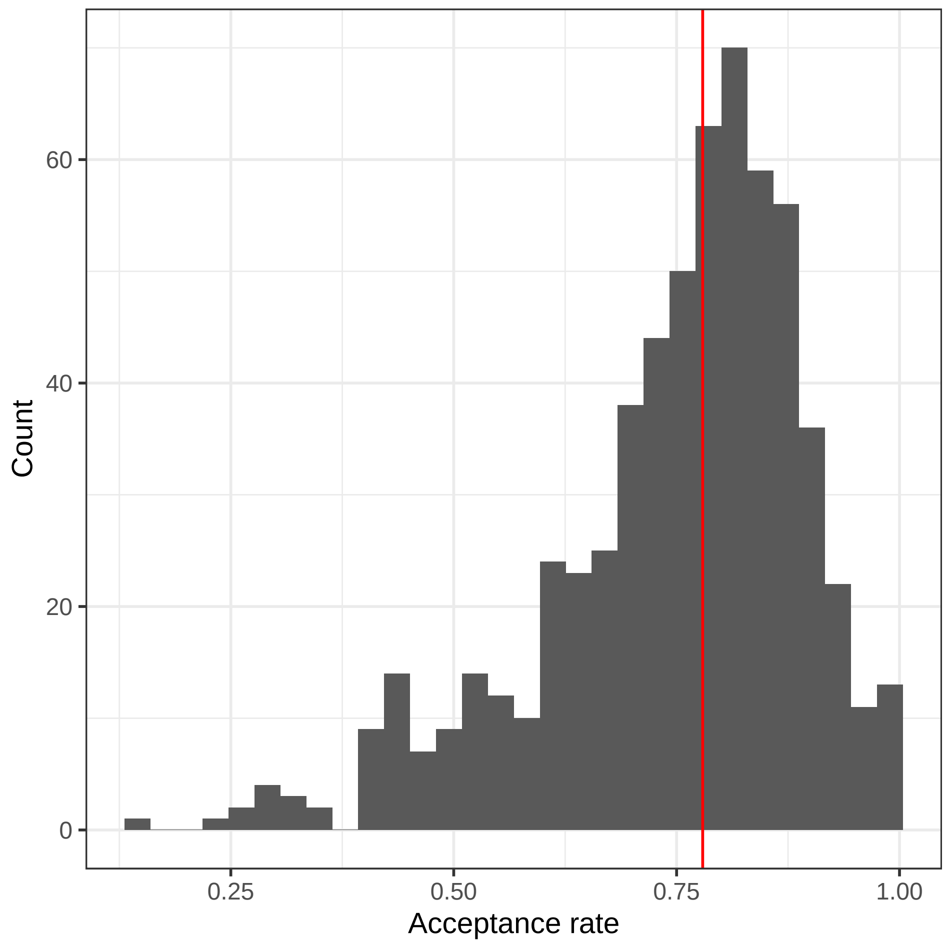

Figure 7: This is a histogram of the acceptance rates for colleges in the data, with a vertical line at the median value.

```r
median_accept = median(college_train$Accept)
sprintf("The median acceptance rate in the data: %f", median_accept)
```

```
## [1] "The median acceptance rate in the data: 0.779250"
```

As observed in the histogram in Figure 7, the median acceptance rate is 0.779, i.e., about 77.93%. From the histogram, we can see that the distribution of acceptance rates in the data is left-skewed with many colleges having high acceptance rates and a few colleges with low acceptance rates making up the tail on the left side.

```r
smallest_accept_name = college_train %>% arrange(Accept) %>%
  head(1) %>% pull(Name)
smallest_accept = college_train %>% arrange(Accept) %>% head(1) %>% pull(Accept)
sprintf("The college with the smallest acceptance rate in the data: %s",
        smallest_accept_name)
```

```
## [1] "The college with the smallest acceptance rate in the data: Harvard University"
```

```r
sprintf("The college has an acceptance rate of %s", smallest_accept)
```

```
## [1] "The college has an acceptance rate of 0.156148575549946"
```

Thus, as discovered immediately above, the college with the smallest acceptance rate in the

training data is Harvard University. The school's acceptance rate in the data is approximately 15.615%. The acceptance rate for the same university in 2020 was 5.2%. Thus, Harvard University's acceptance rate in the data (the data being from 1995) is about three times higher than the college's acceptance rate in 2020.

ii. Produce separate plots to explore the relationships between `Accept` and the following three features: `Grad.Rate`, `Top10perc`, and `Room.Board`.

