STATS 216v

Stanford University, Summer 2017

Problem Set 3 Solutions

Total: 81 points

---

1. *This problem is worth 10 points. 4 for a (1 for correct answer, 3 for correct justification). 3 each for b and c (for correct answer/justification. Only minimal justification is required). If students did not understand what $\hat{g}_1$, $\hat{g}_2$ and $\hat{g}_3$ were and that corrupted their answers they can still earn credit. If they explain what they think $\hat{g}_1$, $\hat{g}_2$ and $\hat{g}_3$ are, AND give answers that are consistent, award up to 6 points. If they says that one of the models will perform better for test data remove 2 points.*

   In general, when we set $\lambda = \infty$, it means that the term we are penalize will be driven to 0. For $\hat{g}_1$, this means that the 2^{nd} derivative of $\hat{g}$ must be 0 everywhere. This also implied that all higher derivatives must be 0 as well. We are left with a linear expression. Similar logic applies to $\hat{g}_2$ and $\hat{g}_3$, meaning that it will be a polynomial of degree 2 and 3 respectively.

   (a) $\hat{g}_3$ will have the smalleest training RSS because it can fit every model $\hat{g}_1$ and $\hat{g}_2$ can fit and more.

   (b) It is impossible to know which will have smaller test RSS. If the true model is cubic then $\hat{g}_3$ will have an advantage. If it is quadratic then $\hat{g}_2$ will (lower variance). If it is linear then $\hat{g}_1$ will. In other cases there will be a more complicated bias variance trade-off.

   (c) In this case all three models would be the same. This is because the penalty term is set to 0, so all we have left is the same optimization problem for all three models. Thus, training and test RSS will both be equal.

2. *This problem is worth 14 points: one point for each correct answer (a, b, and c i-v) and one point for each explanation that correctly explains either the given answer or the correct answer.*

   (a) It is impossible to know which model will have the smallest test RSS. Training is always done independently of the test set, so we could have anything happen in the test data.

   (b) The best subset model will have the smallest training RSS. This is how best subset is defined.

   (c) i. FALSE. The variables selected by forward and backward stepwise need not be related.

      ii. FALSE. Same reason as above.

      iii. FALSE. It is possible to have no overlap between these models. Sometimes a group of variables will be very predictive of a response even when no single variable is.

iv. TRUE. In moving from $k+1$ to k variables, backward stepwise will pick one more variable to exclude.

v. TRUE. In moving from k to $k+1$ variables, forward stepwise will pick one more variable to include.

3. *This problem is worth 22 points: 4 points for a, 4 points for b, 3 points for c, 4 points for d, 5 points for e, 2 points for f.*

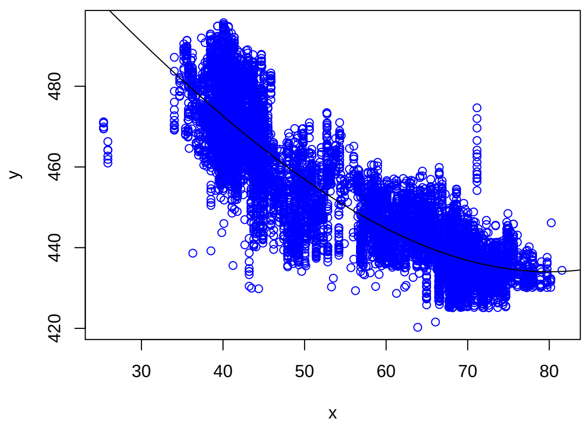
   Note: complete code is attached at the end of this solution.

   (a) *Grading notes for part (a): 2 points for the plot, 2 points for the summary.*

   The summary output and the plot of data points are as following:

```
Call:
lm(formula = y ~ poly(x, 3))

Residuals:
    Min       1Q   Median       3Q      Max
-40.529   -4.983   -0.160    5.251   38.418

Coefficients:
              Estimate Std. Error  t value Pr(>|t|)
(Intercept)  4.544e+02  8.277e-02 5489.342   <2e-16 ***
poly(x, 3)1 -1.452e+03  8.096e+00 -179.333   <2e-16 ***
poly(x, 3)2  2.262e+02  8.096e+00   27.938   <2e-16 ***
poly(x, 3)3  1.995e+01  8.096e+00    2.465   0.0137 *
---
Signif. codes:  0    ***    0.001    **    0.01    *    0.05    .    0.1         1

Residual standard error: 8.096 on 9564 degrees of freedom
Multiple R-squared:  0.775,     Adjusted R-squared:  0.775
F-statistic: 1.098e+04 on 3 and 9564 DF,  p-value: < 2.2e-16
```

