Homework 1

Jerson R. Cochancela

February 28, 2019

Book "An Introduction to Statistical Learning" by James G, Witten D, Hastie T, and Tibshirani (JWHT)

JWHT. Chapter 3. Question 4.

I collect a set of data (n = 100 observations) containing a single predictor and a quantitative response. I then fit a linear regression model to the data, as well as a separate cubic regression, i.e. $Y = \beta_0 + \beta_1 X + \beta_2 X^2 + \beta^3 X^3 + \epsilon$.

(a) Suppose that the true relationship between X and Y is linear, i.e.$Y = \beta_0 + \beta_1 X + \epsilon$. Consider the training residual sum of squares (RSS) for the linear regression, and also the training RSS for the cubic regression. Would we expect one to be lower than the other, would we expect them to be the same, or is there not enough information to tell? Justify your answer.

We would expect the RSS for training in the cubic setting to be smaller than the training RSS in the linear setting. We can see this to be the case as the noise introduced by ϵ to be overfit by the cubic regression, leading to smaller RSS.

(b) Answer (a) using test rather than training RSS.

Now using the test data, comparing the RSS of the linear vs. the cubic regression, we would expect the linear regression to have a smaller RSS on than the cubic. We can see this as a result of the overfitting done in the cubic setting (knowing that the true relationship is indeed linear). The cubic regression will have trouble fitting the noise whereas, on average, the linear setting will capture the relationship better with smaller RSS.

(c) Suppose that the true relationship between X and Y is not linear, but we don't know how far it is from linear. Consider the training RSS for the linear regression, and also the training RSS for the cubic regression. Would we expect one to be lower than the other, would we expect them to be the same, or is there not enough information to tell? Justify your answer.

My intuition is that, in the case where we don't know the true relation **but it is not linear**, then the cubic RSS will be smaller than the linear RSS. The idea here is that the linear model will not capture the curvature of the data as well as a cubic model may.

(d) Answer (c) using test rather than training RSS.

We do not have enough information here given the infinitely many values the beta's can take.

JWHT. Chapter 3. Question 13 (a)-(g).

In this exercise you will create some simulated data and will fit simple linear regression models to it. Make sure to use set.seed(1) prior to starting part (a) to ensure consistent results.

```r
# We set the seed as the problem asks.
set.seed(1)
```

(a) Using the rnorm() function, create a vector, x, containing 100 observations drawn from a $N(0, 1)$ distribution. This represents a feature, X.

```r
# Drawing 100 psuedo RV from N(0, 1).
x <- rnorm(100)
```

(b) Using the rnorm() function, create a vector, eps, containing 100 observations drawn from a $N(0, 0.25)$ distribution i.e. a normal distribution with mean zero and variance 0.25.

```r
# Drawing 100 psuedo RS from N(0, 0.5^2).
eps <- rnorm(100, mean = 0, sd = 0.5)
```

(c) Using x and eps, generate a vector y according to the model

$$Y = -1 + 0.5X + \epsilon.$$

What is the length of the vector y? What are the values of β_0 and β_1 in this linear model?

Before we generate y, we can see that y will have he same dimensions as x and eps. As both are 100×1 vectors, then their transformation according to the model above will results in a vector $\mathbf{y}_{100\times1}$. Additionally, we can see that the intercept is -1, i.e. $\beta_0 = -1$ and the slope is $1/2$, i.e. $\beta_1 = 0.5$.

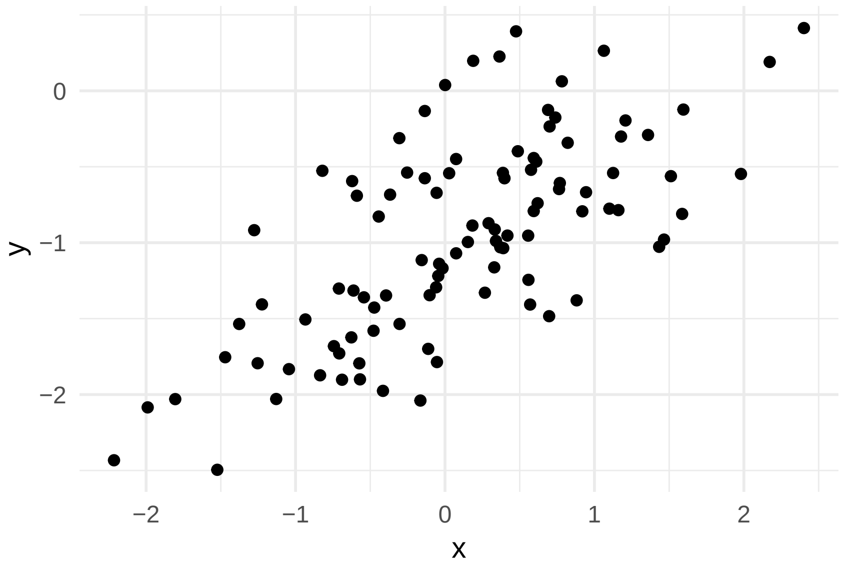
```r
# Build y according to the model.
y <- -1 + 0.5*x + eps
```

Below, we confirm the dimensions of y.

```r
length(y)
```

```
## [1] 100
```

(d) Create a scatterplot displaying the relationship between x andy. Comment on what you observe.



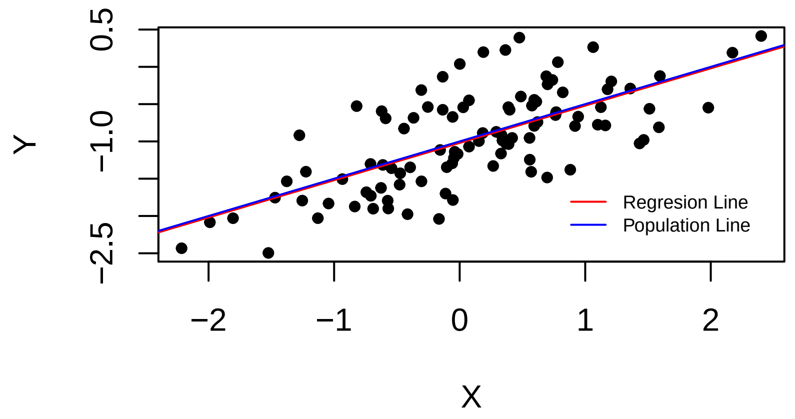
(e) Fit a least squares linear model to predict y using x. Comment on the model obtained. How do $\hat{\beta}_0$ and $\hat{\beta}_1$ compare to β_0 and β_1?

Table 1: Estimated Coefficients

	Estimate	Std. Error	t value	Pr(>|t|)
(Intercept)	-1.02	0.05	-21.01	<0.0001
x	0.5	0.05	9.27	<0.0001

Our estimated values are nearly perfect given the true values generating the data. We see that our slope is on target and that a 95% confidence interval captures the intercept.

(f) Display the least squares line on the scatterplot obtained in (d). Draw the population regression line on the plot, in a different color. Use the legend() command to create an appropriate legend.



