STAT 6340 Mini Project 4

Matthew Lynn

March 27, 2019

Section 1

Section 2 is coded in Section 1

Question 1(a)

We make a scatterplot of gpa against act.
We notice there is a positve correlation but it is not strong (0.27)

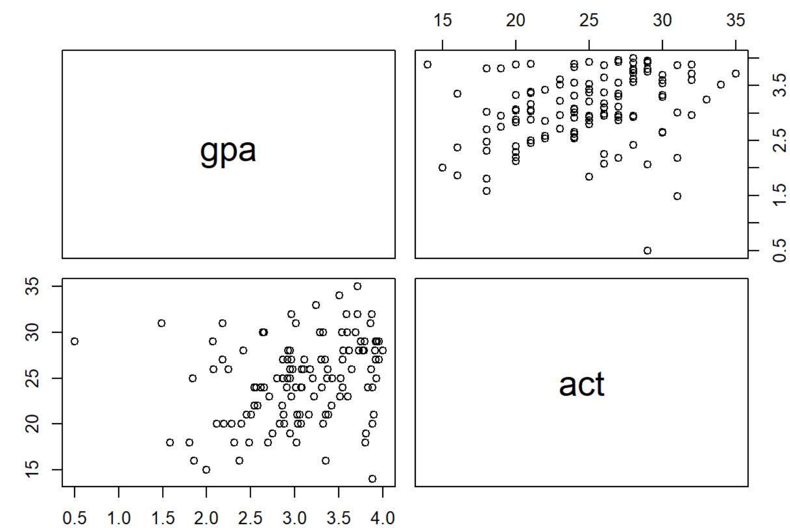
```
gpa = read.csv("gpa.csv", header = T)
str(gpa)
```

```
## 'data.frame':    120 obs. of  2 variables:
##  $ gpa: num  3.9 3.88 3.78 2.54 3.03 ...
##  $ act: int  21 14 28 22 21 31 32 27 29 26 ...
```

```
summary(gpa)
```

```
##       gpa             act
##  Min.   :0.500   Min.   :14.00
##  1st Qu.:2.689   1st Qu.:21.00
##  Median :3.078   Median :25.00
##  Mean   :3.074   Mean   :24.73
##  3rd Qu.:3.593   3rd Qu.:28.00
##  Max.   :4.000   Max.   :35.00
```

```
pairs(subset(gpa))
```




```
round(cor(subset(gpa)), 2)
```

```
##      gpa  act
## gpa 1.00 0.27
## act 0.27 1.00
```

```
# There is a positive correlation between gpa and act
# the relationship is not very strong (0.27)
```

1(b)

ρ is our population correlation between gpa and act.
Here, we gather the bootstrap estimates of bias and standard error of
the point estimate and show the 95% confidence interval from bootstrap

```
library(boot)
corr(gpa)
```

```
## [1] 0.2694818
```

```
# either function below will work
corr.fn <- function(data, i=c(1:length(data))) {
  result <- data[i,]
  return(cor(result$gpa, result$act))
}
corr.fn <- function(data, i=c(1:length(data))) {
  result = corr(data[i,])
  return(result)
}
set.seed(1)
corr.boot <- boot(gpa, corr.fn, R = 1000)
# Point estimate, bootstrap bias, and bootstrap se
corr.boot
```

```
##
## ORDINARY NONPARAMETRIC BOOTSTRAP
##
##
## Call:
## boot(data = gpa, statistic = corr.fn, R = 1000)
##
##
## Bootstrap Statistics :
##      original      bias    std. error
## t1* 0.2694818 0.007800885   0.1072831
```

```
# 95% bootstrap conf int
boot.ci(corr.boot, type = "perc")
```

```
## BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
## Based on 1000 bootstrap replicates
##
## CALL :
## boot.ci(boot.out = corr.boot, type = "perc")
##
## Intervals :
## Level     Percentile
## 95%   ( 0.0728,  0.4917 )
## Calculations and Intervals on Original Scale
```

```
# verified below
sort(corr.boot$t)[c(25, 975)]
```

```
## [1] 0.07274536 0.49136740
```

```
# our original ρ is contained in the bootstrap interval and the bias is only
# 0.004196114 which implies good bootstrap estimates.
```

1(c)

Using gpa~act, we fit a SLR. We show the least square estimates, standard errors and a 95% CI. We include verifications of our model assumptions

