Name: Jingyi Yuan
UNI: jy2736
Class: STAT W4240

Homework 05

Question 1

(a).

For the best subset selection, we choose the smallest RSS for $\binom{p}{k}$ models and there are 2^p models in total. For forward and backward stepwise selection, we fit (p-k) models at each step and fit $1 + \frac{p(p+1)}{2}$ models in total. So the best subset selection has the smallest RSS since it chooses the smallest RSS among all models.

(b).

It depends on the test dataset to decide which of the three models with k predictors has the smallest test RSS. The best subset selection may have the smallest test RSS since it makes the choice among all k predictors models but the other two may still have a smaller RSS depending on the test data.

(c).

i. TRUE.

Forward stepwise selection begins with a model containing no predictors, and then adds predictors to the model, one-at-a-time, until all of the predictors are in the mode. So the predictors in the (k+1)-variable model are formed by adding a predictor to the k-variable model.

ii. TRUE.

Backward stepwise selection begins with the full least squares model containing all p predictors, and then iteratively removes the least useful predictor, one-at-a-time. So the predictors in the k-variable model are formed by removing a predictor to the (k+1)-variable model.

iii. FALSE.

We remove the least useful predictor when doing backward stepwise selection. So from $\mathcal{M}_{k+1}$ to $\mathcal{M}_k$ we may move any variable in the (k+1) variables. The moved one may be different from the adding predictor in forward stepwise selection.

iv. FALSE

Same as in iii, from $\mathcal{M}_{k+1}$ to $\mathcal{M}_k$ we may move any variable in the (k+1) variables. The moved one may be different from the adding predictor in forward stepwise selection.

v. FALSE

We may choose k variables for $\mathcal{M}_k$ and (k+1) variables for $\mathcal{M}_{k+1}$ and they may

not have much same variables since we may not choose the same k variables for $\mathcal{M}_{k+1}$, thus the k-variable model may not be the subset of (k+1) one.

Question 2

(a). iv

As we increase s from 0, β_j will make the model fit the training data better (the model will be more flexible) and RSS for the training will decrease steadily.

(b). ii

For the test data, as β_j makes the model fit the training data better, the RSS will firstly decrease since the model is more accurate. Then it will increase because of overfitting.

(c). iii

As s increase from 0, the model becomes more flexible. The variance starts from 0 (when s is 0) and increase steadily since the model is more flexible.

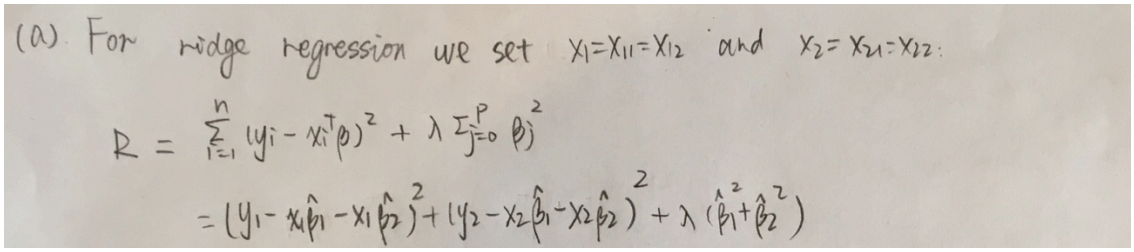
(d). iv

As β_j makes the model fit the training data better, the model becomes more and more flexible and the squared bias will steadily decrease since the model fits the training set well.

(e). v

According to the definition of irreducible error, it is dependent with the model thus the increase of s will not affect on irreducible error.

Question 3



(b). $\frac{\partial R}{\partial \hat{\beta}_1} = 2(y_1 - x_1\hat{\beta}_1 - x_1\hat{\beta}_2)\cdot(-x_1) + 2(y_2 - x_2\hat{\beta}_1 - x_2\hat{\beta}_2)\cdot(-x_2) + \lambda\cdot 2\hat{\beta}_1 = 0$

$\therefore -2x_1y_1 + 2x_1^2\hat{\beta}_1 + 2x_1^2\hat{\beta}_2 + (-2x_2y_2) + 2x_2^2\hat{\beta}_1 + 2x_2^2\hat{\beta}_2 + 2\lambda\hat{\beta}_1 = 0$