(g) Now fit a polynomial regression model that predicts y using x and x2. Is there evidence that the quadratic term improves the model fit? Explain your answer.

```r
# Fit a quadratic model.
quad_fit <- lm(y ~ poly(x, 2, raw = TRUE), data = df)
```

Below we see that the quadratic term is not statistically significant.

```r
summary(quad_fit)$coefficients
```

```
##                           Estimate Std. Error    t value     Pr(>|t|)
## (Intercept)             -0.9716425 0.05882775 -16.516738 6.079012e-30
## poly(x, 2, raw = TRUE)1  0.5085804 0.05399135   9.419666 2.403287e-15
## poly(x, 2, raw = TRUE)2 -0.0594606 0.04238285  -1.402940 1.638275e-01
```

To further test this we can also compare our linear model to our quadratic model since they are nested.

```r
anova(lm_fit, quad_fit)
```

```
## Analysis of Variance Table
##
## Model 1: y ~ x
## Model 2: y ~ poly(x, 2, raw = TRUE)
##   Res.Df    RSS Df Sum of Sq      F Pr(>F)
## 1     98 22.709
## 2     97 22.257  1   0.45163 1.9682 0.1638
```

We see that we fail to reject the null hypothesis that the coefficient of the quadratic term is zero. We conclude as before that there is no evidence that the quadratic term improves the model fit.

JWHT. Chapter 3. Question 14.

This problem focuses on the collinearity problem.

(a) Perform the following commands in R:

```r
# Following the book.
set.seed(1)

# Drawing 100 PRV from a U(0, 1).
x1 <- runif(100)

# Creating x2 which follows a model:
# Y = 0.5*x1 + N(0,1)/10
x2 <- 0.5 * x1 + rnorm (100) / 10

# Builing y according to the model below.
y <- 2 + 2 * x1 + 0.3 * x2 + rnorm(100)
```

The last line corresponds to creating a linear model in which y is a function of x1 and x2. Write out the form of the linear model. What are the regression coefficients?

$$Y = 2 + 2X_1 + 0.3X_2 + \epsilon$$

where $\beta_0 = 2$, $\beta_1 = 2$ and $\beta_3 = 0.3$. We note that $\epsilon \sim N_(0, 1)$ for 100 draws.

(b) What is the correlation between x1 and x2? Create a scatterplot displaying the relationship between the variables.

Below we see that the correlation between X_1 and X_2 is about 0.85.

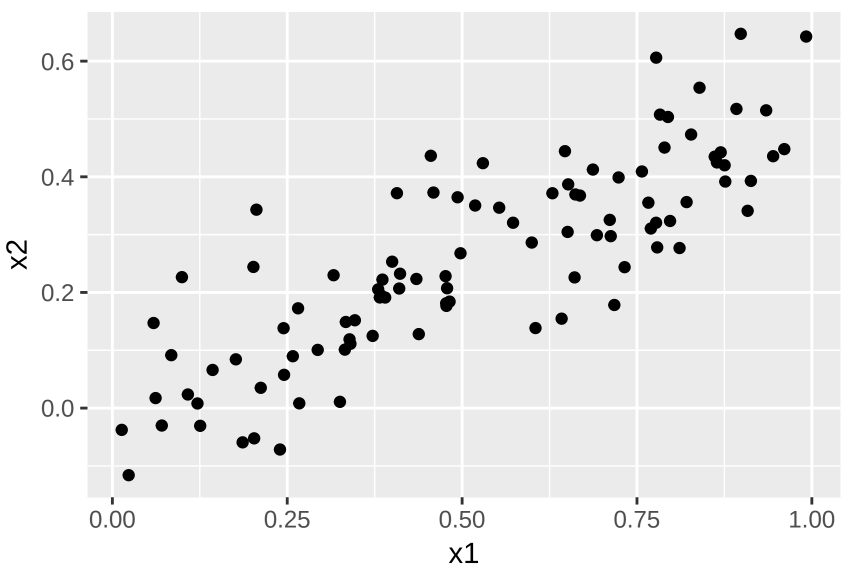
```r
cor(x1, x2)
```

```
## [1] 0.8351212
```

This is no surprise to us as X_2 is linearly related, by construction:

$$X_2 = 0.5X_1 + \epsilon_{x_1}$$

where ϵ_{x_1} are 100 pseudorandomn values(PSV) from a $N(0, 1)/10$, where we have abused the notation but denote the 100 PRV from the standard normal have been divided by 10.



(c) Using this data, fit a least squares regression to predict y using x1 and x2. Describe the results obtained. What are $\hat{\beta}_0$, $\hat{\beta}_1$, and $\hat{\beta}_2$?

First we fit the linear model on x1 and x2.

```r
# The linear model on x1 and x2.
fit <- lm(y ~ x1 + x2, data = df2)
```

How do these relate to the true β_0, β_1, and β_2?

We know the true values are $\beta_0 = 2$, $\beta_1 = 2$ and $\beta_3 = 0.3$. We see that our model is having the most trouble with X_2, and with 95% confidence, we capture the intercept and β_1.

Table 2: Estimated Coefficients

	Estimate	Std. Error	t value	Pr(>|t|)
(Intercept)	2.13	0.23	9.19	<0.0001
x1	1.44	0.72	2	0.05
x2	1.01	1.13	0.89	0.38

Can you reject the null hypothesis $H_0 : \beta_1 = 0$? How about the null hypothesis $H_0 : \beta_2 = 0$?

Since the p-vales for X_1 and X_2 related to a wald test, we know that at a 5% significance level, we would fail to reject the null hypotheses that these beta's are zero. Obviously, a less conservative signifiance level like 10% would reject for β_1 but not for β_2.

(d) Now fit a least squares regression to predict y using only x1. Comment on your results. Can you reject the null hypothesis $H_0 : \beta_1 = 0$?

```r
# Fitting model with only x1.
fit_x1 <- lm(y ~ x1, data = df2)
```

Table 3: Estimated Coefficients

	Estimate	Std. Error	t value	Pr(>|t|)
(Intercept)	2.11	0.23	9.15	<0.0001
x1	1.98	0.4	4.99	<0.0001

Please see part (f) for the derivation, but we know that

$$Y = 2 + 2.15X_1 + \epsilon_{\text{new}}$$

. Using a 5% significance level, we fail to reject the null hypothesis that $\beta_1 = 0$.

(e) Now fit a least squares regression to predict y using only x2. Comment on your results. Can you reject the null hypothesis $H_0 : \beta_1 = 0$?

```r
# Fit model only on x2.
fit_x2 <- lm(y ~ x2, data = df2)
```

Table 4: Estimated Coefficients

	Estimate	Std. Error	t value	Pr(>|t|)
(Intercept)	2.39	0.19	12.26	<0.0001
x2	2.9	0.63	4.58	<0.0001

See part (f) for a derivation. Note that we can rewrite the model as

$$Y = 2 + 4.3X_2 + \epsilon_{\text{new}_2}$$

In this light, we can see why we fail to reject the null hypothsis at the 5% significance level.

