Homework 2

Matthew Reyers, Dani Chu and Ryan Sheehan

2017-09-25

Question 1 (Chapter 3, #3, 6 marks)

(a)

$$\begin{aligned}\text{Female} &= 50 + 20\text{GPA} + 0.07\text{IQ} + 35(\text{Gender} = \text{Female}) + 0.01(\text{GPA})\text{IQ} - 10(\text{GPA})(\text{Gender} = \text{Female}) \\ &= 85 + 20\text{GPA} + 0.07\text{IQ} + 0.01(\text{GPA})\text{IQ} - 10\text{GPA}\end{aligned}$$

$$\text{Male} = 50 + 20\text{GPA} + 0.07\text{IQ} + 0.01(\text{GPA})\text{IQ}$$

$$\text{(Female - Male } | \text{ Fixed IQ and GPA)} = 35 - 10\text{GPA}$$

Female grads make more money than male grads up until GPA hits 3.5. This means that the comment - iii) For a fixed value of IQ and GPA, males earn more on average than emales provided that the GPA is high enough - is correct at explaining the scenario.

(b)

$$\begin{aligned}\text{(Female Salary } | \text{ IQ} = 110, \text{ GPA} = 4\text{)} &= 85 + 20(4) + 0.07(110) + 0.01(4)(110) - 10(4) \\ &= 137.1\end{aligned}$$

(c) False. A small value as a coefficient could be indicative of a variable with a large range being used. Coefficients should not be taken at face value to describe whether or not there is an effect. A comparison must be made with the standard error of the estimate to truly understand what the evidence supports.

Question 2 (Chapter 3, #9, 10 marks)

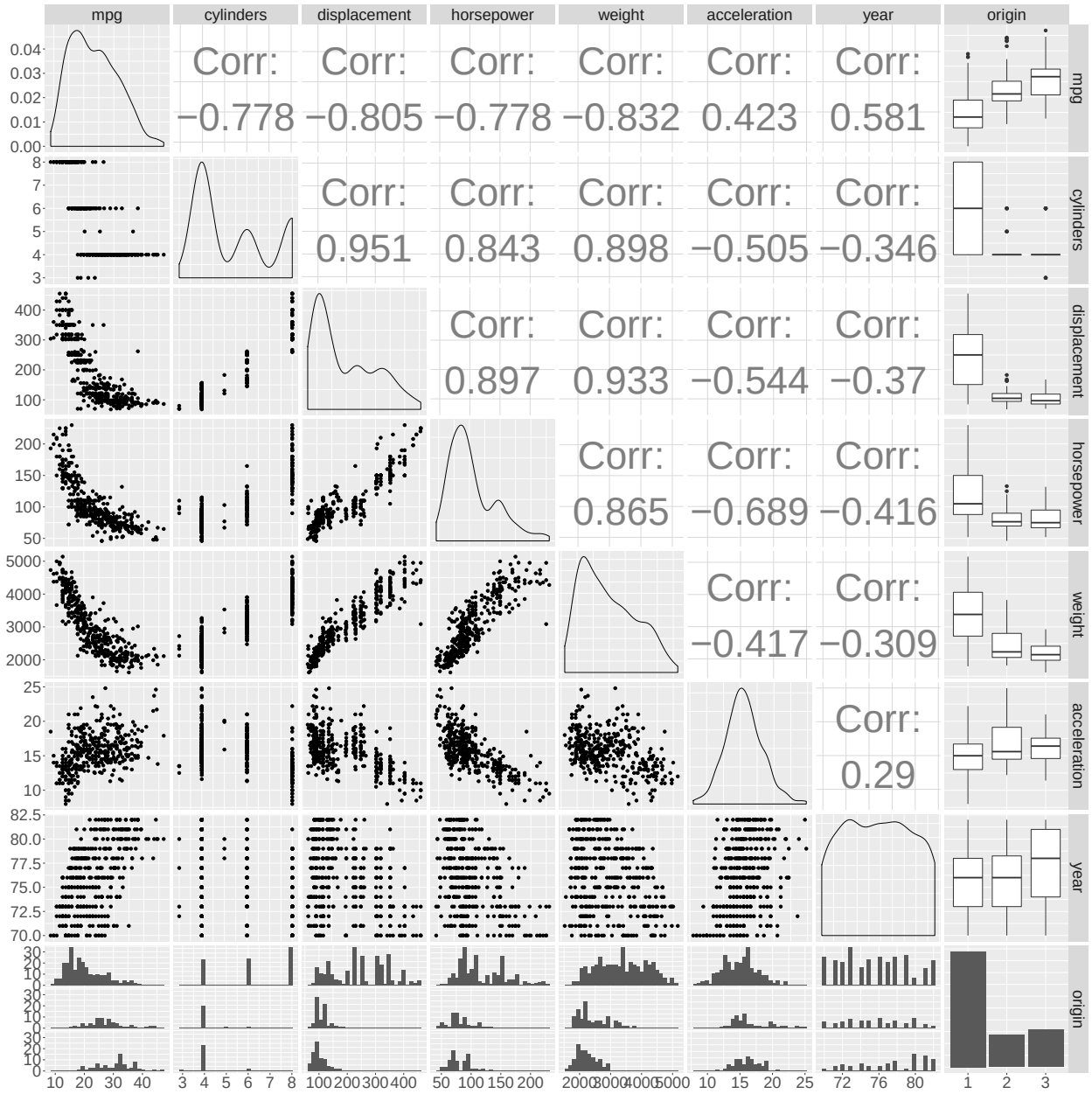
```
library(ISLR)
library(dplyr)
library(knitr)
library(GGally)
library(class)
library(car)
library(MASS)
library(ggplot2)
library(pander)
panderOptions("digits", 2)
data(Auto)

Auto <-
  Auto %>% dplyr::select(-name) %>% mutate(origin = factor(origin))
head(Auto) %>% kable()
```