$(x_1^2 + x_2^2 + 2\lambda)\hat{\beta}_1 + (x_1^2 + x_2^2)\hat{\beta}_2 = x_1y_1 + x_2y_2$

$\frac{\partial R}{\partial \hat{\beta}_2} = 2(y_1 - x_1\hat{\beta}_1 - x_1\hat{\beta}_2)\cdot(-x_1) + 2(y_2 - x_2\hat{\beta}_1 - x_2\hat{\beta}_2)\cdot(-x_2) + \lambda\cdot 2\hat{\beta}_2 = 0$

$\therefore (x_1^2 + x_2^2)\hat{\beta}_1 + (x_1^2 + x_2^2 + 2\lambda)\hat{\beta}_2 = x_1y_1 + x_2y_2$

$\therefore 2\lambda\hat{\beta}_1 = 2\lambda\hat{\beta}_2 \Rightarrow$ we have $\hat{\beta}_1 = \hat{\beta}_2$

(c). We still set $x_1 = x_{11} = x_{12}$ and $x_2 = x_{21} = x_{22}$

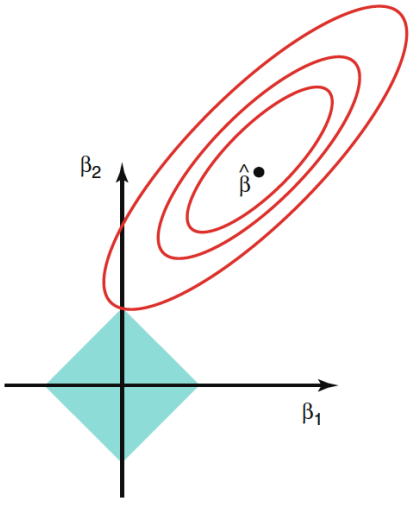
$R = \frac{n}{2}\sum_{i=1}(y_i - x_i^T\beta)^2 + \lambda\sum_{j=1}^{p}|\beta_j|$

$= (y_1 - x_1\hat{\beta}_1 - x_1\hat{\beta}_2)^2 + (y_2 - x_2\hat{\beta}_1 - x_2\hat{\beta}_2)^2 + \lambda(|\hat{\beta}_1| + |\hat{\beta}_2|)$

(d).

Geometrically, the Lasso regression is equivalent to $\sum_{i=1}^{n}(y_i - x_i^T\beta)^2 + \gamma$ subject to $\lambda\sum_{j=1}^{p}|\beta_j| \le \gamma$. The estimator β^{Lasso} is given by the first point at which an ellipse contacts the constraint region.

The ellipses that are centered around β represent regions of constant RSS. In other words, all of the points on a given ellipse share a common value of the RSS. As the ellipses expand away from the least squares coefficient estimates, the RSS increases.



Since $x_1 + x_2 = 0\ and\ y_1 + y_2 = 0$, we need to minimize $y_1 - (\beta_1 + \beta_2)x_1 \geq 0$. We have $\beta_1 + \beta_2 = \frac{y_1}{x_1}$. This is a line and it has a lot of solutions, thus there are many possible solutions to the optimization problem in (c).

Question 4

Majority vote:

When P(Class is Red|X) is 0.1, 0.15, 0.2, 0.2, X is classified as freen. When P(Class is Red|X) is 0.55, 0.6, 0.6, 0.65, 0.7 and 0.75, X is classified as red. In this case, X is red since the majority (6 out of 10) is red.

Average probability:

$$(0.1 + 0.15 + 0.2 + 0.2 + 0.55 + 0.6 + 0.6 + 0.65 + 0.7 + 0.75)/10 = 0.45$$

Since the average of P(Class is Red|X) is 0.45, which is smaller than 0.5, using this approach, X is classified as green.

Question 5

(a).