(f) Do the results obtained in (c)-(e) contradict each other? Explain your answer.

Yes and no. Yes, (c) contradicts (d) and (e), but (d) does not contradict (e). We can see why below.

Recall:

$$Y = 2 + 2X_1 + 0.3X_2 + \epsilon$$

and

$$X_2 = 0.5X_1 + \epsilon_{x_1}$$

then, we know that:

$$Y = 2 + 2X_1 + 0.3(0.5X_1 + \epsilon_{x_1}) + \epsilon = 2 + 2.15X_1 + \epsilon_{\text{new}}$$

Alternatively,

$$X_2 = 0.5X_1 + \epsilon_{x_1} \Rightarrow X_1 = 2(X_2 - \epsilon_{x_1})$$

and in this light we can see that Y can be written as a function soley of X_2.

$$Y = 2 + 2(2(X_2 - \epsilon_{x_1})) + 0.3X_2 + \epsilon = 2 + 4.3X_2 + \epsilon_{\text{new}_2}$$

If we examine the estimated coefficients for X_1 in (d), we see we are very close with our 1.98 estimate. The estimated coefficient for X_2 isn't exactly on the nail but within a 95% confidence interval.

(g) Now suppose we obtain one additional observation, which was unfortunately mismeasured.

```r
x1 <- c(x1 , 0.1)
x2 <- c(x2 , 0.8)
y <- c(y,6)

# Save them for later use.
df3 <- data.frame(cbind(x1, x2, y))
```

Re-fit the linear models from (c) to (e) using this new data.

What effect does this new observation have on the each of the models?

As we can see below, linear models are pretty robust. The mismeasured value has changed the estimates but overall, their confidence intervals still capture the true beta's.

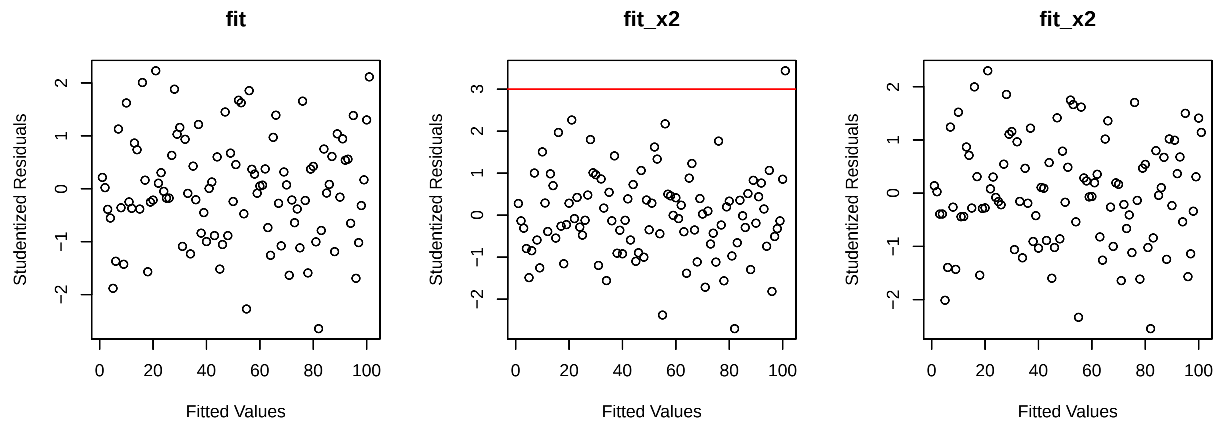
Table 5: Estimated Coefficients

	Estimate	Std. Error	t value	Pr(>|t|)
Full Model				
(Intercept)	2.23	0.23	9.62	<0.0001
x1	0.54	0.59	0.91	0.36
x2	2.51	0.9	2.8	0.01
X__1 Model				
(Intercept)	2.26	0.24	9.44	<0.0001
x1	1.77	0.41	4.28	<0.0001
X__2 Model				
(Intercept)	2.35	0.19	12.26	<0.0001
x2	3.12	0.6	5.16	<0.0001

In each model, is this observation an outlier? A high-leverage point? Both? Explain your answers.

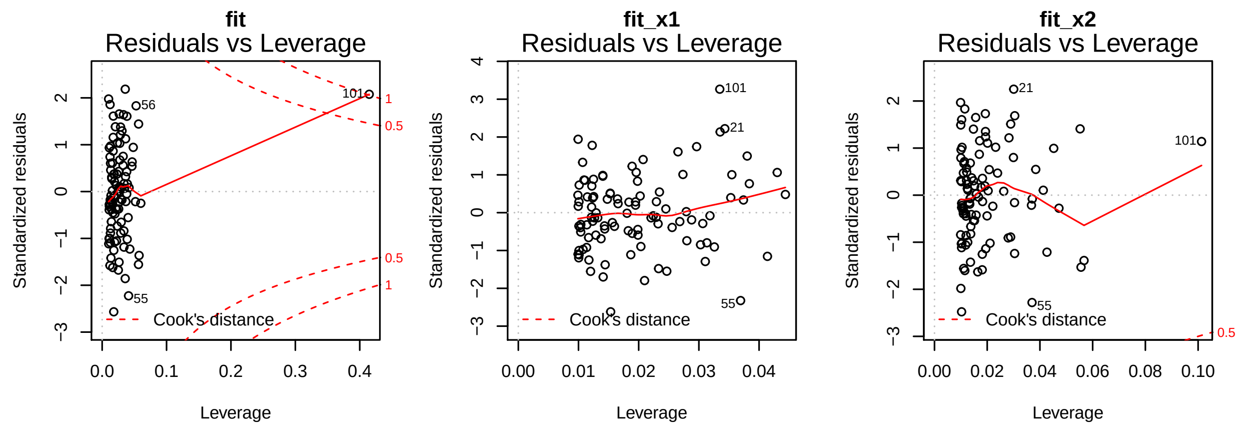
Before identifying the mismeasured value as an outlier and/or leverage, we discuss briefly how we will examine and determine the two.

An outlier may be found by examining the studentized residuals. An observation in absolute greater than 3 will be considered a possible outlier.



Above we see that in fit_x1, our model that only includes x_1 as a predictor, the newly added and mismeasured data point can be considered an outlier.

We examine the Cook's Distance plots for leverage point detection.



Above we see that the 101th data point is considered a leverage point in the first model which includes x_1 and x_2 but not for the remaining models.

JWHT. Chapter 6. Question 5.

It is well-known that ridge regression tends to give similar coefficient values to correlated variables, whereas the lasso may give quite different coefficient values to correlated variables. We will now explore this property in a very simple setting.

Suppose that $n = 2$, $p = 2$, $x_{11} = x_{12}$, $x_{21} = x_{22}$. Furthermore, suppose that $y_1 + y_2 = 0$ and $x_{11} + x_{21} = 0$ and $x_{12} + x_{22} = 0$, so that the estimate for the intercept in a least squares, ridge regression, or lasso model is zero: $\hat{\beta}_0 = 0$.