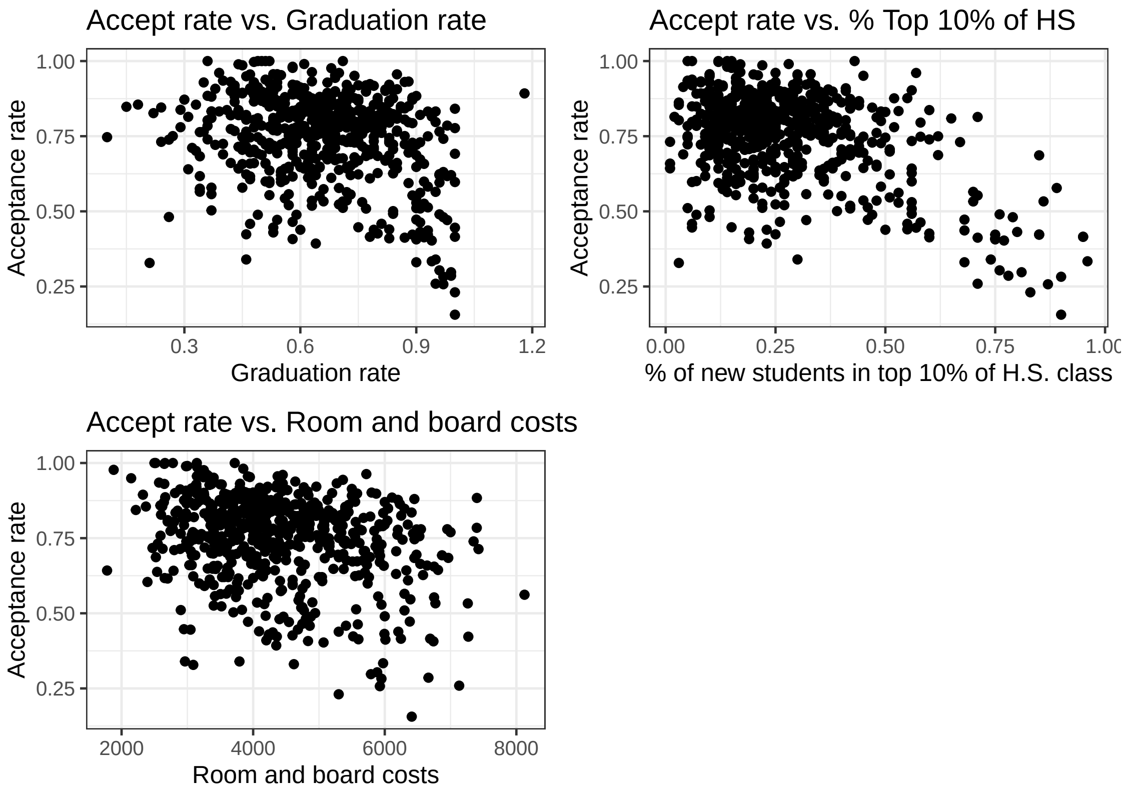

Figure 8: These scatterplots show the relationship between acceptance rate and the following three features: graduation rate, the percent of new students in top ten percent of their high school class, and cost of room and board.

Examining the scatter plots in Figure 8, we can observe that for all three plots, there appears to be a negative association, i.e., between `Accept` and `Grad.Rate`, between `Accept` and `Top10perc`, and between `Accept` and `Room.Board`. This suggests that with higher acceptance rates, there are lower graduation rates, a smaller percentage of new students in the 10% of their high school class, and lower room and board costs. The negative association is most pronounced between the acceptance rate and the percent of new students in the top 10% of their high school class. These associations make intuitive sense as we would expect more selective colleges to have more top students, have higher room and board costs (perhaps as a factor being related to colleges' prestige), and higher graduation rates (perhaps confounded with the fact that the students are coming more from the top of their class and may be more dedicated to their education).

iii. For the most selective college in the training data, what fraction of new students were in the top 10% of their high school class? For the colleges with the largest fraction of new students in the top 10% of their high school class (there may be a tie), what were their acceptance rates?

```
most_selective_top10perc = college_train %>%
  filter(Name == "Harvard University") %>% pull(Top10perc)
sprintf("Percent of Harvard new students in top 10 percent of HS class: %d%%",
        most_selective_top10perc)
```

```
## [1] "Percent of Harvard new students in top 10 percent of HS class: 90%"
```

Thus, for Harvard University, the most selective college in the training data, 90% (i.e., 0.9) of new students were in the top 10 percent of their high school class.

```
college_train %>% arrange(desc(Top10perc)) %>%
  head(5) %>% select(Name, Accept, Top10perc) %>%
  kable(format = "latex", row.names = NA,
                        booktabs = TRUE,
                        digits = 2,
                        col.names = c("Name",
                                      "Acceptance rate",
                                      "% Top 10% in HS class"),
                        caption = "These are the colleges with
        the largest fraction of new students in the top 10 percent of
        their high school class.") %>%
  kable_styling(position = "center") %>%
  kable_styling(latex_options = "HOLD_position")
```

Table 3: These are the colleges with the largest fraction of new students in the top 10 percent of their high school class.