Use rpart classification with Gini impurity coefficient splits:

```
tree.training = rpart(y~., data = training)
```

Then we get a tree:

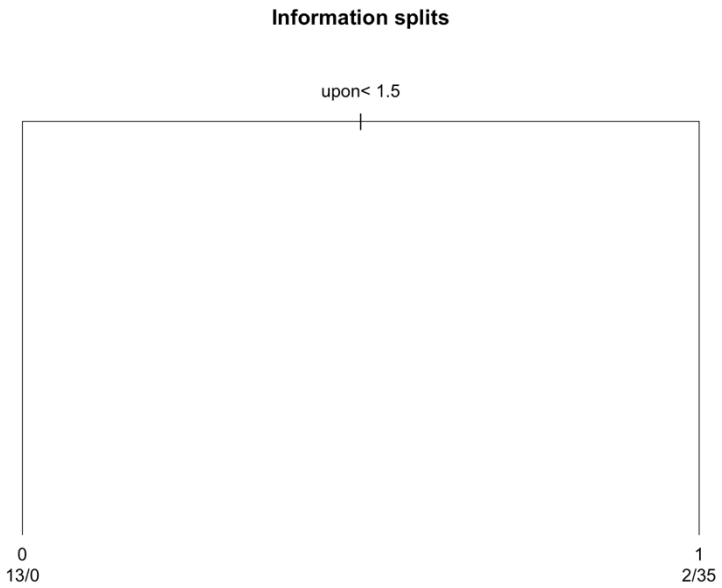
```
> right_a
[1] 0.962963
> false_p_a
[1] 0.09090909
> false_n_a
[1] 0
```

(b).

Use rpart classification with information splits:

```
tree.training = rpart(y~., data = training, parms = list(split = 'information'))
```

Then we get a tree:




```
> right_b
[1] 0.962963
> false_p_b
[1] 0.09090909
> false_n_b
[1] 0
```

They are the same. Both of the two split use the word “upon” as the key word so the result of Gini splits and information splits are the same.

Question 6

(a).

For feature selection, unregularized logistic regression will make the optimization problem computationally more expensive to solve. So we do not use it and we usually scale the dataset at first.

(b).

```
> right_b
[1] 0.5925926
> false_p_b
[1] 1
> false_n_b
[1] 0
```

Ten most important words:

	1
(Intercept)	8.472979e-01
upon	3.204000e-37
whilst	2.986415e-37
februari	2.637540e-37
although	2.497130e-37
form	2.388696e-37
sever	2.349627e-37
among	2.325560e-37
within	2.316157e-37
mix	2.246210e-37
fulli	2.227254e-37

(c).

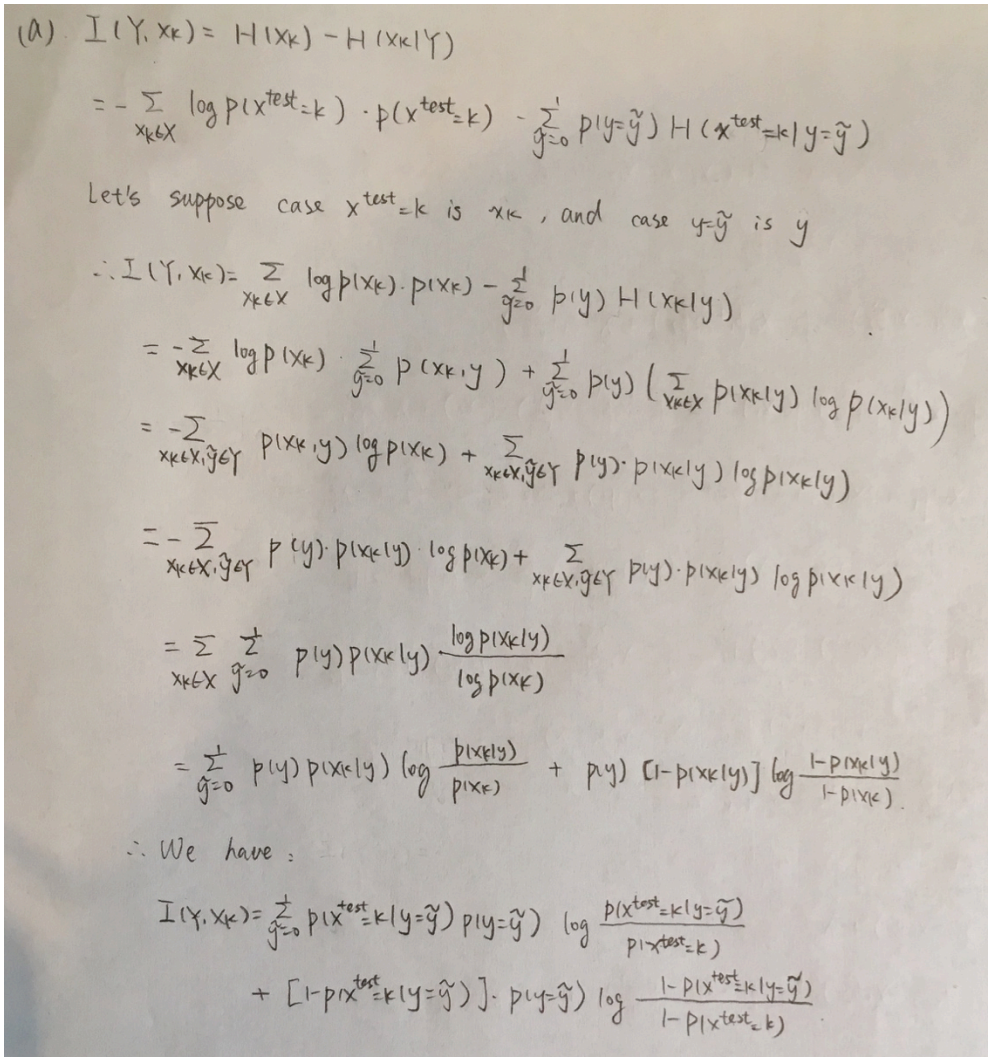
```
> right_c
[1] 0.8888889
> false_p_c
[1] 0.2727273
> false_n_c
[1] 0
```