Let

$$\boldsymbol{Y}_{2\times 1} = [y_1, y_2]^T$$

,

$$\boldsymbol{X}_{2\times 2} = \begin{bmatrix} x_{11} & x_{12} \\ x_{21} & x_{22} \end{bmatrix}$$

Given the first set of conditions (1): $x_{11} = x_{12}$, $x_{21} = x_{22}$, we rewrite $\boldsymbol{X}_{2\times 2}$:

$$\boldsymbol{X}_{2\times 2} = \begin{bmatrix} x_{11} & x_{11} \\ x_{22} & x_{22} \end{bmatrix}$$

Furthermore, given that (2) $y_1 + y_2 = 0$ and $x_{11} + x_{21} = 0$ and $x_{12} + x_{22} = 0$, we rewrite:

$$\boldsymbol{X}_{2\times 2} = \begin{bmatrix} x_{11} & x_{12} \\ -x_{11} & -x_{12} \end{bmatrix}$$

(a) Write out the ridge regression optimization problem in this setting.

Let $\boldsymbol{\beta} = [\beta_1, \beta_2]^T$, and $\boldsymbol{x}_i^T$ the i^{th} row of the design matrix $\boldsymbol{X}_{2\times 2}$ then the ridge regression in this setting is in the form:

$$\sum_i \left(y_i - \boldsymbol{x_i}^T\boldsymbol{\beta}\right)^2 + \lambda \sum_j \beta_j^2 = \sum_{i=1}^{2} \left(y_i - \boldsymbol{x_i}^T\boldsymbol{\beta}\right)^2 + \lambda \sum_{j=1}^{2} \beta_j^2$$

(b) Argue that in this setting, the ridge coefficient estimates satisfy $\hat{\beta}_1 = \hat{\beta}_2$.

We know that the estimates will be equal by the first set of conditions set in the problem. Note that the design matrix, without intercept, would be of the form:

$$\boldsymbol{X}_{2\times 2} = \begin{bmatrix} x_{11} & x_{12} \\ x_{21} & x_{22} \end{bmatrix} = \begin{bmatrix} x_{11} & x_{11} \\ x_{21} & x_{21} \end{bmatrix}$$

Clearly, given the same values for each β_i, we see the estimates should be equal for $\hat{\beta}_1 = \hat{\beta}_2$.

(c) Write out the lasso optimization problem in this setting.

Following our notation above, the LASSO problem in this setting is of the form:

$$\sum_i \left(y_i - \boldsymbol{x_i}^T \boldsymbol{\beta}\right)^2 + \lambda \sum_j |\beta_j| = \sum_{i=1}^{2} \left(y_i - \boldsymbol{x_i}^T \boldsymbol{\beta}\right)^2 + \lambda \sum_{j=1}^{2} |\beta_j|$$

(d) Argue that in this setting, the lasso coefficients $\hat{\beta}_1$ and $\hat{\beta}_2$ are not unique-in other words, there are many possible solutions to the optimization problem in (c). Describe these solutions.

Take note of the design matrix after applying (2) the second set of conditions in this problem. In this setting, we have reduced the row space.

JWHT. Chapter 6. Question 9 (a)-(d).

In this exercise, we will predict the number of applications received using the other variables in the College data set.

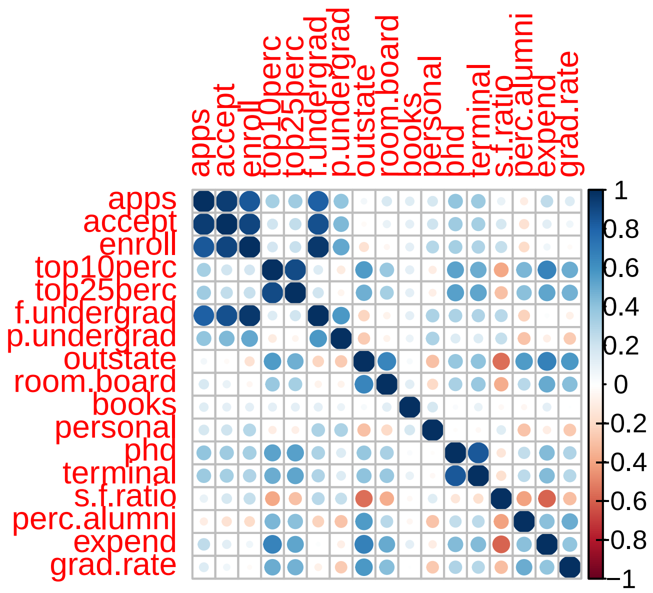
```r
# Bring in the data.
df <- College

# Clean the variable names.
names(df) <- tolower(names(df))
```

(a) Split the data set into a training set and a test set.

```r
set.seed(1)

# We will split the data 80-20 train vs test.
in_training <- createDataPartition(df$apps, p = .80,
                                   list = F, times = 1)
# Split into training.
train <- df[in_training, ]

# Split into testing.
test <- df[-in_training, ]
```

(b) Fit a linear model using least squares on the training set, and report the test error obtained.

We will fit a simple model based on correlated/linear predictors of the outcome of interest: apps. This is done so we have satisfied the assumptions of linearity in the application of a linear model.



Therefore, we would typically fit a simple model that regresses the number of applications a college receives on the number of applications (apps) accepted (accept), the number of students enrolled (enroll), and the number of fulltime undergraduates (f.undergrad).

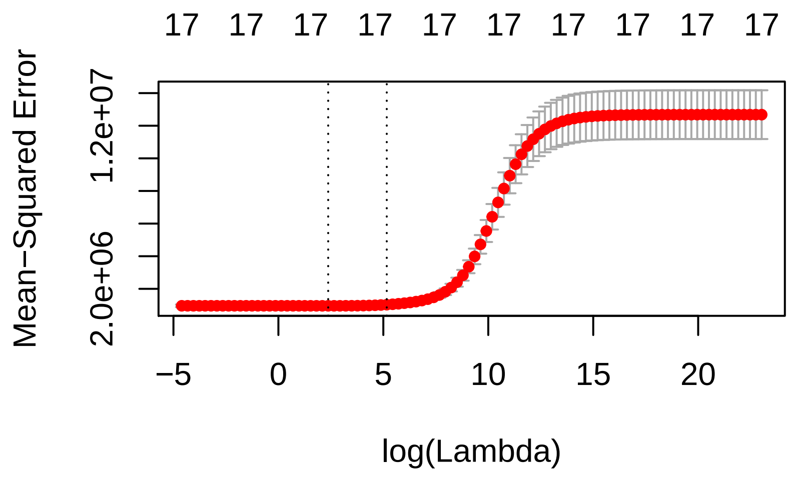
For the purposes of this exercise, we will fit our outcome of apps on all variables in the data. Our hope is that we should see most covariates penalized to zero, and especially this sort of variable selection within LASSO.

```
## [1] 2279815
```

Our mean squared error on the training data is 2279815; not great. Of course, we could have included all the other variables and reduced our MSE but we would have wantonly included covariates without minimally examining that the assumptions of the model are checked.