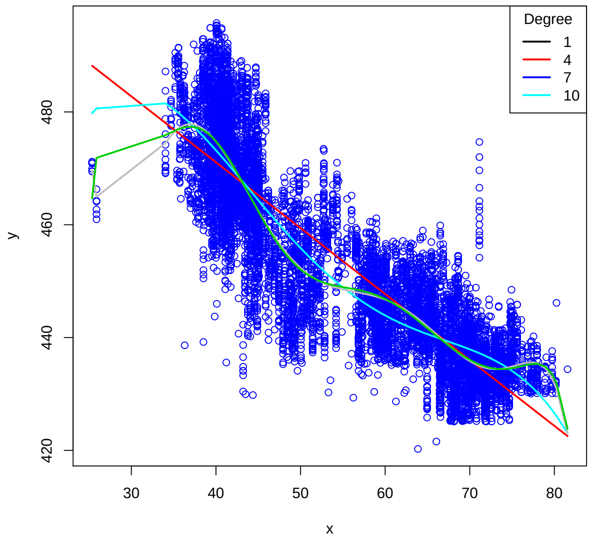
FIGURE 1. Plot for 3a



(b) *Grading notes for part (b): 2 points for the plot, 2 points for the RSS output. Student answers may vary due to different selection of degrees for testing. Award full credits for any valid answer.*

Here I choose degrees 1,4,7,10. The polynomial fits are

FIGURE 2. Plot for 3b



My RSS output is

```
[1] "Degree: 1 , RSS: 678511.381056838"
[1] "Degree: 4 , RSS: 613911.249278815"
[1] "Degree: 7 , RSS: 574388.639735982"
[1] "Degree: 10 , RSS: 573400.41709661"
```

(c) *Grading notes for part (c): 3 points for the correct degree selection, grade by completion. Also, students' answers may vary due to different selection of degrees for testing. Award full credits for any valid answer. The plot is not necessary.*

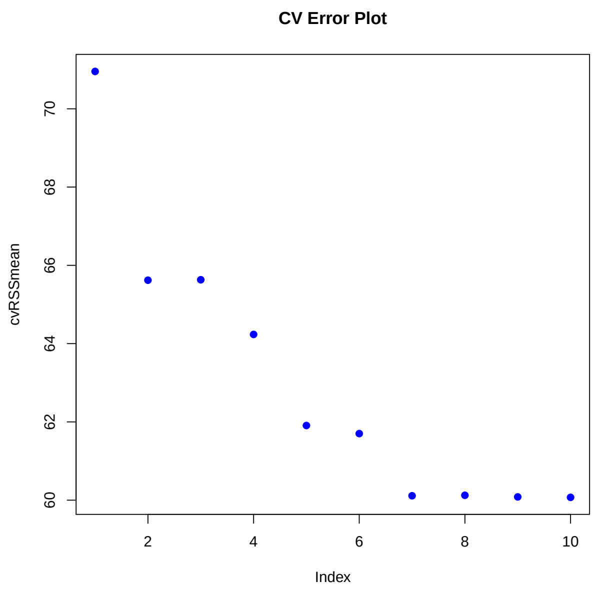
Here we use set.seed(2017). The optimal degree is 10 and the CV error rate plot is as following.

(d) *Grading notes for part (d): 1 points for the plot, 2 points for the output, 1 point for explaining how the knots are chosen.*

The summary output and plot are as following.

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

Residuals:
    Min      1Q  Median      3Q     Max
-39.891  -5.080  -0.132   5.310  37.490

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
```

FIGURE 3. Plot for 3c



```
(Intercept)  475.896       1.917  248.23  < 2e-16 ***
bsX1          17.647       2.454    7.19 6.98e-13 ***
bsX2         -42.799       1.702  -25.15  < 2e-16 ***
bsX3         -32.735       2.243  -14.59  < 2e-16 ***
bsX4         -51.611       1.942  -26.57  < 2e-16 ***
---
Signif. codes:  0    ***    0.001    **    0.01    *    0.05    .    0.1         1

Residual standard error: 7.975 on 9563 degrees of freedom
Multiple R-squared:  0.7817,    Adjusted R-squared:  0.7816
F-statistic:  8562 on 4 and 9563 DF,  p-value: < 2.2e-16
```

I let bs() choose the knots by specifying df. The default is two knots at the endpoints and additional knots that are evenly spaced quantiles.

(e) *Grading notes for part (e): 2 points for the plot, 2 points for the RSS output, 1 point for valid desciption. Student answers may vary due to different selection of degrees of freedom for testing. Award full credits for any valid answer.*

Here I choose degrees of freedom 3,4,5,6,10,12,20,25,30 for testing. The fits are My RSS output is

