Tom Meagher
DS 502 Homework 3
November 15, 2016

1. Section 6.8, page 259, question 2

a.

iii, because lasso is less flexible and has better predictive power due to less variance, more bias. Furthermore, lasso has better predictive power in situations where the bias is higher than variance.

b.

iii, because ridge regression is less flexible than least squares. Similar to the lasso, ridge regression also performs better in the presence of higher bias, lower variance. As lambda increases the flexibility and variance decrease, while the bias increases.

c.

ii, because non-linear methods are more flexible than least squares. In the variance-bias tradeoff, non-linear models perform better in high variance, low bias (relative to each other) situations.

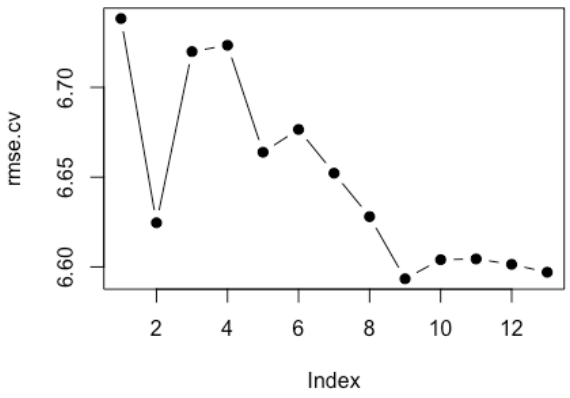
2. Section 6.8, page 264, question 11

a.

```
set.seed(1)
library(MASS)
library(leaps)
library(glmnet)
```

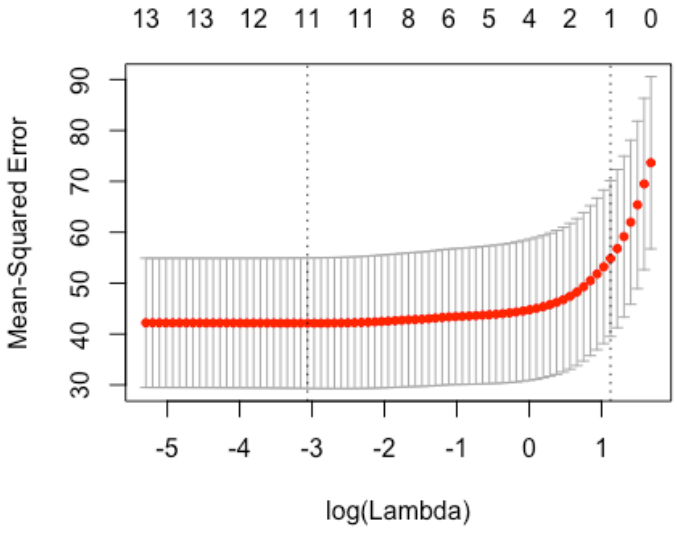
Best Subset Selection

```
predict.regsubsets = function(object, newdata, id, ...) {
  form = as.formula(object$call[[2]])
  mat = model.matrix(form, newdata)
  coefi = coef(object, id = id)
  mat[, names(coefi)] %*% coefi
}

k = 10
p = ncol(Boston) - 1
folds = sample(rep(1:k, length = nrow(Boston)))
cv.errors = matrix(NA, k, p)
for (i in 1:k) {
  best.fit = regsubsets(crim ~ ., data = Boston[folds != i, ], nvmax = p)
  for (j in 1:p) {
    pred = predict(best.fit, Boston[folds == i, ], id = j)
    cv.errors[i, j] = mean((Boston$crim[folds == i] - pred)^2)
  }
}
rmse.cv = sqrt(apply(cv.errors, 2, mean))
plot(rmse.cv, pch = 19, type = "b")

which.min(rmse.cv)
> 9

rmse.cv[which.min(rmse.cv)]
> 6.593396
```