(c) Fit a ridge regression model on the training set, with λ chosen by cross-validation. Report the test error obtained.

To continue consistency, we perform a 5-fold cross validation as our training and test were split 80-20. We apply a grid search within the cv.glmnet() function. Below is a plot of our lambda values on the log scale.



Our lambda that minimizes our MSE is ~19.

```r
lambda_cv$lambda.min
```

```
## [1] 10.72267
```

We will use one standard error away which is ~705.

```r
lambda_cv$lambda.1se
```

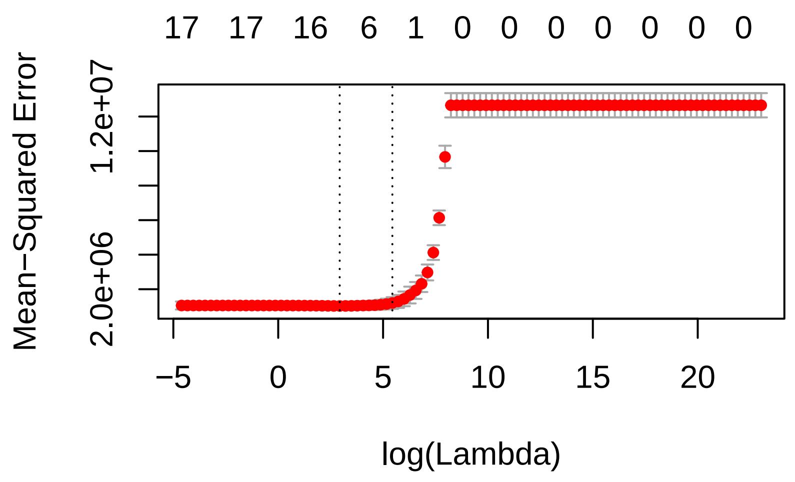
```
## [1] 174.7528
```

Below we see that our ridge model has increased the MSE relative to our simple linear model and is performing worse. In fact, the MSE has increased by 3179214

```r
ridge <- glmnet(design_matrix,
                train$apps, alpha = 0, lambda = lambda_cv$lambda.1se)

test_design_matrix <- model.matrix(apps ~ ., data = test)

test_design_matrix <- test_design_matrix[ , -1]

ridge.pred <- predict.glmnet(ridge, newx = test_design_matrix)

mean((test$apps-ridge.pred)^2)
```

```
## [1] 3758076
```

(d) Fit a lasso model on the training set, with λ chosen by crossvalidation. Report the test error obtained, along with the number of non-zero coefficient estimates.



Our lambda that minimizes our MSE is ~19 in the LASSO setting. We will use this in our LASSO model. We will use one standard error of about 231 for our lambda to be conservative.

```
lambda_cv$lambda.min
```

```
## [1] 18.73817
```

```
lambda_cv$lambda.1se
```

```
## [1] 231.013
```

Below we see that our lasso model has a worse MSE than our simple linear model. In fact, the MSE has increased by 763034.30.

```
lasso <- glmnet(design_matrix,
                train$apps, alpha = 1, lambda = lambda_cv$lambda.1se)

lasso_pred <- predict.glmnet(lasso, newx = test_design_matrix)

mean((test$apps-lasso_pred)^2)
```

```
## [1] 3042849
```

We see that LASSO has penalized to zero the variables enroll, top25perc, books, personal.Essentially, we have whittled away 4 of the 18 variables. These results are not surprising as we left all variables in the model without descerning whether they should even be tested! Garbage in, garbage out.

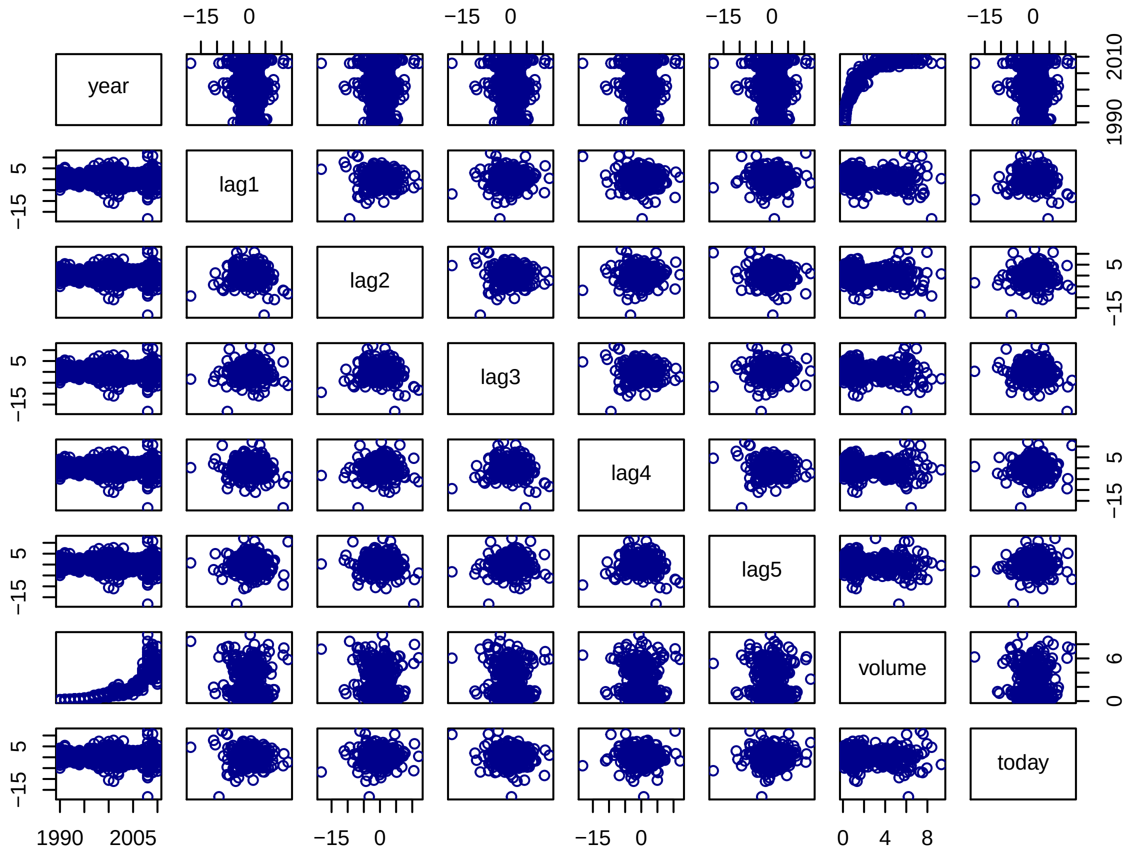
JWHT. Chapter 4. Question 10 (a)-(d), and (i). For (i), consider using model regularizations for model selection and for now, ignore the last sentence about the experiment with KNN classifier.