mpg	cylinders	displacement	horsepower	weight	acceleration	year	origin
18	8	307	130	3504	12.0	70	1
15	8	350	165	3693	11.5	70	1
18	8	318	150	3436	11.0	70	1
16	8	304	150	3433	12.0	70	1
17	8	302	140	3449	10.5	70	1
15	8	429	198	4341	10.0	70	1

(a)

```
# Scatterplot matrix
ggpairs(Auto, upper= list(continuous = GGally::wrap("cor", size = 18))) +
  theme(text = element_text(size = 26))
```



(b)

```
# Name is already excluded by the piping done by first function
cor(Auto %>% dplyr::select(-origin)) %>%  round(.,2) %>%  kable()
```

	mpg	cylinders	displacement	horsepower	weight	acceleration	year
mpg	1.00	-0.78	-0.81	-0.78	-0.83	0.42	0.58
cylinders	-0.78	1.00	0.95	0.84	0.90	-0.50	-0.35
displacement	-0.81	0.95	1.00	0.90	0.93	-0.54	-0.37

	mpg	cylinders	displacement	horsepower	weight	acceleration	year
horsepower	-0.78	0.84	0.90	1.00	0.86	-0.69	-0.42
weight	-0.83	0.90	0.93	0.86	1.00	-0.42	-0.31
acceleration	0.42	-0.50	-0.54	-0.69	-0.42	1.00	0.29
year	0.58	-0.35	-0.37	-0.42	-0.31	0.29	1.00

(c)

```
myModel = lm(data = Auto, mpg ~ .)
summary(myModel) %>% pander(add.significance.stars = TRUE)
```

	Estimate	Std. Error	t value	Pr(>|t|)	
(Intercept)	-18	4.7	-3.8	0.00014	* * *
cylinders	-0.49	0.32	-1.5	0.13	
displacement	0.024	0.0077	3.1	0.0019	* *
horsepower	-0.018	0.014	-1.3	0.19	
weight	-0.0067	0.00066	-10	6.4e-22	* * *
acceleration	0.079	0.098	0.81	0.42	
year	0.78	0.052	15	2.3e-40	* * *
origin2	2.6	0.57	4.6	4.7e-06	* * *
origin3	2.9	0.55	5.2	3.9e-07	* * *

Table 4: Fitting linear model: mpg ~ .