Lasso

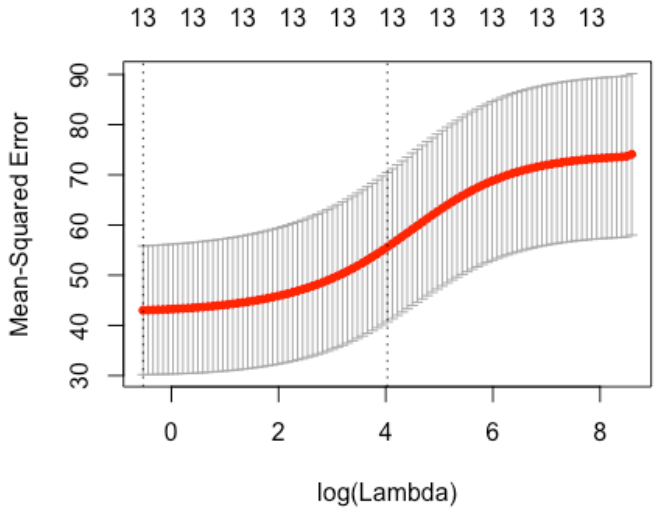
```
x = model.matrix(crim ~ . - 1, data = Boston)
y = Boston$crim
cv_lasso = cv.glmnet(x, y, type.measure = "mse")
plot(cv_lasso)
```




```
rmse = sqrt(cv_lasso$cvm[cv_lasso$lambda == cv_lasso$lambda.1se])
rmse
> 7.405176
```

Ridge Regression

```
x = model.matrix(crim ~ . - 1, data = Boston)
y = Boston$crim
cv_ridge = cv.glmnet(x, y, type.measure = "mse", alpha = 0)
plot(cv_ridge)

rmse =
sqrt(cv_ridge$cvm[cv_ridge$lambda
== cv_ridge$lambda.1se])
rmse
> 7.456946
```



PCR

```
pcr_fit = pcr(crim ~ ., data = Boston, scale = TRUE, validation = "CV")
summary(pcr_fit)
```

```
Data:   X dimension: 506 13
        Y dimension: 506 1
Fit method: svdpc
Number of components considered: 13

VALIDATION: RMSEP
Cross-validated using 10 random segments.
       (Intercept)  1 comps  2 comps  3 comps  4 comps  5 comps  6 comps  7 comps  8 comps  9 comps
CV           8.61    7.170    7.163    6.733    6.728    6.740    6.765    6.760    6.634    6.652
adjCV        8.61    7.169    7.162    6.730    6.723    6.737    6.760    6.754    6.628    6.644
       10 comps  11 comps  12 comps  13 comps
CV        6.642     6.652     6.607     6.546
adjCV     6.635     6.643     6.598     6.536

TRAINING: % variance explained
      1 comps  2 comps  3 comps  4 comps  5 comps  6 comps  7 comps  8 comps  9 comps  10 comps
X       47.70    60.36    69.67    76.45    82.99    88.00    91.14    93.45    95.40     97.04
crim    30.69    30.87    39.27    39.61    39.61    39.86    40.14    42.47    42.55     42.78
      11 comps  12 comps  13 comps
X        98.46     99.52     100.0
crim     43.04     44.13      45.4
```

Out of all the models, PCR has the best root mean squared error (rmse), achieving it on the 13-component fit.

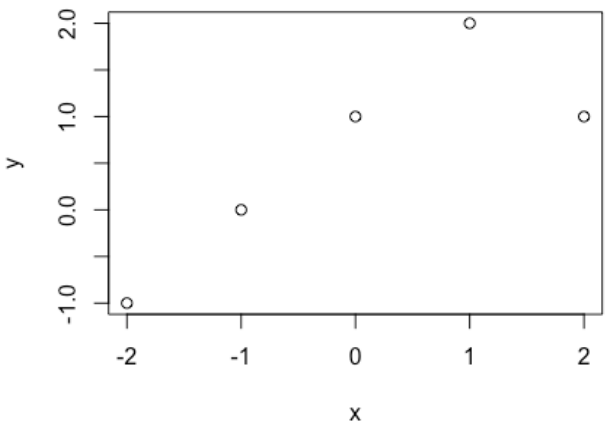
b.
Based on the previous exploration in part a, the model that performs the best after cross validation is the 13-component fit (6.546). Next, the best subset model (#9) does similarly with a rmse of 6.593396. The lasso and ridge regression models still do well, but are definitely worse than the previous two. I would propose using either the 13-component PCR or #9 best subset model. From the perspective of parsimony, the #9 best subset model would be preferred.