Ten most important words:

	1
(Intercept)	1.23385689
upon	0.73757930
whilst	0.61945747
februari	0.38930336
form	0.26393431
within	0.19244100
sever	0.18724603
although	0.16738621
lesser	0.07967237
mix	0.02066491
state	0.00000000

The words selected are almost same but their coefficients and weights are different. In ridge regression, the coefficients are much smaller, but in lasso, the coefficients is much bigger than those in ridge.

Question 7



(a) $I(Y, x_k) = H(x_k) - H(x_k|Y)$

$= -\sum_{x_k \in X} \log p(x^{test}=k) \cdot p(x^{test}=k) - \sum_{\tilde{y}=0}^{1} p(y=\tilde{y}) H(x^{test}=k|y=\tilde{y})$

Let's suppose case $x^{test}=k$ is x_k, and case $y=\tilde{y}$ is y

$\therefore I(Y, x_k) = \sum_{x_k \in X} \log p(x_k) \cdot p(x_k) - \sum_{\tilde{y}=0}^{1} p(y) H(x_k|y)$

$= -\sum_{x_k \in X} \log p(x_k) \sum_{\tilde{y}=0}^{1} p(x_k, y) + \sum_{\tilde{y}=0}^{1} p(y) \left(\sum_{x_k \in X} p(x_k|y) \log p(x_k|y) \right)$

$= -\sum_{x_k \in X, \tilde{y} \in Y} p(x_k, y) \log p(x_k) + \sum_{x_k \in X, \tilde{y} \in Y} p(y) \cdot p(x_k|y) \log p(x_k|y)$

$= -\sum_{x_k \in X, \tilde{y} \in Y} p(y) \cdot p(x_k|y) \cdot \log p(x_k) + \sum_{x_k \in X, \tilde{y} \in Y} p(y) \cdot p(x_k|y) \log p(x_k|y)$

$= \sum_{x_k \in X} \sum_{\tilde{y}=0}^{1} p(y) p(x_k|y) \frac{\log p(x_k|y)}{\log p(x_k)}$

$= \sum_{\tilde{y}=0}^{1} p(y) p(x_k|y) \left(\log \frac{p(x_k|y)}{p(x_k)} + p(y) [1 - p(x_k|y)] \log \frac{1 - p(x_k|y)}{1 - p(x_k)} \right)$

$\therefore$ We have:

$I(Y, x_k) = \sum_{\tilde{y}=0}^{1} p(x^{test}=k|y=\tilde{y}) p(y=\tilde{y}) \log \frac{p(x^{test}=k|y=\tilde{y})}{p(x^{test}=k)}$

$+ [1 - p(x^{test}=k|y=\tilde{y})] \cdot p(y=\tilde{y}) \log \frac{1 - p(x^{test}=k|y=\tilde{y})}{1 - p(x^{test}=k)}$

(b).