Observations	Residual Std. Error	R^2	Adjusted R^2
392	3.3	0.82	0.82

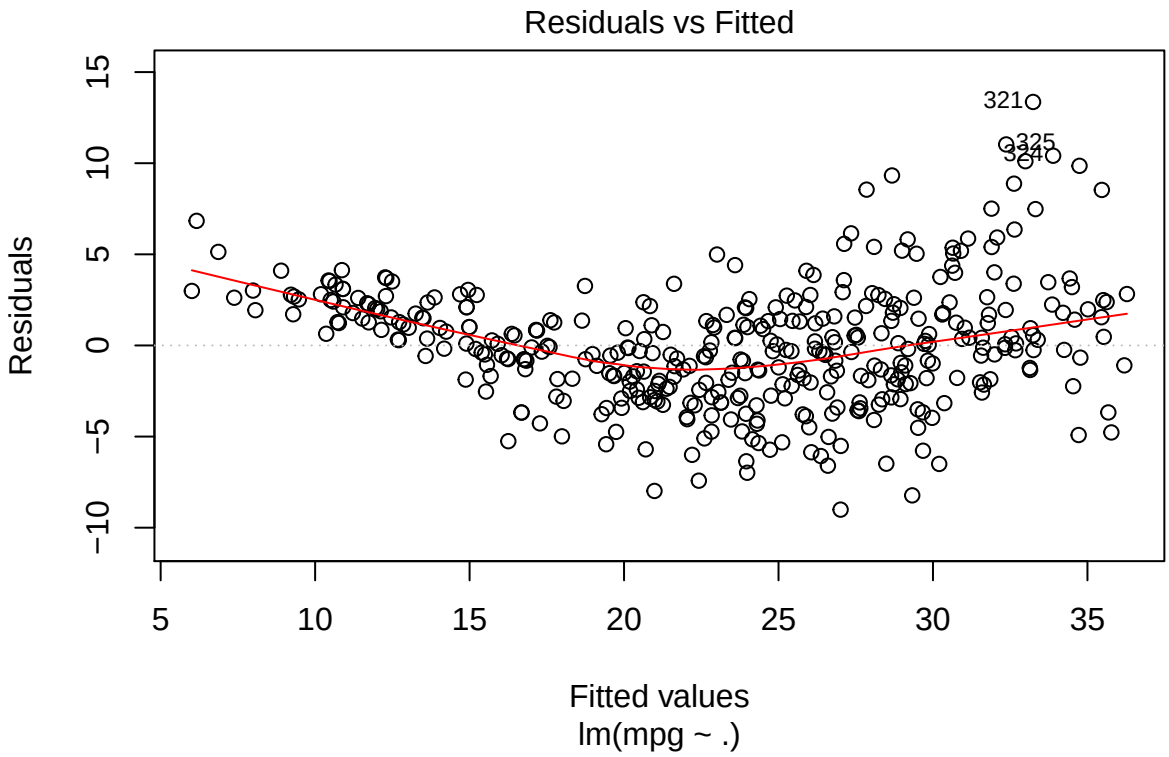
i) There is some relationship between the response and the predictors. This is noticeable in the fact that many of the predictors have a p value smaller than 0.05, meaning that evidence indicates that coefficient is not zero.

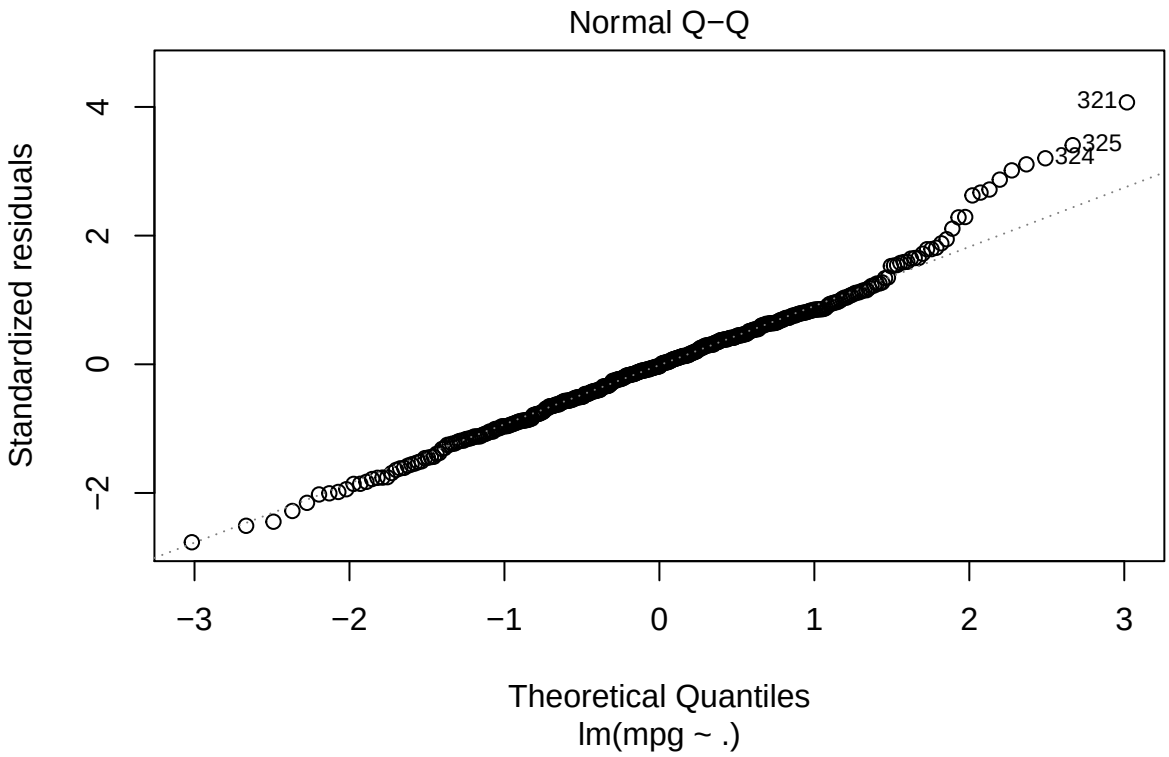
ii) The predictors with a significant relationship to the response are displacement, weight, year, and origin (though part of this effect is hidden in the intercept).