c.
No, the chosen model in the case of parsimony (the number nine best subset model), does not contain all the features. The best performing model (13-component PCR) does contain all the features.

3. Section 7.9, Page 298, question 3

```
x = seq(-2,2)
y = 1 + x + -2 * (x-1)^2 * (x>1)
lm(y~x)
plot(x, y)
```

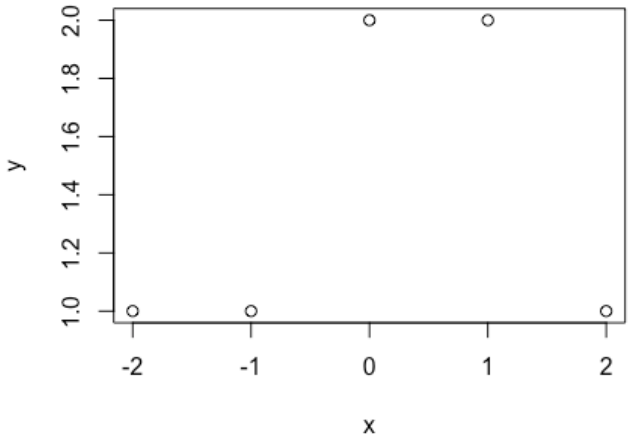
```
Call:
lm(formula = y ~ x)

Coefficients:
(Intercept)            x
        0.6          0.6
```



4. Section 7.9, Page 298, question 4

```
x = seq(-2,2)
y = c(
  1 + 0 + 0, # x = -2
  1 + 0 + 0, # x = -1
  1 + 1 + 0, # x = 0
  1 + (1-0) + 0, # x = 1
  1 + (1-1) + 0 # x =2
)
lm(y~x)
plot(x,y)
```

```
Call:
lm(formula = y ~ x)

Coefficients:
(Intercept)            x
        1.4          0.1
```



5. Section 7.9, Page 299, question 6

a.

gˆ2 will have the smaller *training* RSS because it is an order higher (4) than gˆ1 (3).

b.

gˆ1 will have the smaller *test* RSS because gˆ2 could overfit the data since it has one more degree of freedom than gˆ1.

c.

If lambda is equal to zero then gˆ1 and gˆ2 will be identical because the second part of the equation will be zero.

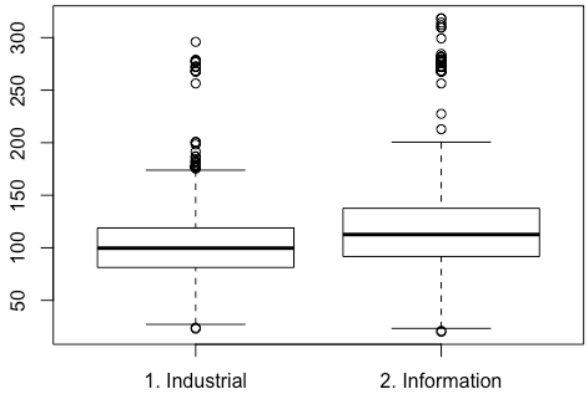
6. Section 7.9, Page 299, question 7

```
library(ISLR)
set.seed(1)
attach(Wage)

summary(maritl)
summary(jobclass)

plot(maritl, wage)
```