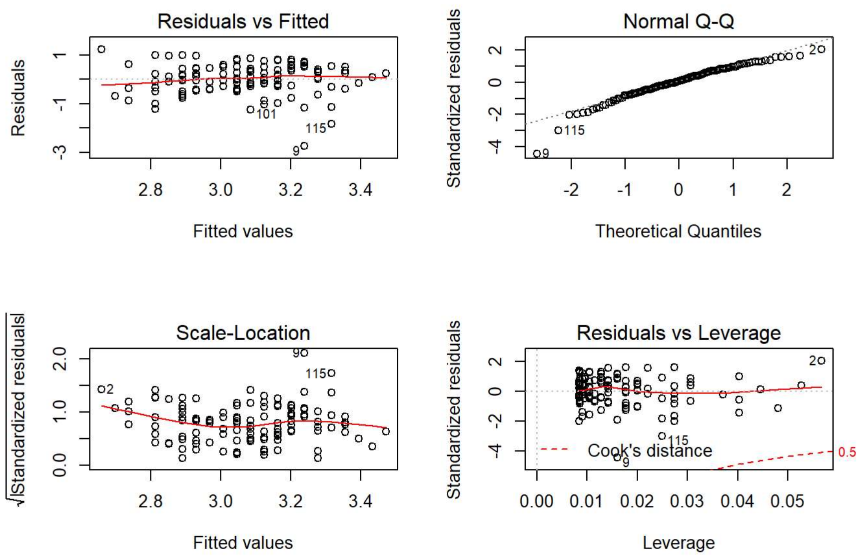
```
fit = lm(gpa~act, data = gpa)
# least square estimate of coefficients, SE, and 95% CI
summary(fit)
```

```
##
## Call:
## lm(formula = gpa ~ act, data = gpa)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -2.74004 -0.33827  0.04062  0.44064  1.22737
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  2.11405    0.32089   6.588  1.3e-09 ***
## act          0.03883    0.01277   3.040  0.00292 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.6231 on 118 degrees of freedom
## Multiple R-squared:  0.07262,    Adjusted R-squared:  0.06476
## F-statistic:  9.24 on 1 and 118 DF,  p-value: 0.002917
```

```
confint(fit)
```

```
##                  2.5 %     97.5 %
## (Intercept) 1.47859015 2.74950842
## act         0.01353307 0.06412118
```

```
par(mfrow = c(2,2))
plot(fit)
```




```
par(mfrow = c(1,1))
# it appears the residual plot could do better via a transformation but not bad
# the qq plot is not bad but does show some outliers.
```

1(d)

We use a nonparametric bootstrap to estimate the previous question's parameters
The bootstrap estimates are higher than was in the linear model
Also, the 95% CI is wider than the linear model

```
library(boot)
# function to output coefficients from linear model
fit.fn <- function(data, index) {
  result <- coef(lm(gpa ~ act , data = gpa, subset = index))
  return(result)
}
set.seed(1)
fit.boot = boot(gpa, fit.fn, R = 1000)
fit.boot
```

```
##
## ORDINARY NONPARAMETRIC BOOTSTRAP
##
##
## Call:
## boot(data = gpa, statistic = fit.fn, R = 1000)
##
##
## Bootstrap Statistics :
##       original        bias    std. error
## t1* 2.11404929 -0.0108361368  0.35962727
## t2* 0.03882713  0.0004350197  0.01455027
```

```
boot.ci(fit.boot, type = "perc", index = 1) # intercept
```

```
## BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
## Based on 1000 bootstrap replicates
##
## CALL :
## boot.ci(boot.out = fit.boot, type = "perc", index = 1)
##
## Intervals :
## Level     Percentile
## 95%   ( 1.387,  2.812 )
## Calculations and Intervals on Original Scale
```

```
boot.ci(fit.boot, type = "perc", index = 2) # act
```

```
## BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
## Based on 1000 bootstrap replicates
##
## CALL :
## boot.ci(boot.out = fit.boot, type = "perc", index = 2)
##
## Intervals :
## Level     Percentile
## 95%   ( 0.0110,  0.0674 )
## Calculations and Intervals on Original Scale
```

Question 2(a)

We want to examine StoreID, STORE and Store7 variables form the OJ data.
After a short EDA we determine that StoreID seems to contain all the info
in STORE and Store7. Thus, we can drop STORE and Store7.
We then use Purchase as our response, take StoreID as categorical, and split
our data into a train set and test set for training and predicting

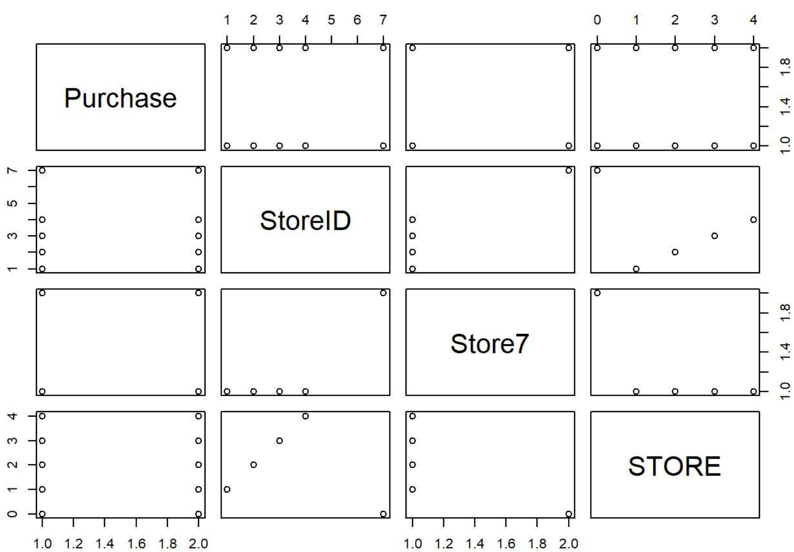
```r
library(ISLR)
names(OJ)
```

```
##  [1] "Purchase"       "WeekofPurchase" "StoreID"        "PriceCH"
##  [5] "PriceMM"        "DiscCH"         "DiscMM"         "SpecialCH"
##  [9] "SpecialMM"      "LoyalCH"        "SalePriceMM"    "SalePriceCH"
## [13] "PriceDiff"      "Store7"         "PctDiscMM"      "PctDiscCH"
## [17] "ListPriceDiff"  "STORE"
```

```r
# use only these 4 variables
OJ = subset(OJ, select = c(Purchase, StoreID, Store7, STORE))
str(OJ)
```

```
## 'data.frame':    1070 obs. of  4 variables:
##  $ Purchase: Factor w/ 2 levels "CH","MM": 1 1 1 2 1 1 1 1 1 1 ...
##  $ StoreID : num  1 1 1 1 7 7 7 7 7 7 ...
##  $ Store7  : Factor w/ 2 levels "No","Yes": 1 1 1 1 2 2 2 2 2 2 ...
##  $ STORE   : num  1 1 1 1 0 0 0 0 0 0 ...
```

```r
# STORE is a subset of StoreID and Store7 is transitive through StoreID
pairs(OJ)
```




```r
fit = glm(Purchase~., family = binomial, data = OJ)
summary(fit)
```

```
##
## Call:
## glm(formula = Purchase ~ ., family = binomial, data = OJ)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.2736  -0.9733  -0.7236   1.1852   1.7136
##
## Coefficients: (1 not defined because of singularities)
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)  0.46471    0.19347   2.402   0.0163 *
## StoreID     -0.24147    0.07339  -3.290   0.0010 **
## Store7Yes    0.01916    0.36632   0.052   0.9583
## STORE             NA         NA      NA       NA
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1430.9  on 1069  degrees of freedom
## Residual deviance: 1360.4  on 1067  degrees of freedom
## AIC: 1366.4
##
## Number of Fisher Scoring iterations: 4
```

```r
rm(OJ)
# It appears that only StoreID and factor level "No" (intercept) from
# Store7 have any significance when all predictors are used.
# we drop Store7 and STORE
newOJ = subset(OJ, select = c(-Store7, -STORE))
# I do not see any reason to force MM = 1 and CH = 0
# When I experimented, I was able to produce the same results
newOJ$StoreID = as.factor(newOJ$StoreID) # take StoreID as factor
# str(newOJ) # uncomment to verify is factor
# split data 50/50, train/test
set.seed(1)
n = nrow(newOJ)
sampler = sample(1:n, n/2) # n/2 is the 50/50 splitter
train = newOJ[sampler, ]
test = newOJ[-sampler, ]
```