Name	Acceptance rate	% Top 10% in HS class
Massachusetts Institute of Technology	0.33	96
Harvey Mudd College	0.42	95
University of California at Berkeley	0.42	95
Duke University	0.28	90
Harvard University	0.16	90

We can observe in Table 3 that for the colleges with the largest fraction of new students in the top 10% of their high school class, their acceptance rates ranged between about 16% and 42%. We can observe that the college with the largest fraction of new students in the top 10% of their high school class is the Massachusetts Institute of Technology (MIT), with 96% of new students in the top 10% of their high school class. MIT had an acceptance rate of 0.334 (i.e., 33.4%).

2.2 Predictive modeling

Now we will build some predictive models for `Accept`. For convenience, let's remove the `Name` variable from the training and test sets since it is not a feature we will be using for prediction:

```
college_train = college_train %>% select(-Name)
college_test = college_test %>% select(-Name)
```

2.2.1 Ordinary least squares

i. Using the training set `college_train`, run a linear regression of `Accept` on the other features and display the regression summary. What fraction of the variation in the response do the features explain?

```r
lm_fit = lm(Accept ~ ., data = college_train)
summary(lm_fit)
```

```
##
## Call:
## lm(formula = Accept ~ ., data = college_train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -0.5515 -0.0701  0.0115  0.0855  0.2988
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  1.10e+00   5.17e-02   21.29  < 2e-16 ***
## Enroll       5.59e-05   2.15e-05    2.60  0.00951 **
## Top10perc   -3.26e-03   7.20e-04   -4.52  7.3e-06 ***
## Top25perc    3.20e-05   5.83e-04    0.05  0.95622
## F.Undergrad -8.85e-06   4.42e-06   -2.00  0.04552 *
## P.Undergrad -9.74e-06   4.07e-06   -2.39  0.01710 *
## Outstate     6.74e-06   2.26e-06    2.98  0.00304 **
## Room.Board  -1.92e-05   6.19e-06   -3.10  0.00205 **
## Books       -7.93e-05   3.03e-05   -2.62  0.00901 **
## Personal     7.43e-06   8.13e-06    0.91  0.36093
## PhD         -1.66e-04   5.93e-04   -0.28  0.78012
## Terminal    -1.04e-04   6.55e-04   -0.16  0.87403
## S.F.Ratio   -5.76e-03   1.69e-03   -3.42  0.00068 ***
## perc.alumni  1.18e-03   5.51e-04    2.14  0.03290 *
## Expend      -7.28e-06   1.54e-06   -4.73  2.8e-06 ***
## Grad.Rate   -1.13e-03   3.84e-04   -2.94  0.00338 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.123 on 606 degrees of freedom
## Multiple R-squared:  0.32,   Adjusted R-squared:  0.303
## F-statistic:   19 on 15 and 606 DF,  p-value: <2e-16
```

```r
r_squared = summary(lm_fit)$r.squared
sprintf("The R-squared value for the linear regression: %f", r_squared)
```

```
## [1] "The R-squared value for the linear regression: 0.319955"
```

From the summary of the linear regression of `Accept` on the other features (and directly computed by pulling the R-squared value), we can observe that the features explain about 0.32 (or 32%) of the variation in the response (i.e., `Accept`).

ii. Do the signs of the fitted coefficients for `Grad.Rate`, `Top10perc`, and `Room.Board` align with the directions of the univariate relationships observed in part iii of the EDA section? **The signs of the fitted coefficients for `Grad.Rate`, `Top10perc`, and `Room.Board` do align with the directions of the univariate relationships observed in part iii of the EDA section. That is, the coefficients are negative, which aligns with the negative associations we observed in the EDA section when plotting `Accept` against each of these three variables (`Grad.Rate`, `Top10perc`, and `Room.Board`).**