On average, it looks like married couples earn more than the other four groups. Married couples is also the only group that has many outliers in each direction.

```
plot(jobclass, wage)
```

On average, it appears information workers earn more than their industrial counterparts. Both groups have many upper adjacent value outliers.




```
lm_maritl = lm(wage~maritl, data=Wage)
lm_jobclass = lm(wage~jobclass, data=Wage)
lm_maritl_jobclass = lm(wage~maritl+jobclass, data=Wage)
library(gam)
lm_gam = gam(wage~maritl+jobclass+s(age,4)+s(year,5), data=Wage)

anova(lm_maritl, lm_jobclass, lm_maritl_jobclass, lm_gam)
```

```
Analysis of Variance Table

Model 1: wage ~ maritl
Model 2: wage ~ jobclass
Model 3: wage ~ maritl + jobclass
Model 4: wage ~ maritl + jobclass + s(age, 4) + s(year, 5)
  Res.Df     RSS      Df Sum of Sq      F    Pr(>F)
1   2995 4858941
2   2998 4998547 -3.0000   -139606 31.229 < 2.2e-16 ***
3   2994 4654752  4.0000    343795 57.679 < 2.2e-16 ***
4   2985 4448035  9.0002    206717 15.413 < 2.2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

I looked at three linear models containing marital, jobless, and marital and jobclass. Also, I looked at a generalized additive model with marital and jobclass, but added in smoothing spline for age and year because splines cannot be fit on categorical predictors (maritl and jobclass) without doing things we haven't learned in this class.

7. Section 8.4, Page 332, question 1

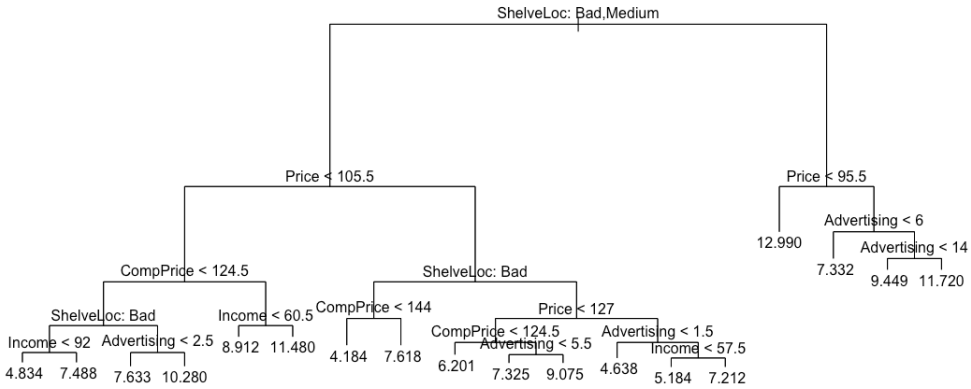
8. Section 8.4, Page 333-334, question 8

a.

```
library(ISLR)
attach(Carseats)
set.seed(1)

dim_carseats = dim(Carseats)[1]
train = sample(dim_carseats, dim_carseats / 2)
Carseats_train = Carseats[train, ]
Carseats_test = Carseats[-train, ]
```

b.

```
library(tree)
tree_carseats = tree(Sales~., data=Carseats_train)