```
[1] "Degrees of Freedom: 3 , RSS: 626945.791546589"
[1] "Degrees of Freedom: 4 , RSS: 608266.876846772"
[1] "Degrees of Freedom: 5 , RSS: 590751.029544271"
[1] "Degrees of Freedom: 6 , RSS: 588554.242043626"
[1] "Degrees of Freedom: 10 , RSS: 567032.885685266"
[1] "Degrees of Freedom: 12 , RSS: 563802.926835622"
[1] "Degrees of Freedom: 20 , RSS: 550736.124718745"
[1] "Degrees of Freedom: 25 , RSS: 548737.800092624"
[1] "Degrees of Freedom: 30 , RSS: 546587.537497312"
```

Looks like past 10df we have too much flexibility. 30df is very wiggly.

(f) *Grading notes for part (f): 2 points for the correct degree selection, grade by completion. Also, students' answers may vary due to different selection of degrees*

FIGURE 4. Plot for 3d

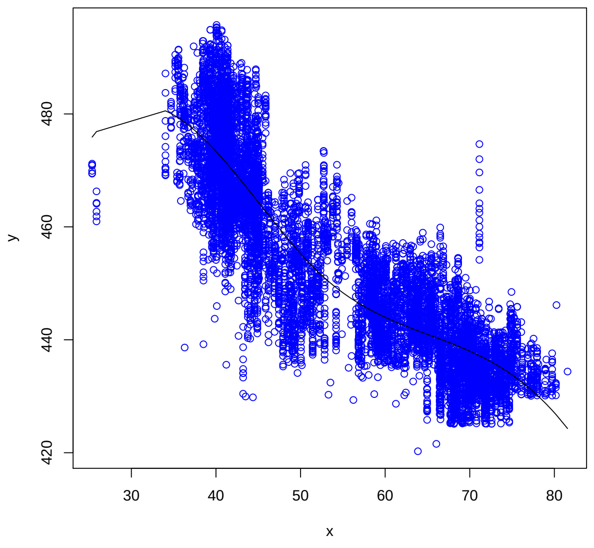
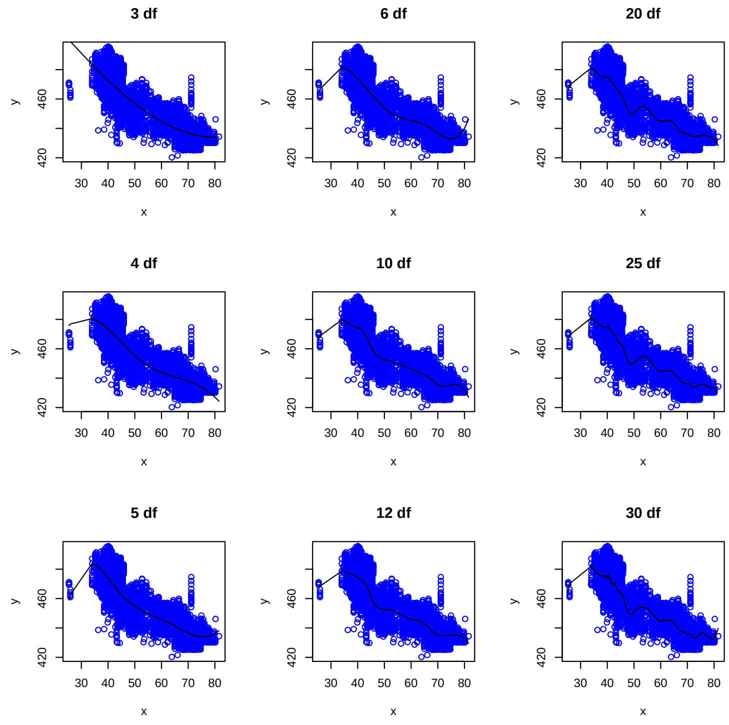


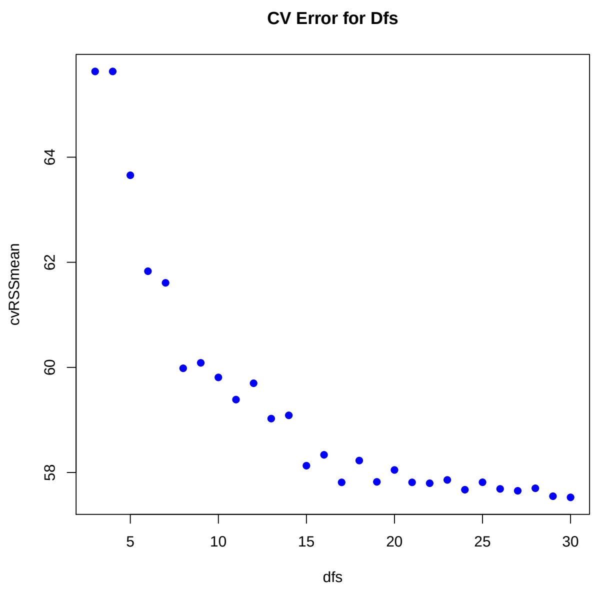
FIGURE 5. Plot for 3e



for testing. Award full credits for any valid answer. Answers should include df of at least 10.

Here we use set.seed(2017). The optimal degrees of freedom is 30 and the CV error rate plot is as following.

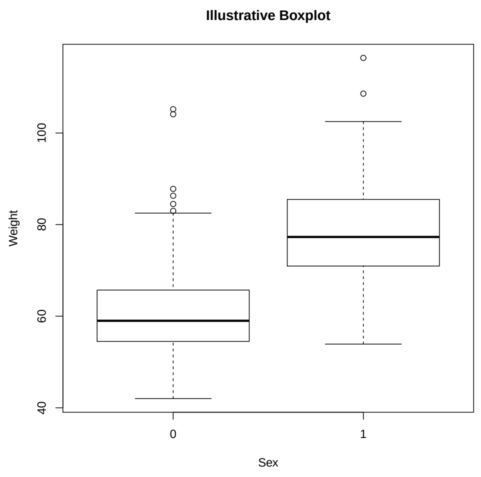
FIGURE 6. Plot for 3f



4. *This problem is worth 15 points: two points each for parts a, b, d, and f. Three points for e. Four points for c.*

   (a) We can recover the encoding of sex in many ways. A simple one would be to look at boxplots of height by sex, knowing that men are taller on average than women. This reveals that a sex value of 1 corresponds to men.

FIGURE 7. Boxplot for 4a



   (b) Although all of the variables are in the same units, we think that they have different meanings. A change of 1cm in my wrist girth has drastic implications on my size in the way that a change of 1cm in my chest girth does not. If we did not scale the variables, the ones with larger variance would control almost all of the pls and pcr fits.

(c) Here is the output from running summary():

For PLS:

```
Data:    X dimension: 307 21
         Y dimension: 307 1
Fit method: kernelpls
Number of components considered: 21

VALIDATION: RMSEP
Cross-validated using 10 random segments.
       (Intercept)  1 comps  2 comps  3 comps  4 comps  5 comps  6 comps  7 comps
CV           13.61    3.300    3.017    2.923    2.873    2.858    2.854    2.846
adjCV        13.61    3.299    3.015    2.918    2.861    2.847    2.843    2.834
       8 comps  9 comps  10 comps  11 comps  12 comps  13 comps  14 comps  15
           comps
CV       2.846    2.846     2.848     2.849     2.849     2.849     2.849
    2.849
adjCV    2.835    2.835     2.836     2.837     2.837     2.837     2.837
    2.837
       16 comps  17 comps  18 comps  19 comps  20 comps  21 comps
CV        2.849     2.849     2.849     2.849     2.849     2.849
adjCV     2.838     2.838     2.838     2.838     2.838     2.838

TRAINING: % variance explained
          1 comps  2 comps  3 comps  4 comps  5 comps  6 comps  7 comps  8 comps
X           64.75    74.16    79.87    81.47    84.06    86.96    88.74    89.89
Y$Weight    94.17    95.25    95.71    96.12    96.19    96.23    96.24    96.25
          9 comps  10 comps  11 comps  12 comps  13 comps  14 comps  15 comps
X           91.31     91.97     92.98     93.63     94.52     95.24     95.88
Y$Weight    96.25     96.25     96.25     96.25     96.25     96.25     96.25
          16 comps  17 comps  18 comps  19 comps  20 comps  21 comps
X            96.56     97.29     98.02     98.95     99.50    100.00
Y$Weight     96.25     96.25     96.25     96.25     96.25     96.25
```

For PCR:

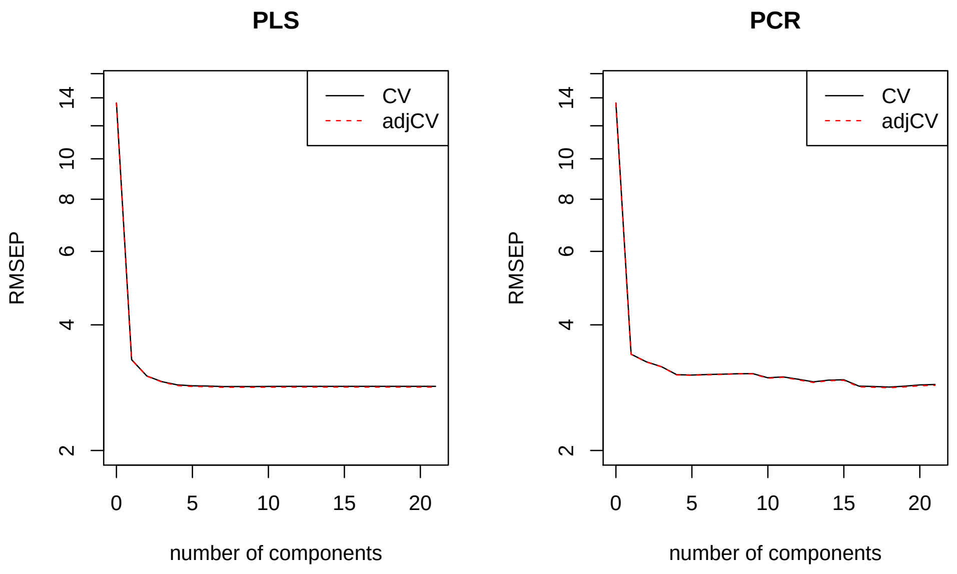
```
Data:    X dimension: 307 21
         Y dimension: 307 1
Fit method: svdpc
Number of components considered: 21

VALIDATION: RMSEP
Cross-validated using 10 random segments.
       (Intercept)  1 comps  2 comps  3 comps  4 comps  5 comps  6 comps  7 comps
CV           13.61    3.402    3.263    3.176    3.038    3.033    3.041    3.047
adjCV        13.61    3.400    3.260    3.173    3.035    3.028    3.037    3.042
       8 comps  9 comps  10 comps  11 comps  12 comps  13 comps  14 comps  15
           comps
CV       3.054    3.057     2.987     3.002     2.963     2.921     2.949
    2.954
adjCV    3.054    3.052     2.980     2.996     2.953     2.911     2.940
    2.945
       16 comps  17 comps  18 comps  19 comps  20 comps  21 comps
CV        2.853     2.847     2.838     2.852     2.872     2.878
adjCV     2.841     2.834     2.828     2.840     2.859     2.865

TRAINING: % variance explained
          1 comps  2 comps  3 comps  4 comps  5 comps  6 comps  7 comps  8 comps
X           64.76    75.83    80.48    84.85    87.20    89.13    90.53    91.88
Y$Weight    93.86    94.40    94.76    95.21    95.25    95.27    95.27    95.27
```