2.2.2 Ridge regression

i. Fit a 10-fold cross-validated ridge regression to the training data and display the CV plot. What is the value of lambda selecting according to the one-standard-error rule?

```
set.seed(3) # set seed before cross-validation for reproducibility

ridge_fit = cv.glmnet(Accept ~ ., alpha = 0, nfolds = 10, data = college_train)
```

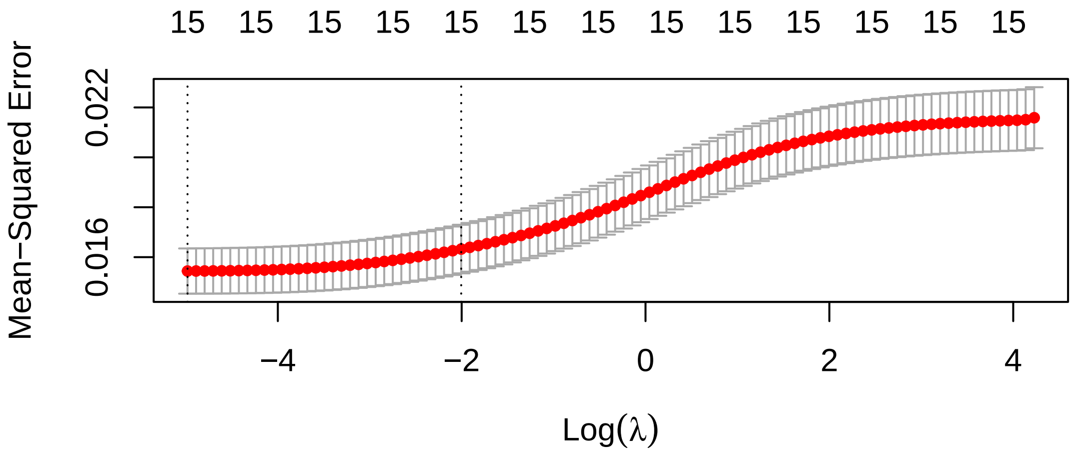

Figure 9: This is the CV plot for the 10-fold cross-validated ridge regression model on the training data.

```
lambda = ridge_fit$lambda.1se
sprintf("The value of lambda based on the one-standard-error rule: %f",
        lambda)
```

```
## [1] "The value of lambda based on the one-standard-error rule: 0.134640"
```

In Figure 9, we have the CV plot for a 10-fold cross-validated ridge regression model to the training data. Corresponding to the right vertical dashed line on the plot (on the log scale), the value of lambda selected according to the one-standard-error rule is about 0.13.

ii. UPenn is one of the colleges in the training set. During the above cross-validation process (excluding any subsequent refitting to the whole training data), how many ridge regressions were fit on data that included UPenn?

```
num_lambda = length(ridge_fit$lambda)
sprintf("The number of lambda values we fit for each fold: %i", num_lambda)
```

```
## [1] "The number of lambda values we fit for each fold: 100"
```

The number of ridge regressions fit on data that included UPenn is $(10 - 1) \cdot 100 = 9 \cdot 100 = 900$. This is because for one of the ten folds, UPenn is considered out of fold. That is, there is one fold in which UPenn is in a fold such that that fold is held out. For each of the other nine folds, we fit 100 ridge regression fits corresponding to the 100 values of lambda that we explore (i.e., for each of these, a regression is fit for each value of lambda). Thus, we fit 900 ridge regression fits on data that included UPenn, excluding subsequent refitting to the whole training data.

iii. Use `plot_glmnet` (introduced in Unit 3 Lecture 3) to visualize the ridge regression fitted coefficients, highlighting 6 features using the `features_to_plot` argument. By examining this plot, answer the following questions. Which of the **highlighted** features' coefficients change sign as lambda increases? Among the highlighted features whose coefficient does not change sign, which feature's coefficient magnitude does not increase monotonically as lambda decreases?