2(b)

Here we train a logisitc regression model and use the confusion matrix
to obtain sensitivity, specificity, overall misclassification, then
plot the ROC curve and estimate using a 10 fold cross validation
The caret package does all of this nicely and compactly, so we proceed

through the next few questions using similar coding.
The accuracy from our model (train data) has accuracy (1-test error) very close to our confusion matrix accuracy (test data, predicted)
The ROC curve has a nice left corner shape, notice that the graph continues to the left til -0.5, needs to be fixed to origin preferably (sorry!)
pROC shapes it this way with asp = 1 to keep the shape square

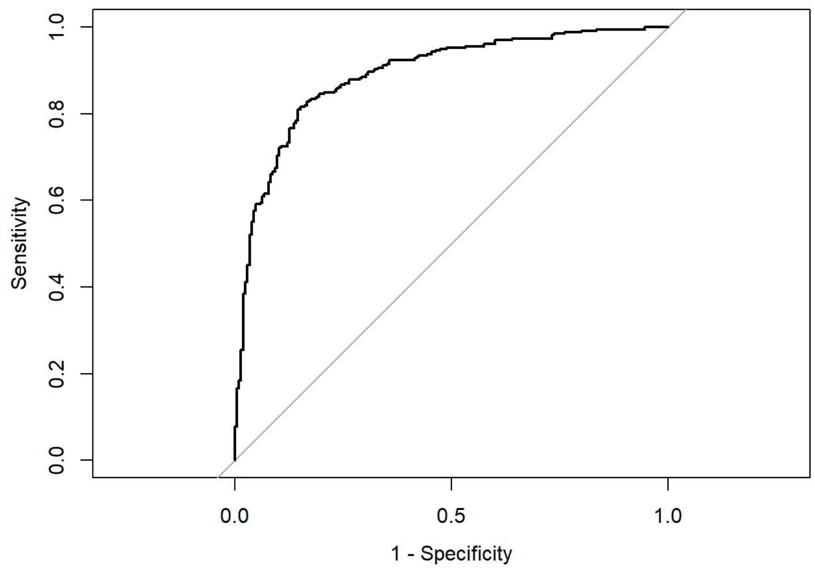
```
# 10 fold cv
  # library(crossval) # could not make sense of its inner workings
  # Even their examples needed tailoring, and even after that
  # it was not very intuitive.  I instead proceed with Caret package.
  # When using K-fold CV, we use the original data set unsplit.
  # However, I split the data beforehand so I could make a predict set
  # with the test data.  But in general this package's function should
  # do its own K splits as it trains/tests on the data.
  # A rather robust way of estimating accuracy (or test error rate)
library(caret)
set.seed(1)
tc <- trainControl(method = "cv", number = 10, verboseIter = F)
model <- train(Purchase~., train,
               method="glm", family="binomial", trControl = tc)
model
```

```
## Generalized Linear Model
##
## 535 samples
##  15 predictor
##   2 classes: 'CH', 'MM'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 481, 482, 481, 481, 482, 482, ...
## Resampling results:
##
##   Accuracy   Kappa
##   0.8204403  0.6202817
```

```
# 10-fold CV estimated test error rate = 1 - overall misclassification
# also, this is 1 - accuracy = test error rate (estimated)
glm.err = 1 - model$results[,2]
prob = predict(model, test, type = "prob")
# pred = predict(model, test) # or
pred <- ifelse(prob$MM >= 0.5, "MM", "CH")
con.mat = table(test[, "Purchase"], pred)
confusionMatrix(con.mat)
```

```
## Confusion Matrix and Statistics
##
##     pred
##       CH  MM
##   CH 287  43
##   MM  53 152
##
##                Accuracy : 0.8206
##                  95% CI : (0.7854, 0.8522)
##     No Information Rate : 0.6355
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.6169
##  Mcnemar's Test P-Value : 0.3583
##
##             Sensitivity : 0.8441
##             Specificity : 0.7795
##          Pos Pred Value : 0.8697
##          Neg Pred Value : 0.7415
##              Prevalence : 0.6355
##          Detection Rate : 0.5364
##    Detection Prevalence : 0.6168
##       Balanced Accuracy : 0.8118
##
##        'Positive' Class : CH
##
```

```
library(pROC)
roc <- roc(test[, "Purchase"], prob$MM, levels = c("MM","CH"))
plot(roc, legacy.axes = T)
```