```
          9 comps  10 comps  11 comps  12 comps  13 comps  14 comps  15 comps
X           93.12     94.28     95.25     96.09     96.88     97.59     98.16
Y$Weight    95.36     95.58     95.58     95.74     95.80     95.81     95.85
          16 comps  17 comps  18 comps  19 comps  20 comps  21 comps
X            98.63     99.02     99.37     99.63     99.84    100.00
Y$Weight     96.13     96.20     96.22     96.24     96.24     96.25
```

As we can see, for a given number of components, pcr ends up with a better % variance explained of X and a worse % variance explained of Weight compared to pls. This is to be expected because pcr optimizes for % variance explained of X while pls optimizes for % variance explained of X AND Weight.

(d) To decide on a number of components to use in each model I examined the cross validated error:

FIGURE 8. CV Err for 4d



By examining that plot and the output of the summaries (which contains the CV error as well), I decided on using 7 components for pls and 18 components for pcr. These were the models with minimal CV error. Model selection can also be done with a 1-se rule or even potentially by sight as it seems that more than 2-3 components does not cause much improvement. Note that there are many correct answers on this problem depending on random seeds (including seeds used for the cross validation).

(e) Unfortunately, both of these methods still require all 21 measurements. That is because even though we only need 7 components for pls, the components themselves are constructed from the original 21 variables. Instead, I did a forward stepwise regression and selected the model that minimizes C_p. This resulted in selecting 13 components.

(f) My pls model had a test error around 7.77, pcr had 8.21, and forward stepwise 7.73. It seems like pcr is doing a little worse than the other two models.

5. *Grading notes: this problem is worth 20 points.*
(a,b): 1 point each. (c): 0 points. (d,e,g,h,i,j): 2 points each. (f,k): 3 points each.

(a) The summary should be