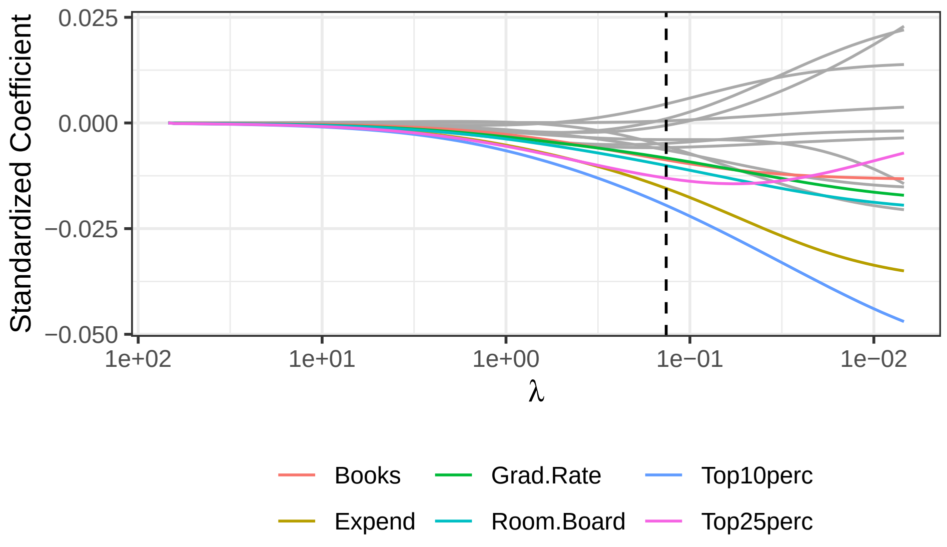

Figure 10: This is the trace plot for the ridge regression model, visualizing the ridge regression fitted coefficients, highlighting 6 features.

By examining Figure 10, we can observe that of the highlighted features' coefficients, none of them change sign as lambda increases. Among the highlighted features whose coefficient does not change sign, one feature's coefficient magnitude does not increase monotonically as lambda decreases, i.e., Top25perc. That is, in Figure 10, we can see that as lambda decreases, the Top25perc coefficient goes from increasing in magnitude (becoming more negative) until lambda is a bit less than 0.1, at which point the coefficient begins decreasing in magnitude (becoming more positive).

iv. Let's collect the least squares and ridge coefficients into a tibble:

```
coeffs = tibble(lm_coef = coef(lm_fit)[-1],
                ridge_coef = coef(ridge_fit, s = "lambda.1se")[-1,1],
                features = names(coef(lm_fit)[-1]))
coeffs
```

```
## # A tibble: 15 x 3
##        lm_coef   ridge_coef features
##          <dbl>        <dbl> <chr>
##  1  0.0000559  -0.000000638 Enroll
##  2 -0.00326    -0.00110     Top10perc
##  3  0.0000320  -0.000658    Top25perc
##  4 -0.00000885 -0.000000814 F.Undergrad
##  5 -0.00000974 -0.00000406  P.Undergrad
##  6  0.00000674  0.000000249 Outstate
##  7 -0.0000192  -0.00000916  Room.Board
##  8 -0.0000793  -0.0000511   Books
##  9  0.00000743  0.000000707 Personal
## 10 -0.000166   -0.000348    PhD
## 11 -0.000104   -0.000321    Terminal
## 12 -0.00576    -0.00140     S.F.Ratio
## 13  0.00118     0.000371    perc.alumni
## 14 -0.00000728 -0.00000294  Expend
## 15 -0.00113    -0.000485    Grad.Rate
```

Answer the following questions by calling `summarise` on `coeffs`. How many features' least squares and ridge regression coefficients have different signs? How many features' least squares coefficient is smaller in magnitude than their ridge regression coefficient?

```r
coeffs %>%
  mutate(diff_signs = lm_coef*ridge_coef) %>%
  mutate(magnitude_diff = abs(ridge_coef) - abs(lm_coef)) %>%
  # summarize sums of interest
  summarise(num_diff_signs = sum(diff_signs < 0),
            lm_coef_smaller = sum(magnitude_diff > 0)) %>%
  kable(format = "latex", row.names = NA,
                        booktabs = TRUE,
                        digits = 2,
                        col.names =
          c("Number of features with different sign coefficients",
            "Number of features with smaller least squares coefficient"),
                        caption = "This is a summary of the number of
                        features with least squares and ridge regression
                        coefficients that have different signs and the
        number of features that have a least squares coefficient that is
        smaller in magnitude than their ridge regression coefficient.") %>%
  kable_styling(position = "center") %>%
  kable_styling(latex_options = "HOLD_position")
```

Table 4: This is a summary of the number of features with least squares and ridge regression coefficients that have different signs and the number of features that have a least squares coefficient that is smaller in magnitude than their ridge regression coefficient.

Number of features with different sign coefficients	Number of features with smaller least squares coefficient
2	3

Thus, as can be seen in Table 4, there are two features that have least squares and ridge regression coefficients that have different signs. There are three features that have a least squares coefficient that is smaller in magnitude than their ridge regression coefficient.

v. Suppose instead that we had a set of training features X^{train} such that $n_{\text{train}} = p$ and

$$X_{ij}^{\text{train}} = \begin{cases} 1, & \text{if } i = j \\ 0, & \text{if } i \neq j. \end{cases}$$

Which of the following phenomena would have been possible in this case?