2(c)

We employ the same methods but using LDA instead

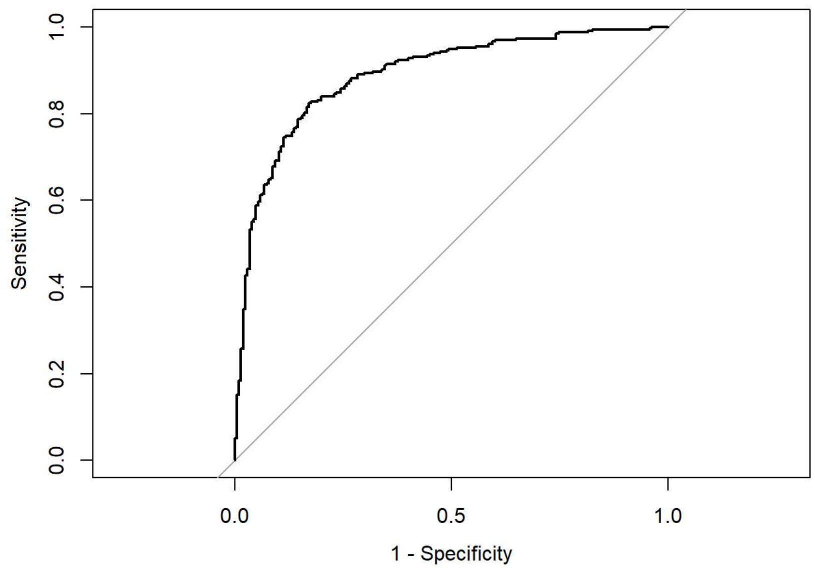
```
library(caret)
set.seed(1)
model <- train(Purchase~., train,
               method="lda", trControl = tc)
model
```

```
## Linear Discriminant Analysis
##
## 535 samples
##  15 predictor
##   2 classes: 'CH', 'MM'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 481, 482, 481, 481, 482, 482, ...
## Resampling results:
##
##   Accuracy   Kappa
##   0.8205101  0.6215449
```

```
lda.err = 1 - model$results[,2]
prob = predict(model, test, type = "prob")
pred <- ifelse(prob$MM >= 0.5, "MM", "CH")
confusionMatrix(table(test$Purchase, pred))
```

```
## Confusion Matrix and Statistics
##
##     pred
##       CH  MM
##   CH 285  45
##   MM  52 153
##
##                Accuracy : 0.8187
##                  95% CI : (0.7834, 0.8504)
##     No Information Rate : 0.6299
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.6139
##  Mcnemar's Test P-Value : 0.5424
##
##             Sensitivity : 0.8457
##             Specificity : 0.7727
##          Pos Pred Value : 0.8636
##          Neg Pred Value : 0.7463
##              Prevalence : 0.6299
##          Detection Rate : 0.5327
##    Detection Prevalence : 0.6168
##       Balanced Accuracy : 0.8092
##
##        'Positive' Class : CH
##
```

```
library(pROC)
roc <- roc(test$Purchase, prob$MM, levels = c("MM","CH"))
plot(roc, legacy.axes = T)
```



2(d)

We employ the same methods but using QDA instead
QDA fails if there is rank deficiency, in other words we have columns that fully or partially (near fully) explain other columns. This means our data has linear dependency or multicollinearity. If we can't invert a matrix then we cannot use qda effectively or at all. Below is what would be the qda procedure if there were no multicollinearity. However, it will result in rank deficiency and ultimately fail.

```
# library(caret)
# set.seed(1)
# model <- train(Purchase~., train,
#                method="qda", trControl = tc)
# model
qda.err = "No good" # 1 - model$results[,2]
# prob = predict(model, test, type = "prob")
# pred <- ifelse(prob$MM >= 0.5, "MM", "CH")
# confusionMatrix(table(test$Purchase, pred))
# library(pROC)
# roc <- roc(test$Purchase, prob$MM, levels = c("MM","CH"))
# plot(roc, legacy.axes = T)
```

2(e)

We employ the same methods but using KNN, we also use caret
to find the optimal KNN k value for lowest test error rate