```
Regression tree:
tree(formula = train.y ~ ., data = train.X)
Variables actually used in tree construction:
 [1] "Onset.Delta"             "last.alsfrs.score"       "mean.climbing.stairs"
 [4] "alsfrs.score.slope"      "max.slope.salivation"    "last.slope.weight"
 [7] "min.slope.alsfrs.score" "sd.slope.alsfrs.score"   "slope.fvc.liters.slope"
[10] "last.slope.fvc.liters"
Number of terminal nodes:  13
Residual mean deviance:  0.2264 = 268 / 1184
Distribution of residuals:
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
-1.9920 -0.2619  0.0560  0.0000  0.3336  1.8380
```

(b) The tree is plotted below.

(c) There is nothing to grade.

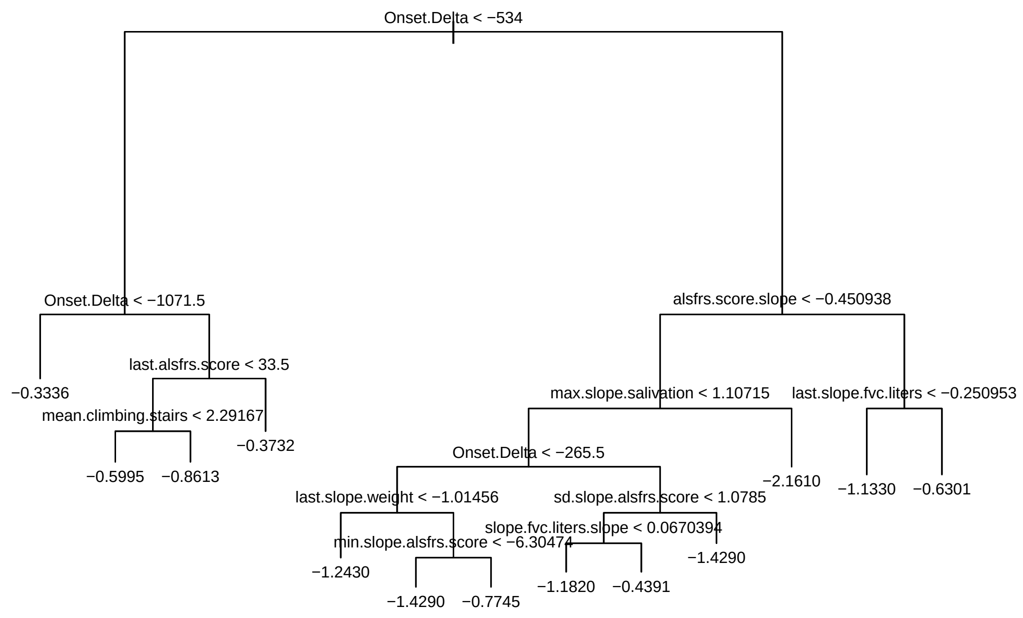
(d) The figure is shown below.

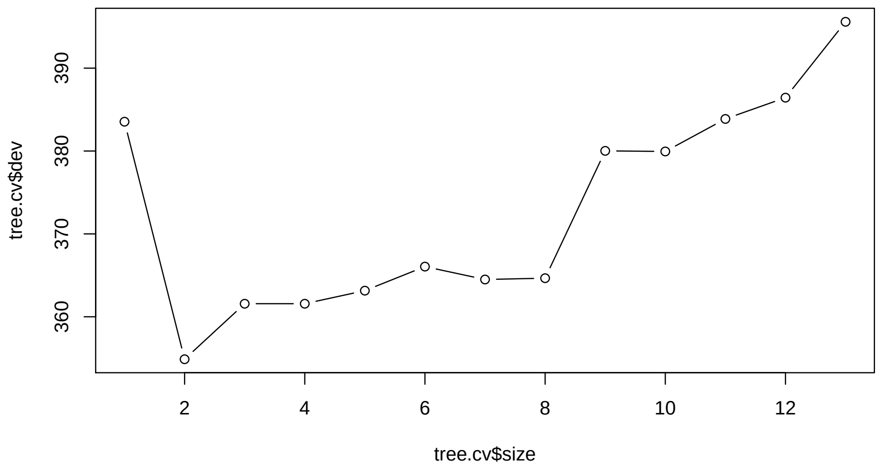
(e) The best pruned tree has size 2. The figure is shown below.

(f) The full tree has RMSE 0.572. The pruned tree has MSRE 0.543, which is lower. Indeed, this is to be expected, since it is likely the full tree is overfitting. The Lasso (0.521, from problem set 2) still performs better than both.

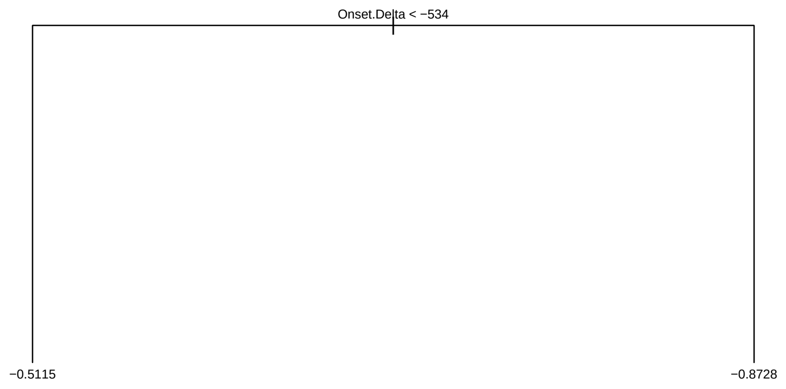
(g) The output is

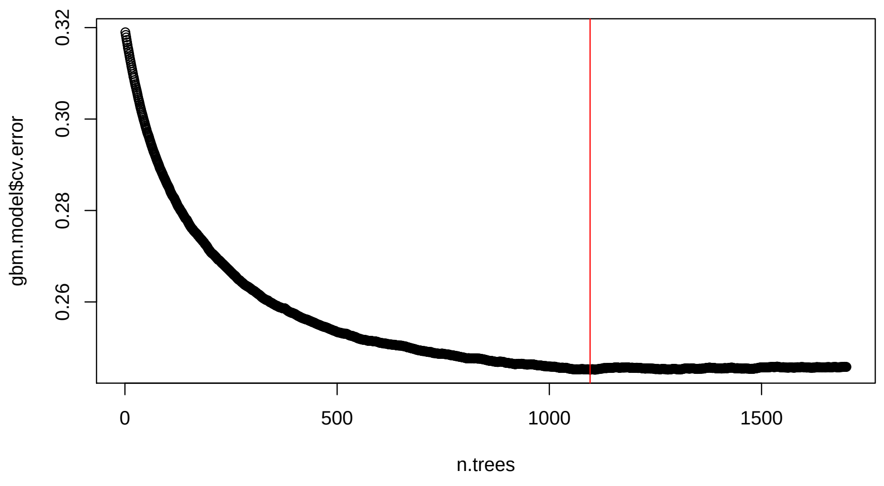
```
gbm(formula = train.y ~ ., distribution = "gaussian", data = train.X,
n.trees = max.trees, shrinkage = 0.01, cv.folds = 5)
A gradient boosted model with gaussian loss function.
1700 iterations were performed.
The best cross-validation iteration was 1428.
There were 323 predictors of which 193 had non-zero influence.
```


Onset.Delta < −534
Onset.Delta < −1071.5
alsfrs.score.slope < −0.450938
−0.3336
last.alsfrs.score < 33.5
mean.climbing.stairs < 2.29167
−0.3732
−0.5995
−0.8613
max.slope.salivation < 1.10715
last.slope.fvc.liters < −0.250953
−2.1610
−1.1330
−0.6301
Onset.Delta < −265.5
last.slope.weight < −1.01456
sd.slope.alsfrs.score < 1.0785
slope.fvc.liters.slope < 0.0670394
min.slope.alsfrs.score < −6.30474
−1.2430
−1.4290
−0.7745
−1.1820
−0.4391
−1.4290



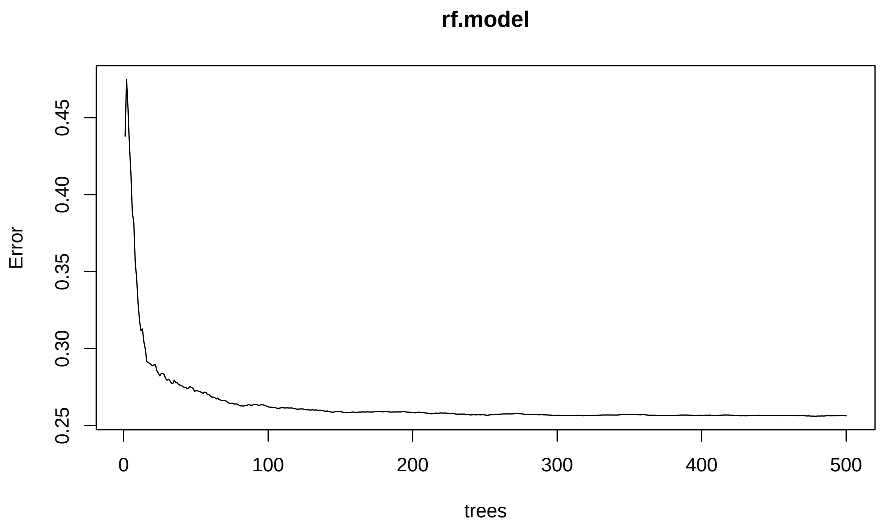
(h) The number of trees that minimize the CV error is 1096. The picture is:





(i) The test RMSE is 0.512.

(j) We obtain the plot below.



(k) The test RMSE is 0.512. It is similar to boosting, and better than decision trees, as we expected.

Appendix: Relevant R code