This question should be answered using the Weekly data set, which is part of the ISLR package. This data is similar in nature to the Smarket data from this chapter's lab, except that it contains 1,089 weekly returns for 21 years, from the beginning of 1990 to the end of 2010.

```r
# Load in the ISLR Weekly data.
df <- Weekly

# Fix names for ease.
names(df) <- names(df) %>% tolower()
```

(a) Produce some numerical and graphical summaries of the Weekly data. Do there appear to be any patterns?

Since `direction` is a recoding of today, we examine today. Below we note that there seems to be no linear relationships between the variables and `today`.

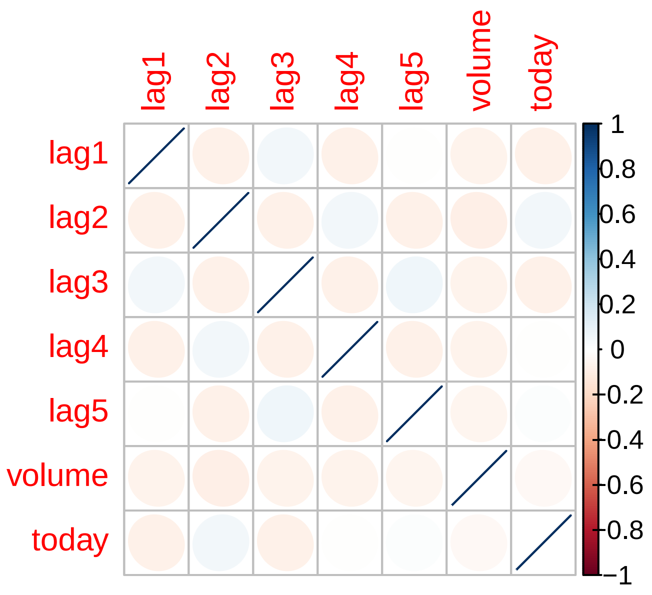


Additonally, we note a relationship, possibly logistic, between `year` and `volume`, suggesting collinearity between our predictors. We can verify this simply by examining the VIF statistics.

Table 6: Variance Inflation Factor

Variables	VIF
year	3.484
lag1	1.026
lag2	1.036
lag3	1.030
lag4	1.025
lag5	1.020
volume	3.560
today	1.015

Indeed, we see that there is a relationship between `year` and `volume`. Below, we find that that `lag1` and `lag3` are positively correlated with `today` while `lag2` is negatively correlated with the continuous outcome.



(b) Use the full data set to perform a logistic regression with Direction as the response and the five lag variables plus Volume as predictors. Use the summary function to print the results. Do any of the predictors appear to be statistically significant? If so, which ones?

Below, at the 5%, we can see that lag2 is statistically significant .

(c) Compute the confusion matrix and overall fraction of correct predictions. Explain what the confusion matrix is telling you about the types of mistakes made by logistic regression.

```
##           Reference
## Prediction Down  Up
##       Down   54  48
##       Up    430 557
```

Table 7: Estimated Coefficients

	Estimate	Std. Error	z value	Pr(>|z|)
(Intercept)	0.27	0.09	3.11	<0.001
lag1	-0.04	0.03	-1.56	0.12
lag2	0.06	0.03	2.18	0.03
lag3	-0.02	0.03	-0.6	0.55
lag4	-0.03	0.03	-1.05	0.29
lag5	-0.01	0.03	-0.55	0.58
volume	-0.02	0.04	-0.62	0.54

We note that the accuracy is around 56% which is not much better than naive guessing. Typically, the setting of the study would help us determine what we would want to optimize. This meaning, are we interested in increasing our sensitivity or specificity In this case, as we are trying to predict the direction of the stock market, we would want to increase both our values in the diagonal of the confusion matrix. We note that the sensitivity is poor (~11%), and so logistic regression is having trouble predicting the markets Down direction.

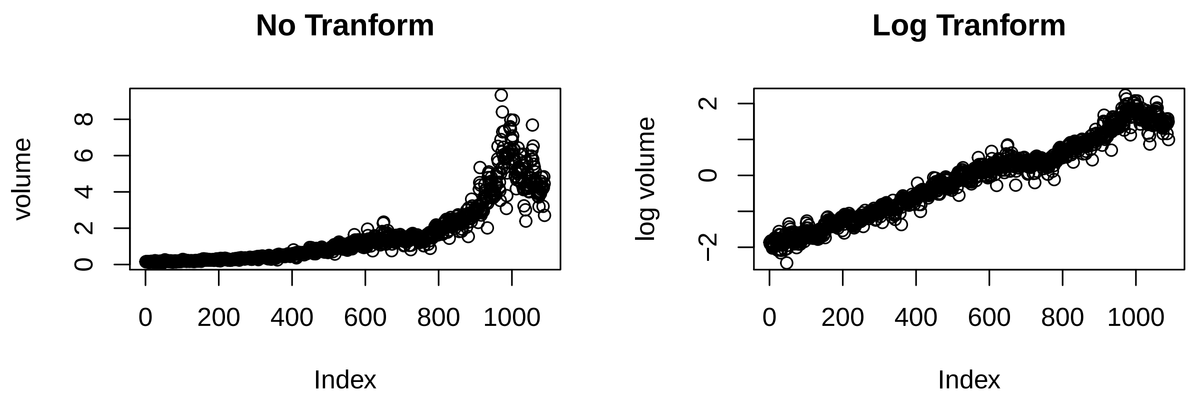
(d) Now fit the logistic regression model using a training data period from 1990 to 2008, with Lag2 as the only predictor. Compute the confusion matrix and the overall fraction of correct predictions for the held out data (that is, the data from 2009 and 2010).

```
##             Reference
## Prediction Down Up
##       Down    9  5
##       Up     34 56
```

We see that on the held out data, the test, we have better accuracy at 62.5%. While specificity has remained about the same (~92%), sensitivity has almost doubled to ~21%. The Positive Predicted Value is 64.28% and the Negative Predicted Value is 62.22%.

(i) Experiment with different combinations of predictors, including possible transformations and interactions, for each of the methods. Report the variables, method, and associated confusion matrix that appears to provide the best results on the held out data.

We ran 5-fold cross validation on the training data. Intermittently, we performed forward and backward selection to perform variable selection on a set of interactions and a log tranformation on `volume` as we noted earlier.