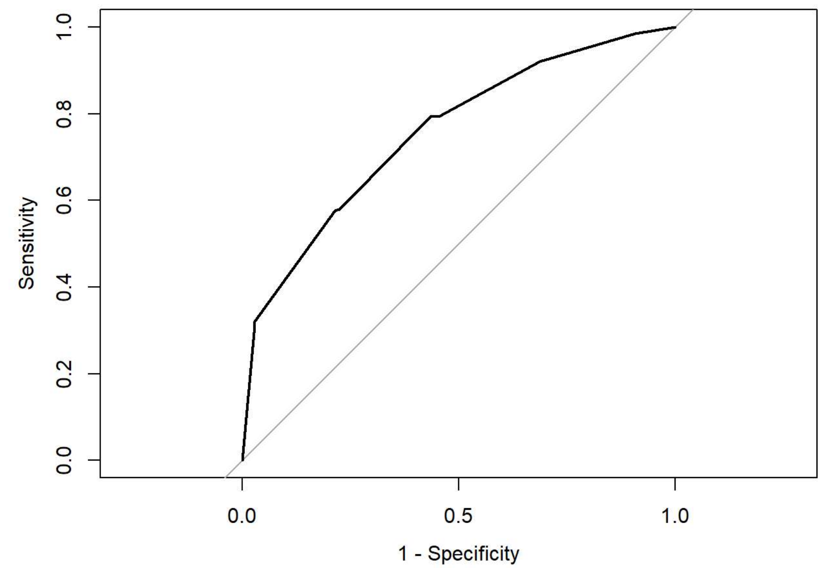
```
library(caret)
set.seed(1)
model <- train(Purchase~., train,
               method="knn", trControl = tc)
model # optimal k is 5, caret automatically uses this k
```

```
## k-Nearest Neighbors
##
## 535 samples
##  15 predictor
##   2 classes: 'CH', 'MM'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 481, 482, 481, 481, 482, 482, ...
## Resampling results across tuning parameters:
##
##   k  Accuracy   Kappa
##   5  0.6895178  0.3432383
##   7  0.6743885  0.3045327
##   9  0.6633124  0.2722348
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was k = 5.
```

```
knn.err = 1 - model$results[,2]
prob = predict(model, test, type = "prob")
pred <- ifelse(prob$MM >= 0.5, "MM", "CH")
confusionMatrix(table(test$Purchase, pred))
```

```
## Confusion Matrix and Statistics
##
##     pred
##       CH  MM
##   CH 262  68
##   MM  89 116
##
##                Accuracy : 0.7065
##                  95% CI : (0.666, 0.7448)
##     No Information Rate : 0.6561
##     P-Value [Acc > NIR] : 0.007396
##
##                   Kappa : 0.3669
##  Mcnemar's Test P-Value : 0.110450
##
##             Sensitivity : 0.7464
##             Specificity : 0.6304
##          Pos Pred Value : 0.7939
##          Neg Pred Value : 0.5659
##              Prevalence : 0.6561
##          Detection Rate : 0.4897
##    Detection Prevalence : 0.6168
##       Balanced Accuracy : 0.6884
##
##        'Positive' Class : CH
##
```

```r
library(pROC)
roc <- roc(test$Purchase, prob$MM, levels = c("MM","CH"))
plot(roc, legacy.axes = T)
```



2(f)

Now lets combine all the test error rates together and see which is the lowest
It seems that lda is our best model for 10 fold CV
KNN had nearly double of the other models

```r
rbind(glm.err, lda.err, qda.err, knn.er = min(knn.err))
```

```
##         [,1]
## glm.err "0.179559748427673"
## lda.err "0.179489867225716"
## qda.err "No good"
## knn.er  "0.310482180293501"
```

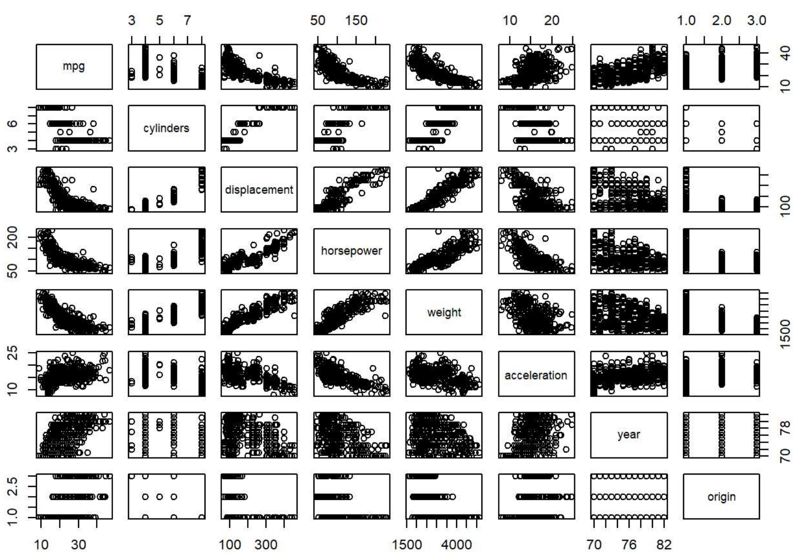
Question 3(a)

Let us dive into the Auto data from ISLR and use MPG as our response

```r
library(ISLR)
library(leaps)
set.seed(1)
auto = subset(Auto, select = -name)
# eda
str(auto)
```

```
## 'data.frame':    392 obs. of  8 variables:
##  $ mpg         : num  18 15 18 16 17 15 14 14 14 15 ...
##  $ cylinders   : num  8 8 8 8 8 8 8 8 8 8 ...
##  $ displacement: num  307 350 318 304 302 429 454 440 455 390 ...
##  $ horsepower  : num  130 165 150 150 140 198 220 215 225 190 ...
##  $ weight      : num  3504 3693 3436 3433 3449 ...
##  $ acceleration: num  12 11.5 11 12 10.5 10 9 8.5 10 8.5 ...
##  $ year        : num  70 70 70 70 70 70 70 70 70 70 ...
##  $ origin      : num  1 1 1 1 1 1 1 1 1 1 ...
```