```
# Problem 3
rm(list=ls())
library(MASS)

# Part a
pplant<-read.csv("power_plant.csv")
x = pplant$V
y = pplant$PE
fitCubic = lm(y ~ poly(x,3))
sink("../output/prob3a.txt", append=FALSE, split=FALSE)
summary(fitCubic)
sink()

myx = list(x = seq(min(x)*.9, max(x)*1.1, length.out=1000))
pdf("../figures/prob3a.pdf", width = 6, height = 5)
plot(x, y, col="blue")
lines(myx$x, predict(fitCubic, newdata=myx))
dev.off()

# Part b
sink("../output/prob3b.txt", append=FALSE, split=FALSE)
pdf("../figures/prob3b.pdf")
yHat = predict.lm(fitCubic, newdata=poly(x,3))
plot(x, y, pch=1,cex=1,col="blue")
degLs = seq(1,10,3)
RSS = rep(0, length(degLs))
for( d in degLs) {
  fitPoly = lm(y ~ poly(x,d))
  yHat = predict.lm(fitPoly, newdata=poly(x,d))
  lines(sort(x), yHat[order(x)], col=d+1,lwd=2)
  RSS[d] = sum(fitPoly$residuals^2)
  print(paste("Degree:",d,", RSS:",RSS[d]))
}
legend('topright', legend=degLs,lty=1,lwd=2,col=(degLs+1)/2,title="Degree")
sink()
dev.off()

# Part c
# Choose degree with 10-fold cross validation
set.seed(2017)
K = 10
testFolds = rep(1:K, length.out=length(x))
one.fold <- function(fold, x, y, testFolds) {
  testInd = which(testFolds==fold)
  testData  = data.frame(x=x[testInd], y=y[testInd])
  trainData = data.frame(x=x[-testInd], y=y[-testInd])
  testRSS = sapply(1:10, function(d){
    fitPoly = lm(y ~ poly(x,d), data=trainData)
    yHat = predict(fitPoly, newdata=testData)
    testRSS = mean((yHat-testData$y)^2)
  })
}
cvRSS = sapply(1:K, one.fold, x,y,testFolds)
cvRSSmean = rowMeans(cvRSS)
which.min(cvRSSmean)
pdf("../figures/prob3c.pdf")
plot(cvRSSmean,col="blue",pch=19,main="CV Error Plot")
dev.off()

# Part d
require(splines)
bsX = bs(x, df = 4)
myspline = lm(y ~ bsX)
sink("../output/prob3d.txt", append=FALSE, split=FALSE)
summary(myspline)
sink()
# We can choose the number of knots by setting df, or knots directly.
# This uses equally spaced quantiles as knots.
pdf("../figures/prob3d.pdf")
plot(x, y, col="blue")
lines(sort(x), predict(myspline, newdata=bs(x, df = 4))[order(x)])
dev.off()

# Part e
sink("../output/prob3e.txt", append=FALSE, split=FALSE)
pdf("../figures/prob3e.pdf")
par(mfcol = c(3,3))
myRSS = numeric(9)
dfs = c(3,4,5,6,10,12,20,25,30)
for(ii in 1:length(dfs)) {
  bsX = bs(x, df = dfs[ii])
  myspline = lm(y ~ bsX)
  myRSS[ii] = sum(myspline$residuals^2)
  plot(x, y, col="blue", main=paste(dfs[ii], "df"))
  lines(sort(x), predict(myspline, newdata=bs(x, df = dfs[ii]))[order(x)])
  print(paste("Degrees of Freedom:",dfs[ii],", RSS:",myRSS[ii]))
}
sink()
dev.off()
```

```
# Part f
# Choose spline df with 10-fold cross validation
set.seed(2017)
K = 10
dfs = 3:30
testFolds = rep(1:K, length.out=length(x))
one.fold <- function(fold, x, y, testFolds) {
  testInd = which(testFolds==fold)
  testData  = data.frame(x=x[testInd], y=y[testInd])
  trainData = data.frame(x=x[-testInd], y=y[-testInd])
  testRSS = sapply(dfs, function(d){
    trainData$bsX = bs(trainData$x, df = d-1)
    testData$bsX = predict(trainData$bsX, newx = testData$x)
    fitSpline = lm(y ~ bsX, data = trainData)
    yHat = predict(fitSpline, newdata = testData)
    testRSS = mean((yHat-testData$y)^2)
  })
}
cvRSS = sapply(1:K, one.fold, x,y, testFolds)
cvRSSmean = rowMeans(cvRSS)
dfs[which.min(cvRSSmean)]

pdf("../figures/prob3f.pdf")
plot(dfs, cvRSSmean,col="blue",pch=19,main="CV Error for Dfs")
dev.off()