plot(tree_carseats)
text(tree_carseats, pretty=0)
```



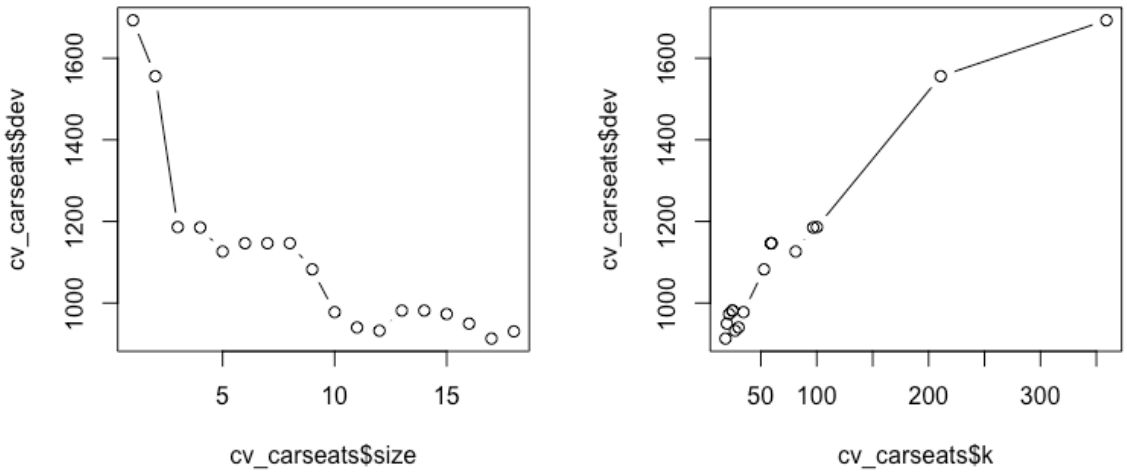

```
pred_carseats = predict(tree_carseats, Carseats_test)
mean((Carseats_test$Sales - pred_carseats)^2)
> 4.970093
```

The test MSE is 4.97. Out of the 11 variables in the mode, only five are used: ShelveLoc, Price, CompPrice, Income, and Advertising.

c.

```
cv_carseats = cv.tree(tree_carseats, FUN=prune.tree)

par(mfrow=c(1, 2))
plot(cv_carseats$size, cv_carseats$dev, type="b")
plot(cv_carseats$k, cv_carseats$dev, type="b")
```

In the plot below (next page), it looks like the 9th model does the best job when optimizing for parsimony and fit.




```
pruned_carseats = prune.tree(tree_carseats, best=9)
pred_pruned = predict(pruned_carseats, Carseats_test)
mean((Carseats_test$Sales - pred_pruned)^2)
> 5.018936
```

Pruning actually increases the test MSE to 5.018.

d.

```
library(randomForest)
bag_carseats = randomForest(Sales~., data=Carseats_train, mtry=10, ntree=500,
importance=T)
bag_pred = predict(bag_carseats, Carseats_test)
mean((Carseats_test$Sales - bag_pred)^2)
> 2.65099
```

The test MSE was reduced to 2.65.

```
importance(bag_carseats)
```

	%IncMSE	IncNodePurity
CompPrice	22.4787408	155.100910
Income	5.1771015	78.615594
Advertising	18.7254956	152.668343
Population	0.4592137	59.773966
Price	55.3953219	491.914582
ShelveLoc	52.7869879	441.393573
Age	19.4453267	158.376309
Education	1.4589418	46.482424
Urban	0.9982170	8.660750
US	3.4396458	9.387027

Price, ShelveLoc, and CompPrice are the most important variables.

e.

```
rf_carseats = randomForest(Sales~., data=Carseats_train, mtry=5, ntree=500,
importance=T)
rf_pred = predict(rf_carseats, Carseats_test)
mean((Carseats_test$Sales - rf_pred)^2)
```

```
> 2.75834
```

The test MSE was increased to 2.75.

```
importance(rf_carseats)
```

	%IncMSE	IncNodePurity
CompPrice	18.2829427	151.18058
Income	4.8789701	87.46495
Advertising	13.6059157	156.96432
Population	0.9805890	82.93559
Price	42.7427462	477.66748
ShelveLoc	44.7428399	385.98136
Age	16.1881773	165.10743
Education	1.4530194	54.16966
Urban	0.2006829	10.04724
US	5.5083414	15.03084

ShelveLoc, Price, and CompPrice are the most important variables.

Changing *m* varies the MSE. When *m* is 1, the MSE is 4.577363. In contrast, when *m* is 10, the MSE is 2.707132.