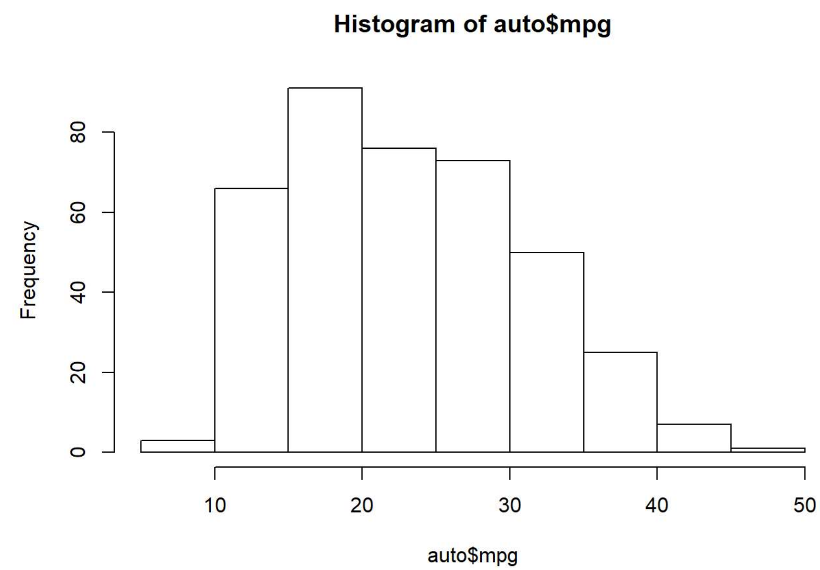
```r
# summary(auto) # nice way to get some quick stats, but messy so I commented
# Just looking at the scatterplots we see that mpg has many trends with others
pairs(subset(auto))
```




```r
cor(auto)
```

```
##                     mpg  cylinders displacement horsepower     weight
## mpg           1.0000000 -0.7776175   -0.8051269 -0.7784268 -0.8322442
## cylinders    -0.7776175  1.0000000    0.9508233  0.8429834  0.8975273
## displacement -0.8051269  0.9508233    1.0000000  0.8972570  0.9329944
## horsepower   -0.7784268  0.8429834    0.8972570  1.0000000  0.8645377
## weight       -0.8322442  0.8975273    0.9329944  0.8645377  1.0000000
## acceleration  0.4233285 -0.5046834   -0.5438005 -0.6891955 -0.4168392
## year          0.5805410 -0.3456474   -0.3698552 -0.4163615 -0.3091199
## origin        0.5652088 -0.5689316   -0.6145351 -0.4551715 -0.5850054
##              acceleration       year     origin
## mpg             0.4233285  0.5805410  0.5652088
## cylinders      -0.5046834 -0.3456474 -0.5689316
## displacement   -0.5438005 -0.3698552 -0.6145351
## horsepower     -0.6891955 -0.4163615 -0.4551715
## weight         -0.4168392 -0.3091199 -0.5850054
## acceleration    1.0000000  0.2903161  0.2127458
## year            0.2903161  1.0000000  0.1815277
## origin          0.2127458  0.1815277  1.0000000
```

```r
# seems to be right skewed
hist(auto$mpg)
```




```r
# here we look at what would be worthy predictor variables.  However,
# we are using all of them and whittling down with variable selection methods.
```

3(b)

For this part we want to use MLR using least squares
To make this part short we simply fit all predictors as our full model
Then run a summary and refit using only the significant ones
Lastly, we compare the models using partial F test and see that
the reduced model is better.

```
# fit a MLR
fit = lm(mpg~., data = auto)
summary(fit)
```

```
##
## Call:
## lm(formula = mpg ~ ., data = auto)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -9.5903 -2.1565 -0.1169  1.8690 13.0604
##
## Coefficients:
##                Estimate Std. Error t value Pr(>|t|)
## (Intercept)  -17.218435   4.644294  -3.707  0.00024 ***
## cylinders     -0.493376   0.323282  -1.526  0.12780
## displacement   0.019896   0.007515   2.647  0.00844 **
## horsepower    -0.016951   0.013787  -1.230  0.21963
## weight        -0.006474   0.000652  -9.929  < 2e-16 ***
## acceleration   0.080576   0.098845   0.815  0.41548
## year           0.750773   0.050973  14.729  < 2e-16 ***
## origin         1.426141   0.278136   5.127 4.67e-07 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 3.328 on 384 degrees of freedom
## Multiple R-squared:  0.8215, Adjusted R-squared:  0.8182
## F-statistic: 252.4 on 7 and 384 DF,  p-value: < 2.2e-16
```

```
# lets refit using only significant values seen in summary
anova(fit, lm(mpg~.-cylinders-horsepower-acceleration, data = auto))
```

	Res.Df	RSS	Df	Sum of Sq	F	Pr(>F)
	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>
1	384	4252.213	NA	NA	NA	NA
2	387	4332.729	-3	-80.51617	2.423696	0.06542561

2 rows

```
# we reject the null saying they are equal thus we use the reduced model
fit = lm(mpg~.-cylinders-horsepower-acceleration, data = auto)
lm.ar2 = summary(fit)$adj.r.squared
```

3(c)

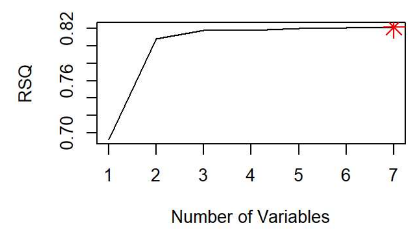
We use best subest variable selection method here to and use R-square alongside
Adjusted R-square to determine our model.