# Problem 4
rm(list=ls()) # Clear workspace

# Part a
load('body.RData')
pdf("../figures/prob4a.pdf")
with(Y, boxplot(Weight~Sex))
title(main = "Illustrative Boxplot", xlab = "Sex", ylab="Weight")
dev.off()

# Part b
set.seed(2017)
test = sample(1:nrow(X), 200)
train = (1:nrow(X))[-test]
library(pls)
pls.fit = plsr(Y$Weight ~ .,data = X, scale = TRUE, validation="CV", subset=train)
pcr.fit = pcr(Y$Weight ~ .,data = X, scale = TRUE, validation="CV", subset=train)
```

```
# Part c
sink("../output/4.c.pls.txt", append=FALSE, split=FALSE)
summary(pls.fit)
sink()
sink("../output/4.c.pcr.txt", append=FALSE, split=FALSE)
summary(pcr.fit)
sink()
closeAllConnections()

# Part d
pdf("../figures/prob4d.pdf", width = 8, height = 5)
par(mfcol=c(1,2))
validationplot(pls.fit, val.type="RMSEP", legendpos="topright", main="PLS", ylim=c(2,15),
validationplot(pcr.fit, val.type="RMSEP", legendpos="topright", main="PCR", ylim=c(2,15),
dev.off()
ncomp.pls = 7
ncomp.pcr = 18

# Part e
require(leaps)
reg.fit = regsubsets(Y$Weight~., data=X, method="forward", nvmax = 21, subset=train)
reg.sum = summary(reg.fit)
myid = which.min(reg.sum$cp)

predict.regsubsets = function(object, newdata, id, ...) {
  newdata = cbind(`(Intercept)` = 1, newdata)
  coefi = coef(object, id = id)
  do.call(cbind, newdata[, names(coefi)]) %*% coefi
}

# Part f
mean((Y$Weight[test] - predict(pls.fit, newdata = X[test,], id = ncomp.pls))^2)
mean((Y$Weight[test] - predict(pcr.fit, newdata = X[test,], id = ncomp.pcr))^2)
mean((Y$Weight[test] - predict(reg.fit, newdata = X[test,], id = myid))^2)




# Problem 5
rm(list=ls()) # Clear workspace

# Part a (Fit the tree)
load('als.RData')
library(tree)
full.tree.model <- tree(train.y ~ ., train.X)
sink("../output/5.a.txt", append=FALSE, split=FALSE)
```

```
summary(full.tree.model)
sink()
closeAllConnections()

# Part b (Plot the tree)
pdf("../figures/5_b.pdf", width = 8, height = 6)
plot(full.tree.model)
text(full.tree.model, cex = 0.65)
dev.off()

# Part c (Prune the tree)
set.seed(2017)
tree.cv <- cv.tree(full.tree.model)

# Part d (plot the tree)
pdf("../figures/5_d.pdf", width = 8, height = 5)
plot(tree.cv$size, tree.cv$dev, type= 'b')
dev.off()

# Part e
best.tree.size <- tree.cv$size[which.min(tree.cv$dev)]
best.tree.size
prune.tree.model <- prune.tree(full.tree.model, best = best.tree.size)
pdf("../figures/5_e.pdf", width = 8, height = 5)
plot(prune.tree.model)
text(prune.tree.model, cex = .7)
dev.off()

# Part f
full.tree.test.pred <- predict(full.tree.model, test.X)
sqrt(mean((full.tree.test.pred - test.y)^2))
prune.tree.test.pred <- predict(prune.tree.model, test.X)
sqrt(mean((prune.tree.test.pred - test.y)^2))

# Part g (boosting)
library(gbm)
set.seed(2017)
max.trees <- 1700
gbm.model <- gbm(train.y ~ ., data = train.X, distribution = "gaussian",
                 n.trees= max.trees, shrinkage = 0.01, cv.folds = 5)
print(gbm.model)

# Part h
best.n.trees <- which.min(gbm.model$cv.error)
best.n.trees
pdf("../figures/5_h.pdf", width = 8, height = 5)
```

```
plot(gbm.model$cv.error, xlab = "n.trees")
abline(v = best.n.trees, col = "red")
dev.off()

# Part i
gbm.test.pred <- predict(gbm.model, test.X, n.trees = best.n.trees)
sqrt(mean((gbm.test.pred - test.y)^2))

# Part j (random forests)
library(randomForest)
set.seed(2017)
rf.model<-randomForest(x = train.X, y = train.y, mtry = 80, importance = TRUE)
pdf("../figures/5_j.pdf", width = 8, height = 5)
plot(rf.model)
dev.off()

# Part k
rf.test.pred<-predict(rf.model, test.X)
sqrt(mean((rf.test.pred - test.y)^2))
```