The final model selected by `caret` includes `lag1`, `lag2` and `lag5:log(volume)`.

```
glm_fit$finalModel
```

```
##
## Call:  NULL
##
## Coefficients:
##           (Intercept)                   lag1                   lag2
##               0.21984               -0.05601                0.05737
## `lag5:I(log(volume))`
##               0.04457
##
## Degrees of Freedom: 984 Total (i.e. Null);  981 Residual
## Null Deviance:       1355
## Residual Deviance: 1344  AIC: 1352
```

On the test data, we have an AUC of 0.57. We also find that the optimal threshold is ~53%.

```
## Area under the curve: 0.5703
```

```
##   threshold specificity       1-npv
##   0.5293889   0.4418605   0.4062500
```

Below, we have confusion matrix of the model selected. Because `Down` is the direction we are predicting in our model, we note that the specificity of this model is good at ~79% but the sensitivity is poor at around 44%. Our accuracy is around 64%.

```
##           Reference
## Prediction Down Up
##       Down   19 13
##       Up     24 48
```

Code Appendix

```
## ---- echo = FALSE-------------------------------------------------------
# Libraries.
pacman::p_load(dplyr, kableExtra, knitr, ggplot2,
               ISLR, corrplot, caret, glmnet, MASS,
               usdm, leaps, pROC)

## ------------------------------------------------------------------------
# We set the seed as the problem asks.
set.seed(1)

## ------------------------------------------------------------------------
# Drawing 100 psuedo RV from N(0, 1).
x <- rnorm(100)

## ------------------------------------------------------------------------
# Drawing 100 psuedo RS from N(0, 0.5^2).
eps <- rnorm(100, mean = 0, sd = 0.5)

## ------------------------------------------------------------------------
# Build y according to the model.
y <- -1 + 0.5*x + eps

## ------------------------------------------------------------------------
length(y)

## ---- echo = FALSE, fig.align = "center", fig.asp = .68, fig.width=4.5, warning = FALSE----
# Bind the data for ease in ggplot.
df <- data.frame(cbind(y, x))

# Scatterplot with ggplot.
ggplot(df, aes(x, y)) +
  geom_point() +
  theme_minimal()

## ---- echo = FALSE-------------------------------------------------------
# Now we fit a model to the data.
lm_fit <- lm(y ~ x, data = df)

# Let's examine the estimated intercept and slope.
coefs <- round(summary(lm_fit)$coefficients, 2)

# Clean up p-vals.
coefs[,4] <- "<0.0001"

# Make table.
kable(coefs, caption = "Estimated Coefficients",
      booktabs = TRUE, format = "latex") %>%
  kable_styling(latex_options=c("hold_position"))

## ---- echo = FALSE, fig.align = "center", fig.asp = .68, fig.width=4.5, warning = FALSE----
```

```
# Scatterplot with ggplot.
plot(df$x, df$y, pch  = 20, xlab = "X", ylab = "Y")
abline(a = as.numeric(coefs[1, 1]),
       b = as.numeric(coefs[2,1]), col="red")
abline(a =-1, b = 0.5, col="blue")
legend(x = 0.75, y =-1.5, legend = c("Regresion Line", "Population Line"), lty=c(1,1), col = c("red", "
       box.lty=0)

## ------------------------------------------------------------------------
# Fit a quadratic model.
quad_fit <- lm(y ~ poly(x, 2, raw = TRUE), data = df)

## ------------------------------------------------------------------------
summary(quad_fit)$coefficients

## ------------------------------------------------------------------------
anova(lm_fit, quad_fit)

## ------------------------------------------------------------------------
# Following the book.
set.seed(1)

# Drawing 100 PRV from a U(0, 1).
x1 <- runif(100)

# Creating x2 which follows a model:
# Y = 0.5*x1 + N(0,1)/10
x2 <- 0.5 * x1 + rnorm (100) / 10

# Builing y according to the model below.
y <- 2 + 2 * x1 + 0.3 * x2 + rnorm(100)

## ------------------------------------------------------------------------
cor(x1, x2)

## ---- echo = FALSE, fig.align = "center", fig.asp = .68, fig.width=4.5, warning = FALSE----
# Bind the new data.
df2 <- data.frame(cbind(x1, x2, y))

# Scatterplot with ggplot.
ggplot(df2, aes(x1, x2)) +
  geom_point()


## ------------------------------------------------------------------------
# The linear model on x1 and x2.
fit <- lm(y ~ x1 + x2, data = df2)

## ---- echo = FALSE-------------------------------------------------------
# Let's examine the estimated intercept and slope.
coefs <- round(summary(fit)$coefficients, 2)

# Clean up p-vals.
```

```r
coefs[1,4] <- "<0.0001"

# Make table.
kable(coefs, caption = "Estimated Coefficients",
      booktabs = TRUE, format = "latex") %>%
  kable_styling(latex_options=c("hold_position"))

## ------------------------------------------------------------------------
# Fitting model with only x1.
fit_x1 <- lm(y ~ x1, data = df2)

## ---- echo = FALSE-------------------------------------------------------
# Let's examine the estimated intercept and slope.
coefs <- round(summary(fit_x1)$coefficients, 2)

# Clean up p-vals.
coefs[,4] <- "<0.0001"

# Make table.
kable(coefs, caption = "Estimated Coefficients",
      booktabs = TRUE, format = "latex") %>%
  kable_styling(latex_options=c("hold_position"))

## ------------------------------------------------------------------------
# Fit model only on x2.
fit_x2 <- lm(y ~ x2, data = df2)

## ---- echo = FALSE-------------------------------------------------------
# Let's examine the estimated intercept and slope.
coefs <- round(summary(fit_x2)$coefficients, 2)

# Clean up p-vals.
coefs[,4] <- "<0.0001"

# Make table.
kable(coefs, caption = "Estimated Coefficients",
      booktabs = TRUE, format = "latex") %>%
  kable_styling(latex_options=c("hold_position"))

# Save the three fits for later comparison.
fits_100 <- list(fit, fit_x1, fit_x2)

## ------------------------------------------------------------------------
x1 <- c(x1 , 0.1)
x2 <- c(x2 , 0.8)
y <- c(y,6)

# Save them for later use.
df3 <- data.frame(cbind(x1, x2, y))

## ---- echo = FALSE-------------------------------------------------------
# Refitting the three models.
fit <- lm(y ~ x1 + x2, data = df3)
```

```
fit_x1 <- lm(y ~ x1, data = df3)

fit_x2 <- lm(y ~ x2, data = df3)

# Save for later use.
fits_101 <- list("fit" = fit, "fit_x2" = fit_x1, "fit_x2" = fit_x2)

## ---- echo = FALSE--------------------------------------------------------
# Let's examine the estimated intercept and slope.
coefs <- round(summary(fit)$coefficients, 2)
coefs_x1 <- round(summary(fit_x1)$coefficients, 2)
coefs_x2 <- round(summary(fit_x2)$coefficients, 2)

all_coefs <- rbind(coefs, coefs_x1, coefs_x2)

# Clean up p-vals.
all_coefs[-c(2,3),4] <- "<0.0001"

# Make table.
kable(all_coefs, caption = "Estimated Coefficients",
      booktabs = TRUE, format = "latex") %>%
  kable_styling(latex_options=c("hold_position")) %>%
  group_rows("Full Model", 1, 3) %>%
  group_rows("X_1 Model", 4, 5) %>%
  group_rows("X_2 Model", 6, 7)

## ---- echo = FALSE, fig.width=8, fig.height=3----------------------------
par(mfrow=c(1 ,3))

for(i in 1:3){