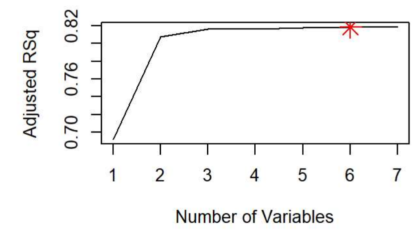
```
# best subset method

totpred <- ncol(auto) - 1
fit.full <- regsubsets(mpg~., auto, nvmax = totpred)
fit.summary <- summary(fit.full)
fit.summary
```

```
## Subset selection object
## Call: regsubsets.formula(mpg ~ ., auto, nvmax = totpred)
## 7 Variables  (and intercept)
##              Forced in Forced out
## cylinders        FALSE      FALSE
## displacement     FALSE      FALSE
## horsepower       FALSE      FALSE
## weight           FALSE      FALSE
## acceleration     FALSE      FALSE
## year             FALSE      FALSE
## origin           FALSE      FALSE
## 1 subsets of each size up to 7
## Selection Algorithm: exhaustive
##          cylinders displacement horsepower weight acceleration year origin
## 1  ( 1 ) " "       " "          " "        "*"    " "          " "  " "
## 2  ( 1 ) " "       " "          " "        "*"    " "          "*"  " "
## 3  ( 1 ) " "       " "          " "        "*"    " "          "*"  "*"
## 4  ( 1 ) " "       "*"          " "        "*"    " "          "*"  "*"
## 5  ( 1 ) " "       "*"          "*"        "*"    " "          "*"  "*"
## 6  ( 1 ) "*"       "*"          "*"        "*"    " "          "*"  "*"
## 7  ( 1 ) "*"       "*"          "*"        "*"    "*"          "*"  "*"
```

```
best.ar2 = mean(fit.summary$adjr2)

# Plot model fit measures for best model of each size against size
par(mfrow = c(2, 2))

# rsq
plot(fit.summary$rsq, xlab = "Number of Variables", ylab = "RSQ",
     type = "l")
point = as.numeric(which.max(fit.summary$rsq))
points(point, fit.summary$rsq[point],
       col = "red", cex = 2, pch = 8)
# Adjusted R^2
plot(fit.summary$adjr2, xlab = "Number of Variables", ylab = "Adjusted RSq",
     type = "l")
point = as.numeric(which.max(fit.summary$adjr2))
points(point, fit.summary$adjr2[point],
       col = "red", cex = 2, pch = 8)
# rsq says our best model is all predictors, this is bc it lessens the error
# with each additional predictor.  Not a good indicator.
# Adjr2 says our very best is at 6, however the elbow can be seen around 2 or 3
# usually the elbow is our best bet with consideration to complexity

# Get coefficients of best model for a given size
coef(fit.full, 6)
```

```
##   (Intercept)     cylinders  displacement    horsepower        weight
## -15.563492306  -0.506685137   0.019269286  -0.023895029  -0.006218311
##          year        origin
##   0.747515952   1.428241885
```





3(d)

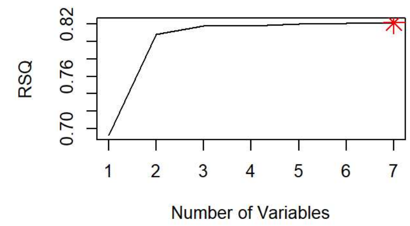
Now we use forward stepwise selection

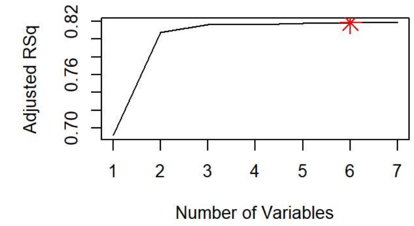
```
fit.fwd = regsubsets(mpg~., auto, nvmax = totpred, method = "forward")
fit.summary <- summary(fit.fwd)
fit.summary
```

```
## Subset selection object
## Call: regsubsets.formula(mpg ~ ., auto, nvmax = totpred, method = "forward")
## 7 Variables  (and intercept)
##              Forced in Forced out
## cylinders        FALSE      FALSE
## displacement     FALSE      FALSE
## horsepower       FALSE      FALSE
## weight           FALSE      FALSE
## acceleration     FALSE      FALSE
## year             FALSE      FALSE
## origin           FALSE      FALSE
## 1 subsets of each size up to 7
## Selection Algorithm: forward
##          cylinders displacement horsepower weight acceleration year origin
## 1  ( 1 ) " "       " "          " "        "*"    " "          " "  " "
## 2  ( 1 ) " "       " "          " "        "*"    " "          "*"  " "
## 3  ( 1 ) " "       " "          " "        "*"    " "          "*"  "*"
## 4  ( 1 ) " "       "*"          " "        "*"    " "          "*"  "*"
## 5  ( 1 ) " "       "*"          "*"        "*"    " "          "*"  "*"
## 6  ( 1 ) "*"       "*"          "*"        "*"    " "          "*"  "*"
## 7  ( 1 ) "*"       "*"          "*"        "*"    "*"          "*"  "*"
```

```
fwd.ar2 = mean(fit.summary$adjr2)

# Plot model fit measures for best model of each size against size
par(mfrow = c(2, 2))
# rsq
plot(fit.summary$rsq, xlab = "Number of Variables", ylab = "RSQ",
     type = "l")
point = as.numeric(which.max(fit.summary$rsq))
points(point, fit.summary$rsq[point],
       col = "red", cex = 2, pch = 8)
# Adjusted R^2
plot(fit.summary$adjr2, xlab = "Number of Variables", ylab = "Adjusted RSq",
     type = "l")
point = as.numeric(which.max(fit.summary$adjr2))
points(point, fit.summary$adjr2[point],
       col = "red", cex = 2, pch = 8)

# Get coefficients of best model for a given size
coef(fit.fwd, 6)
```

```
##   (Intercept)     cylinders  displacement    horsepower        weight
## -15.563492306  -0.506685137   0.019269286  -0.023895029  -0.006218311
##          year        origin
##   0.747515952   1.428241885
```