We calculate each $P(x^{test} = k|y = \tilde{y})$ for dtm.hamilton.train

	V1	V2	V3	V4	V5	V6	V7	V8	V9
1	0.012738854	0.028901734	0.057636888	0.038369305	0.02071006	0.020000000	0.021739130	0.040723982	0.0187793
2	0.015923567	0.017341040	0.011527378	0.021582734	0.01183432	0.030000000	0.027173913	0.031674208	0.0375586
3	0.028662420	0.021194605	0.034582133	0.014388489	0.01183432	0.016666667	0.027173913	0.054298643	0.0093896
4	0.022292994	0.025048170	0.025936599	0.028776978	0.02958580	0.016666667	0.048913043	0.040723982	0.0234741
5	0.017515924	0.065510597	0.051873199	0.009592326	0.01183432	0.013333333	0.043478261	0.022624434	0.0093896
6	0.028662420	0.025048170	0.063400576	0.016786571	0.02071006	0.010000000	0.065217391	0.040723982	0.0187793
7	0.017515924	0.021194605	0.025936599	0.031175060	0.02071006	0.016666667	0.048913043	0.031674208	0.0187793
8	0.019108280	0.023121387	0.048991354	0.021582734	0.02662722	0.013333333	0.021739130	0.058823529	0.0234741
9	0.020700637	0.011560694	0.048991354	0.043165468	0.02958580	0.016666667	0.032608696	0.040723982	0.0093896

and for dtm.madison.train

	V1	V2	V3	V4	V5	V6	V7	V8	V9
1	0.012738854	0.019267823	0.040345821	0.011990408	0.05325444	0.016666667	0.032608696	0.004524887	0.0281690
2	0.033439490	0.023121387	0.060518732	0.057553957	0.02662722	0.053333333	0.032608696	0.004524887	0.0610328
3	0.057324841	0.028901734	0.097982709	0.033573141	0.01775148	0.063333333	0.135869565	0.144796380	0.0610328
4	0.058917197	0.013487476	0.083573487	0.052757794	0.02071006	0.046666667	0.070652174	0.031674208	0.0469483
5	0.082802548	0.011560694	0.020172911	0.067146283	0.05029586	0.016666667	0.016304348	0.040723982	0.0328638
6	0.113057325	0.038535645	0.080691643	0.023980815	0.07396450	0.080000000	0.016304348	0.009049774	0.0610328
7	0.081210191	0.019267823	0.034582133	0.105515588	0.01775148	0.103333333	0.032608696	0.013574661	0.0234741
8	0.065286624	0.061657033	0.100864553	0.055155875	0.02662722	0.016666667	0.097826087	0.004524887	0.0187793
9	0.078025478	0.078998073	0.144092219	0.016786571	0.03846154	0.006666667	0.152173913	0.036199095	0.0422535

Also, we calculate $P(y = 1)$ and $P(y = 0)$

pyisone_train	0.7
pyiszore_train	0.3

Then we calculate I to help deciding top n features:

Question 8

In algorithm 8.2, first we set $f(x) = 0$ and $r_i = y_i$ for all i in the training set. Then for b = 1, we set $f(x) = \lambda f^1(x)$ and $r_1 = r_1 - \lambda f^1(x) = y_1 - \lambda f^1(x)$. We can assume $f_1(x_1) = \lambda f^1(x)$.

Then for b = 2, we set $f(x) = \lambda f^1(x) + \lambda f^2(x)$ and $r_2 = r_2 - \lambda f^2(x) = y_2 - \lambda f^1(x) - \lambda f^2(x)$. We can still assume $f_2(x_2) = \lambda f^2(x)$. Thus $f(x) = f_1(x_1) + f_2(x_2)$.

For b = 1, 2, …, p, we repeat the step and we have $f(x) = \sum_{j=0}^{p} f_j(x_j)$.