    plot(studres(fits_101[[i]]), main = names(fits_101)[i],
         ylab = "Studentized Residuals",
         xlab = "Fitted Values") +
    abline(h = 3, col = "red") +
    abline(h = -3, col = "red")

  }
par(mfrow=c(1,3))


## ---- echo = FALSE, fig.width=8, fig.height=3----------------------------
par(mfrow=c(1,3))
plot(fit, 5, main = "fit")
plot(fit_x1, 5, main ="fit_x1")
plot(fit_x2, 5, main ="fit_x2")
par(mfrow=c(1,3))

## ------------------------------------------------------------------------
# Bring in the data.
df <- College

# Clean the variable names.
names(df) <- tolower(names(df))
```

```r
## --------------------------------------------------------------------------
set.seed(1)

# We will split the data 80-20 train vs test.
in_training <- createDataPartition(df$apps, p = .80,
                                   list = F, times = 1)
# Split into training.
train <- df[in_training, ]

# Split into testing.
test <- df[-in_training, ]

## ---- echo = FALSE, fig.align = "center", fig.asp = .68, fig.width=4.5, warning = FALSE----
M <- cor(df %>% dplyr::select(-private))
corrplot(M, method = "circle")

## ---- echo = FALSE-----------------------------------------------------
fit <- lm(apps ~ ., data = train)

fit.pred <- predict(fit, newdata = test)

mean((test$apps-fit.pred)^2)

## ---- echo = FALSE, fig.align = "center", fig.asp = .68, fig.width=4.5, warning = FALSE----
# We set a grid; for now we use one the book ISLR uses.
grid <- 10^seq (10,-2, length =100)

design_matrix <- model.matrix(apps ~ ., data = train)
design_matrix <- design_matrix[, -1]# remove int.

# Setting alpha = 0 chooses the ridge within the glmnet.
lambda_cv <- cv.glmnet(design_matrix, train$apps,
                       lambda = grid, nfolds = 5, alpha = 0)

plot(lambda_cv)

## --------------------------------------------------------------------------
lambda_cv$lambda.min

## --------------------------------------------------------------------------
lambda_cv$lambda.1se

## --------------------------------------------------------------------------
ridge <- glmnet(design_matrix,
                train$apps, alpha = 0, lambda = lambda_cv$lambda.1se)

test_design_matrix <- model.matrix(apps ~ ., data = test)

test_design_matrix <- test_design_matrix[ , -1]

ridge.pred <- predict.glmnet(ridge, newx = test_design_matrix)

mean((test$apps-ridge.pred)^2)
```

```r
## ---- echo = FALSE, fig.align = "center", fig.asp = .68, fig.width=4.5, warning = FALSE----
# Setting alpha = 1 chooses the lasso within the glmnet.
lambda_cv <- cv.glmnet(design_matrix, train$apps,
                       lambda = grid, nfolds = 5, alpha = 1)

plot(lambda_cv)

## ------------------------------------------------------------------------
lambda_cv$lambda.min

lambda_cv$lambda.1se

## ------------------------------------------------------------------------
lasso <- glmnet(design_matrix,
                train$apps, alpha = 1, lambda = lambda_cv$lambda.1se)

lasso_pred <- predict.glmnet(lasso, newx = test_design_matrix)

mean((test$apps-lasso_pred)^2)

## ------------------------------------------------------------------------
# Load in the ISLR Weekly data.
df <- Weekly

# Fix names for ease.
names(df) <- names(df) %>% tolower()

## ---- echo = FALSE, fig.align = "center", fig.asp = .78, fig.width=6.5, warning = FALSE----
# Pairs plot.
pairs(df %>% dplyr::select(-direction), cex.labels=1, col="dark blue")

## ---- echo = FALSE-------------------------------------------------------

vifval <- vif(df %>% dplyr::select(-direction))

# Make table.
kable(vifval, caption = "Variance Inflation Factor", digits = 3,
      booktabs = TRUE, format = "latex") %>%
  kable_styling(latex_options=c("hold_position"))


## ---- echo = FALSE, fig.align = "center", fig.asp = .68, fig.width=4.5, warning = FALSE----

M <- cor(df %>% dplyr::select(-direction, -year))
corrplot(M, method = "ellipse")

## ----echo = FALSE--------------------------------------------------------
logit_fit <- glm(direction ~ . - today - year, data = df,
                 family = binomial(link = "logit"))

# Let's examine the estimated intercept and slope.
coefs <- round(summary(logit_fit)$coefficients, 2)
```

```r
# Clean up p-vals.
coefs[1 , 4] <- "<0.001"

# Make table.
kable(coefs, caption = "Estimated Coefficients",
      booktabs = TRUE, format = "latex") %>%
  kable_styling(latex_options=c("hold_position"))

## ---- echo = FALSE-------------------------------------------------------
prediction <- predict(logit_fit, data = df, type =  "response")
predicted_direction <- ifelse(prediction >0.5, 1, 0) %>% as.factor()
levels(predicted_direction) <- c("Down", "Up")

confusionMatrix(predicted_direction, df$direction)$table

## ---- echo = FALSE-------------------------------------------------------
# Filter to data only up to 2208 including.
train <- df %>% filter(year < 2009 )  %>% dplyr::select(-today, -year)

# Remaining is test.
test <- df %>% filter(!year < 2009 )  %>% dplyr::select(-today, -year)


logit_fit <- glm(direction ~ lag2, data = train,
                 family = binomial(link = "logit"))

prediction <- predict(logit_fit,
                      newdata = test %>% dplyr:: select(direction, lag2),
                      type =  "response")

predicted_direction <- ifelse(prediction >0.5, 1, 0) %>% as.factor()
levels(predicted_direction) <- c("Down", "Up")
confusionMatrix(predicted_direction, test$direction)$table

## ---- echo = FALSE, fig.width=8, fig.height=3----------------------------
par(mfrow=c(1 ,2))
plot(df$volume, ylab = "volume", main = "No Tranform")
plot(log(df$volume), ylab = "log volume", main = "Log Tranform")

par(mfrow=c(1,2))


## ---- echo = FALSE, include = FALSE--------------------------------------
set.seed(14)
fitControl <- trainControl(## 5-fold CV
  method = "cv",
  number = 5,
  classProbs = TRUE,
  summaryFunction = twoClassSummary)

glm_fit <- train(direction ~ (.-volume+I(log(volume)))^2, data = train,
                    method = "glmStepAIC",
                    trControl = fitControl,
                    na.action = na.omit,
                    metric = "ROC")

## --------------------------------------------------------------------------
glm_fit$finalModel

## ---- echo = FALSE-------------------------------------------------------
final_model <-  glm(direction ~ lag1 +lag2 + lag5:volume,
                data = train,
                family=binomial)



test$pred <- predict(final_model, newdata = test, type = "response")

auc_roc <- roc(test$direction, test$pred)

auc(auc_roc)

coords(auc_roc, "best", ret=c("threshold", "specificity", "1-npv"))


## ---- echo = FALSE-------------------------------------------------------
predicted_direction <- ifelse(test$pred >0.5293889, 1, 0) %>% as.factor()
levels(predicted_direction) <- c("Down", "Up")
confusionMatrix(predicted_direction, test$direction)$table

## ----code = readLines(knitr::purl("C:/Users/jcochanc/Documents/Big-Data/Homework/hw1.Rmd", documentat
## NA
```