- Having a feature's ridge regression coefficient change signs based on lambda
- Having a feature's ridge regression coefficient decrease in magnitude as lambda decreases
- Having a feature's coefficients from least squares and ridge regression (the latter based on lambda.1se) have different signs
- Having a feature's coefficient from least squares be smaller in magnitude than its coefficient from ridge regression (based on lambda.1se)

In this particular case, we have 1's on the diagonal. The features are uncorrelated, so we do not have that features are interacting with each other. The nth observation is only a function of ith parameter, and we are trying to minimize the difference between the response and prediction, which is minimized when beta = y. So we are shrinking the OLS coefficient, and the amount we shrink by is dictated by lambda. If lambda = 0, we are not shrinking at all. When we increase lambda, we are putting a bigger penalty on the betas being wild, shrinking them

more. We end up with the result that the ridge coefficient is the OLS coefficient multiplied by the factor $\frac{1}{1+\lambda}$. This makes it such that with this simple case, wedo not get such unpredictable weirdness as these phenomenon refer to. It is the interactions among features wreaking havoc that could lead to such strange phenomena. NONE of the described phenomena would have been possible in this simple case.

It would NOT have been possible in this case to have a feature's ridge regression coefficient change signs based on lambda. The coefficient is simply changing by a factor of $\frac{1}{1+\lambda}$ (and $\lambda \geq 0$) (thus, in a sense, scrunching). Thus, changes in lambda cannot contribute to a change in sign of a feature's ridge regression coefficient, i.e., there is never a case where a ridge regression coefficient changes sign based on lambda.

NO, it would not have been possible in this case to have a feature's ridge regression coefficient decrease in magnitude as lambda decreases. This is because the magnitude in the denominator of $\frac{1}{1+\lambda}$ would be decreasing with a smaller lambda such that (relative to OLS) a feature's ridge regression coefficient is increasing (rather than decreasing) as lambda decreases.

It would NOT have been possible in this case to have a feature's coefficients from least squares and ridge regression (the latter based on lambda.1se) have different signs. The ridge regression coefficient is simply the beta from OLS multiplied by a factor of $\frac{1}{1+\lambda}$, and since this multiplicative factor is positive, the sign of a feature's coefficients from least squares and ridge regression would be the same, i.e., we do not have a lambda such that a feature's coefficients from least squares and ridge regression have different signs.

It would NOT have been possible in this case to have a feature's coefficient from least squares be smaller in magnitude than its coefficient from ridge regression (based on lambda.1se). The multiplicative factor of $\frac{1}{1+\lambda}$ to a OLS beta to have the corresponding ridge beta in this simple case makes it such that a feature's coefficient from least squares cannot be smaller in magnitude than its coefficient from ridge regression (as $\lambda \geq 0$). The ridge regression coefficient can only be equal to (in the case that $\lambda = 0$) or smaller in magnitude (by a factor of $\frac{1}{1+\lambda}$) than the least squares coefficient for a particular feature.

2.2.3 Lasso regression

i. Fit a 10-fold cross-validated lasso regression to the training data and display the CV plot.

```r
set.seed(5) # set seed before cross-validation for reproducibility

lasso_fit = cv.glmnet(Accept ~ ., alpha = 1, nfolds = 10, data = college_train)
```