Question 9

(a).

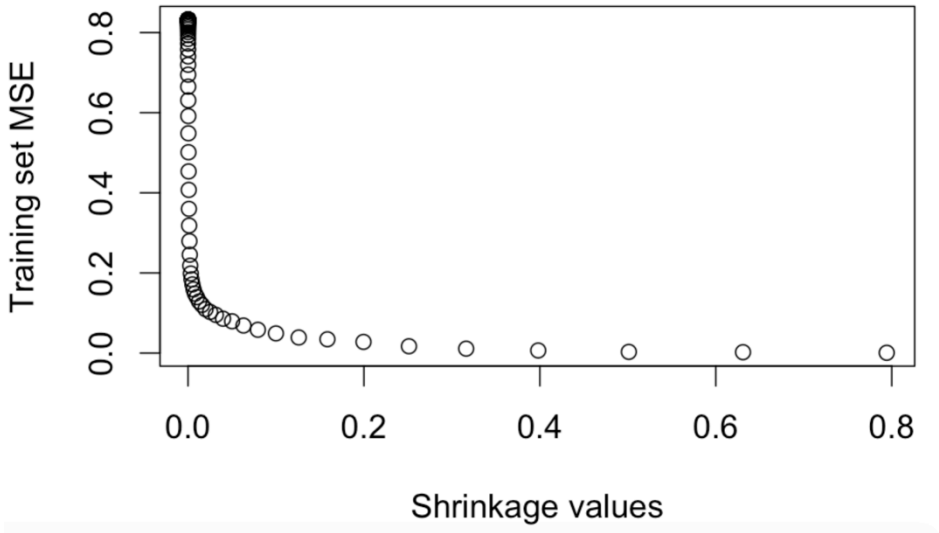
```
#(a)
Hitters = na.omit(Hitters)
Hitters$Salary = log(Hitters$Salary)
```

```
> Hitters$Salary
  [1] 6.163315 6.173786 6.214608 4.516339 6.620073 4.248495 4.605170 4.317488 7.003065 6.248319 6.239301
 [12] 6.309918 6.551080 5.480639 6.652863 5.164786 4.905275 4.605170 4.744932 6.396930 6.655012 6.639876
 [23] 6.562914 6.620073 6.437752 6.802395 4.700480 6.417549 5.703782 6.745236 4.499810 4.212128 5.192957
 [34] 5.720312 5.370638 5.511411 6.703188 6.774224 4.248495 7.090077 6.514713 6.028279 5.828946 6.032287
 [45] 7.207860 4.499810 5.616771 5.438079 5.416100 6.856462 4.317488 4.653960 5.768321 6.745236 6.282267
 [56] 6.838762 6.745236 5.347108 5.783825 5.616771 6.109248 7.588324 7.549609 6.396930 6.948578 4.700480
 [67] 5.560682 6.163315 6.067268 7.106606 4.248495 4.976734 6.388561 7.529116 5.703782 6.194405 7.807917
 [78] 5.926926 6.620073 7.069023 4.248495 7.313220 5.953243 7.562978 5.370638 6.802395 5.043425 6.551080
 [89] 6.282267 5.893024 6.597600 5.298317 5.991465 5.991465 6.603266 6.214608 6.396930 6.496021 6.856462
[100] 6.620073 5.695414 5.783825 4.471639 5.164786 4.499810 7.120848 6.063785 4.605170 5.105945 5.521461
[111] 7.170120 6.650710 6.916054 5.616771 6.652863 6.745236 5.899897 4.553877 4.700480 4.605170 5.625821
[122] 4.382027 6.396930 5.298317 6.487684 4.317488 7.788419 5.521461 5.043425 6.461468 5.703782 4.700480
[133] 6.715383 5.273000 6.109248 6.445720 4.460144 7.170120 6.907755 7.495542 7.177782 6.603266 6.437752
[144] 4.828314 6.950176 6.586172 5.703782 5.899897 4.317488 7.076090 5.310740 5.416100 6.263398 5.579730
[155] 6.668863 6.684612 6.375876 4.976734 6.040255 4.317488 6.354370 6.659294 4.499810 5.010635 6.551080
[166] 6.309918 6.476972 4.219508 4.605170 6.507278 5.164786 4.919981 7.662624 6.774224 4.787492 4.941642
[177] 5.347108 6.684612 5.480639 5.857933 5.164786 5.298317 7.570443 6.551080 6.620073 6.109248 5.147494
[188] 7.138867 6.620073 5.247024 6.363028 4.867534 6.109248 5.703782 5.521461 6.956545 5.370638 5.991465
[199] 6.327937 7.420579 6.189290 6.052089 6.214608 5.521461 5.991465 6.109248 6.620073 4.248495 6.774224
[210] 5.247024 5.252273 6.606650 5.521461 4.941642 4.579852 6.606650 4.941642 5.833837 6.907755 4.605170
[221] 4.499810 5.298317 4.905275 5.043425 6.163315 7.279319 5.010635 4.653960 5.857933 4.499810 6.272877
[232] 5.833837 6.845880 5.857933 5.788941 5.521461 6.606650 6.052089 6.829794 5.220356 6.824374 5.658321
[243] 5.501258 5.459586 7.047517 5.075174 6.052089 6.802395 6.214608 5.625821 6.620073 5.075174 7.170120
[254] 6.263398 6.309918 7.377759 4.787492 5.105945 6.551080 6.774224 5.953243 6.866933 6.907755
```