3(e)

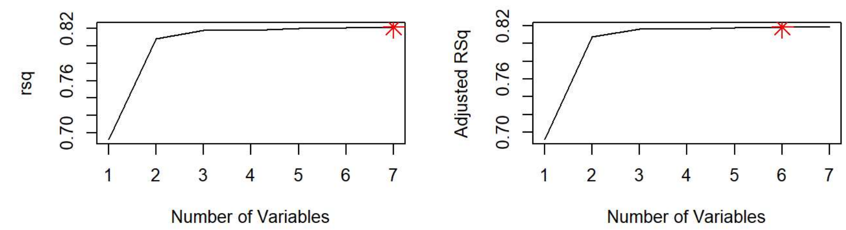
Last but not least is backward stepwise selection

```
fit.bwd = regsubsets(mpg~., auto, nvmax = totpred, method = "backward")
fit.summary <- summary(fit.bwd)
fit.summary
```

```
## Subset selection object
## Call: regsubsets.formula(mpg ~ ., auto, nvmax = totpred, method = "backward")
## 7 Variables  (and intercept)
##              Forced in Forced out
## cylinders        FALSE      FALSE
## displacement     FALSE      FALSE
## horsepower       FALSE      FALSE
## weight           FALSE      FALSE
## acceleration     FALSE      FALSE
## year             FALSE      FALSE
## origin           FALSE      FALSE
## 1 subsets of each size up to 7
## Selection Algorithm: backward
##          cylinders displacement horsepower weight acceleration year origin
## 1  ( 1 ) " "       " "          " "        "*"    " "          " "  " "
## 2  ( 1 ) " "       " "          " "        "*"    " "          "*"  " "
## 3  ( 1 ) " "       " "          " "        "*"    " "          "*"  "*"
## 4  ( 1 ) " "       "*"          " "        "*"    " "          "*"  "*"
## 5  ( 1 ) " "       "*"          "*"        "*"    " "          "*"  "*"
## 6  ( 1 ) "*"       "*"          "*"        "*"    " "          "*"  "*"
## 7  ( 1 ) "*"       "*"          "*"        "*"    "*"          "*"  "*"
```

```
bwd.ar2 = mean(fit.summary$adjr2)

# Plot model fit measures for best model of each size against size
par(mfrow = c(2, 2))
# rsq
plot(fit.summary$rsq, xlab = "Number of Variables", ylab = "rsq",
     type = "l")
point = as.numeric(which.max(fit.summary$rsq))
points(point, fit.summary$rsq[point],
       col = "red", cex = 2, pch = 8)
# Adjusted R^2
plot(fit.summary$adjr2, xlab = "Number of Variables", ylab = "Adjusted RSq",
     type = "l")
point = as.numeric(which.max(fit.summary$adjr2))
points(point, fit.summary$adjr2[point],
       col = "red", cex = 2, pch = 8)

# Get coefficients of best model for a given size
coef(fit.bwd, 6)
```

```
##   (Intercept)     cylinders  displacement    horsepower        weight
## -15.563492306  -0.506685137   0.019269286  -0.023895029  -0.006218311
##          year        origin
##   0.747515952   1.428241885
```



3(f)

Now we compare each model against a LOOCV on the data to see which model
is closest to the LOOCV adjusted R-square
Interestingly all subest methods resulted in the same model
and our quick and easy MLR model is the best of the 4

```
# we compare each model using LOOCV from caret

library(caret)
set.seed(1)
tc <- trainControl(method = "loocv", verboseIter = F)
model <- train(mpg~., auto,
               method="lm", trControl = tc)
model
```

```
## Linear Regression
##
## 392 samples
##   7 predictor
##
## No pre-processing
## Resampling: Leave-One-Out Cross-Validation
## Summary of sample sizes: 391, 391, 391, 391, 391, 391, ...
## Resampling results:
##
##   RMSE      Rsquared  MAE
##   2.556369  NaN       2.556369
##
## Tuning parameter 'intercept' was held constant at a value of TRUE
```

```
loocv.ar2 = summary(model)$adj.r.squared

rbind(loocv.ar2, lm.ar2, best.ar2, fwd.ar2, bwd.ar2)
```

```
##                [,1]
## loocv.ar2 0.8182238
## lm.ar2    0.8162176
## best.ar2  0.7979419
## fwd.ar2   0.7979419
## bwd.ar2   0.7979419
```