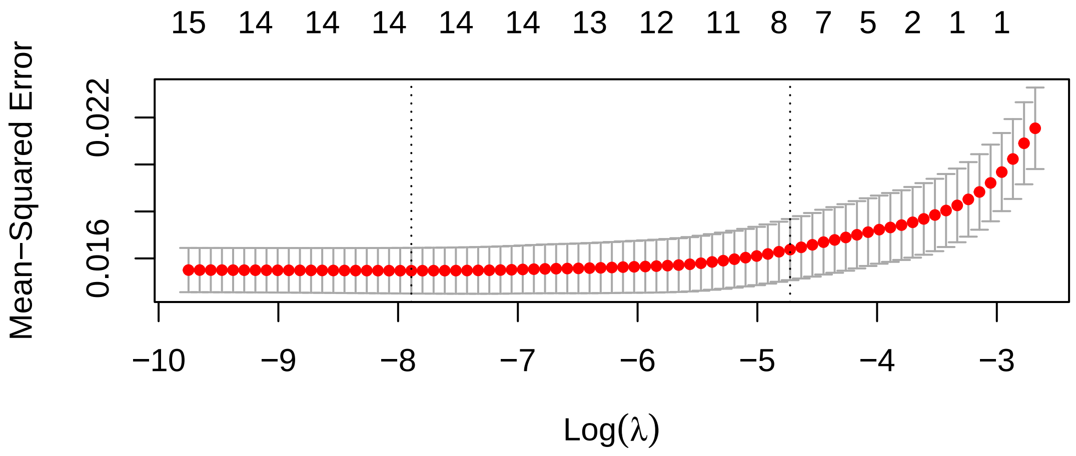

Figure 11: This is the CV plot for the 10-fold cross-validated lasso regression model on the training data.

```r
lambda_lasso = lasso_fit$lambda.1se
sprintf("The value of lambda based on the one-standard-error rule: %f",
        lambda_lasso)
```

```
## [1] "The value of lambda based on the one-standard-error rule: 0.008858"
```

In Figure 11, we have the CV plot for a 10-fold cross-validated lasso regression model to the training data. (We can also note that corresponding to the right vertical dashed line on the plot (on the log scale), the value of lambda selected according to the one-standard-error rule is about 0.009.)

ii. How many features (excluding the intercept) are selected if lambda is chosen according to the one-standard-error rule?

```r
num_features = lasso_fit$nzero[lasso_fit$lambda == lasso_fit$lambda.1se]
sprintf("The number of features (excluding intercept) selected (1se): %i",
        num_features)
```

```
## [1] "The number of features (excluding intercept) selected (1se): 8"
```

Thus, there are 8 features (excluding the intercept) that are selected if lambda is chosen according to the one-standard-error rule.

Although not explicitly asked, we can observe these features with nonzero standardized coefficients in Table 5.

```r
extract_std_coefs(lasso_fit, college_train) %>%
  filter(coefficient != 0) %>%
  kable(format = "latex", row.names = NA,
                         booktabs = TRUE,
                         digits = 6,
                         caption = "These are the nonzero coefficients
        from the lasso model based on a value for lambda determined by
        the one-standard-error rule.") %>%
  kable_styling(position = "center") %>%
  kable_styling(latex_options = "HOLD_position")
```

Table 5: These are the nonzero coefficients from the lasso model based on a value for lambda determined by the one-standard-error rule.

feature	coefficient
Top10perc	-0.049657
P.Undergrad	-0.008180
Room.Board	-0.009512
Books	-0.006903
S.F.Ratio	-0.007198
perc.alumni	0.001520
Expend	-0.015532
Grad.Rate	-0.000289

iii. Use `plot_glmnet` to visualize the lasso fitted coefficients, which by default will highlight the features selected by the lasso. By examining this plot, answer the following questions. Which feature is the first to enter the model as lambda decreases? Which feature has the largest absolute coefficient for the most flexibly fitted lasso model?

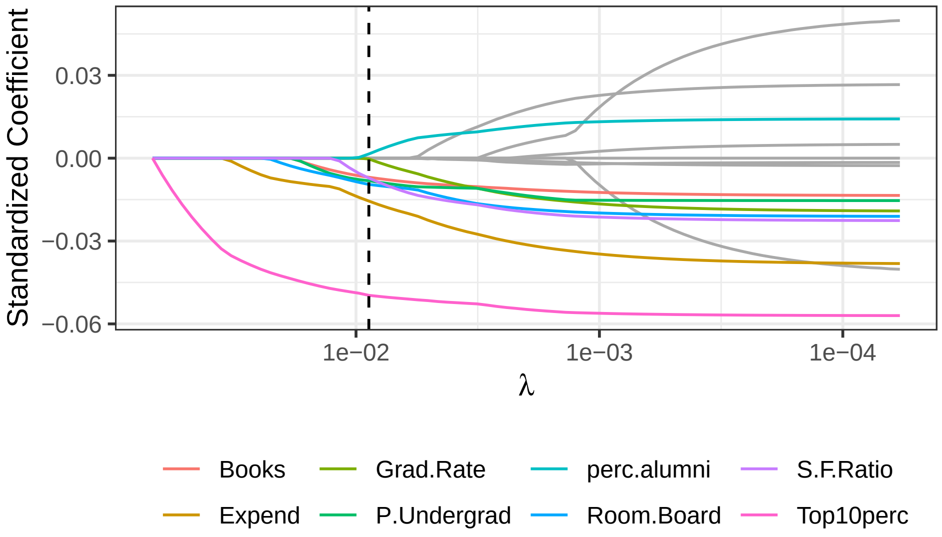

Figure 12: This is the trace plot for the lasso regression model, visualizing the lasso regression fitted coefficients.