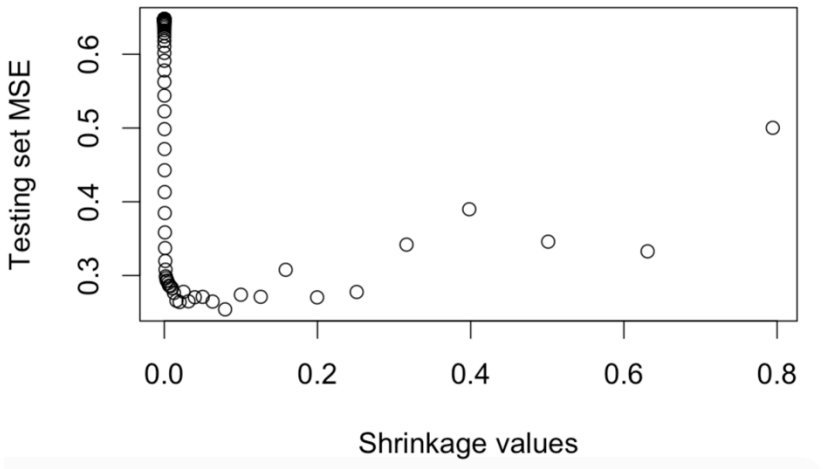
(b).

```
#(b)
row_H = nrow(Hitters)
train_set = Hitters[1:200,]
test_set = Hitters[201:nrow(Hitters),]
```

(c).



(d).



(e).

```
> MSE_liner
[1] 0.4917959
> MSE_lda
[1] 0.751275
> min(MSE_test)
[1] 0.2540265
```

The test MSE of boosting is smaller that the test MSE of linear regression and LDA.

(f).

```
                var    rel.inf
CAtBat       CAtBat 20.2245983
CWalks       CWalks 11.1835993
PutOuts     PutOuts  7.6928475
Years         Years  6.4307812
CRBI           CRBI  6.3064257
Walks         Walks  5.9182699
Hits           Hits  5.8240425
CHits         CHits  5.4837385
CRuns         CRuns  4.9304814
Assists     Assists  4.7160209
RBI             RBI  4.7133030
CHmRun       CHmRun  4.3214612
HmRun         HmRun  3.2470809
AtBat         AtBat  2.7418579
Runs           Runs  2.2641746
Errors       Errors  2.2321636
Division   Division  0.7965017
NewLeague NewLeague  0.7293346
League       League  0.2433170
```

CAtBat appears to be the most important variables in the boosted model. Other variables are shown in the picture above.

(g).

We use randomForest to apply bagging to the training set:

```
bagging = randomForest(Salary ~ ., data = train_set, ntree = 1000)
pred_y_bag = predict(bagging, newdata = test_set)
MSE_bagging = mean((pred_y_bag - test_set$Salary)^2)
```

And we get:

```
> MSE_bagging
[1] 0.2186588
```