Examining this lasso trace plot in Figure 12, we see that lambda decreases from left to right. Thus, Top10perc is the first feature to enter the model as lambda decreases (by examining the far left of the plot and seeing how Top10perc becomes non-zero earliest as we move right). As for the feature that has the largest absolute coefficient for the most flexibly fitted lasso model, we can note that the most flexibly fitted lasso model corresponds to the model with the lowest lambda. Thus, looking at the far right of the plot, we can see that the feature that has the largest absolute coefficient here is Top10perc (since the absolute magnitude of the coefficient is almost 0.06 whereas the most positive coefficient does not reach past 0.045). Thus, the feature that is both the first to enter the model as lambda decreases and the largest absolute coefficient for the most flexibly fitted lasso model is Top10perc, i.e., the percentage of new students in the top 10% of their high school class.

2.2.4 Test set evaluation

i. Calculate the root mean squared test errors of the linear model, ridge regression, and lasso regression (the latter two using lambda.1se) on `college_test`, and print these in a table. Which of the three models has the least test error?

```
# generate linear model predictions and test error
linear_predictions = predict(lm_fit, newdata = college_test)
linear_RMSE = sqrt(mean((linear_predictions - college_test$Accept)^2))

# generate ridge regression model predictions and test error
ridge_predictions = predict(ridge_fit,
                            newdata = college_test,
                            s = "lambda.1se")
ridge_RMSE = sqrt(mean((ridge_predictions - college_test$Accept)^2))

# generate lasso regression model predictions and test error
lasso_predictions = predict(lasso_fit,
                            newdata = college_test,
                            s = "lambda.1se")
lasso_RMSE = sqrt(mean((lasso_predictions - college_test$Accept)^2))
```

```r
# create table of these three model test errors
error_for_models = tribble(
  ~Model, ~RMSE,
  #------/-------
  "Linear", linear_RMSE,
  "Ridge", ridge_RMSE,
  "Lasso", lasso_RMSE,
  )

# print these metrics in nice table
error_for_models %>% kable(format = "latex", row.names = NA,
                        booktabs = TRUE,
                        digits = 5,
                        col.names = c("Model type",
                                      "Root mean squared error"),
                        caption = "These are the root mean squared test
                        errors of the linear model, ridge regression,
                        and lasso regression.") %>%
  kable_styling(position = "center") %>%
  kable_styling(latex_options = "HOLD_position")
```

Table 6: These are the root mean squared test errors of the linear model, ridge regression, and lasso regression.

Model type	Root mean squared error
Linear	0.116
Ridge	0.122
Lasso	0.122

As can be seen in Table 6, which shows the root mean squared test errors of the linear model, ridge regression, and lasso regression (the latter two using lambda.1se) on `college_test`, the linear model RMSE is 0.116, the ridge model RMSE is 0.122, and the lasso model RMSE is 0.122. Thus, of the three models, the linear regression model has the least test error.

ii. Given which model has the lowest test error from part i, as well as the shapes of the CV curves for ridge and lasso, do we suspect that bias or variance is the dominant force in driving the test error in this data? Why do we have this suspicion? Does this suspicion make sense, given the number of features relative to the sample size?

We may suspect that bias (rather than variance) is the dominant force in driving the test error in this data. We can observe from the shape of the CV curves for ridge and lasso that higher values for lambda are contributing to higher mean-squared error. We can see this in the CV plots, as for larger values of lambda, the error simply increases. Thus, it makes sense that the linear model performs best (as lambda = 0), and we are NOT acting to reduce the variance, trading variance off for higher bias, as we are with ridge and lasso. (Also, in the lasso CV plot, we can see that we are not improving the model by shrinking and selecting a smaller subset of features, as there does not appear a point where this increase in bias is enabling us to have a lower test error based on the reduction in variance.) Least squares has the highest variance and is performing best, so it would make sense that bias is the dominant force in driving the test error in the data (i.e., bias is the bigger issue). Furthermore, the sample size n is fairly large relative to the number of features, so this suspicion makes sense, for as n is larger, we worry less about variance (as variance is inversely proportional to the sample size). Thus, given that the ridge and lasso models perform worst, it makes sense to suspect that bias is a bigger issue relative to variance. Furthermore, this makes sense, as the ratio of sample size to

features (certainly more than ten times larger) is big enough so that variance is not as much of an issue.