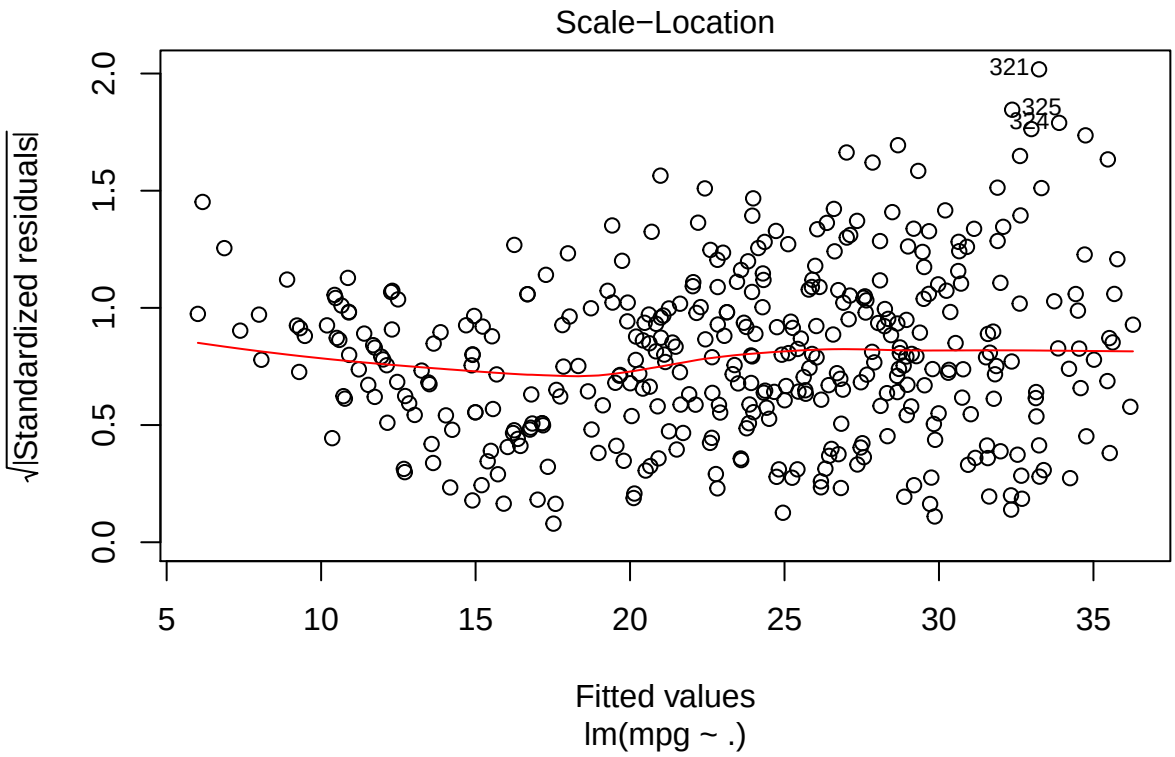
iii) The coefficient for year (0.777) indicates that when all else is held equal, a 1 unit increase in year leads to a 0.777 unit increase in mpg.

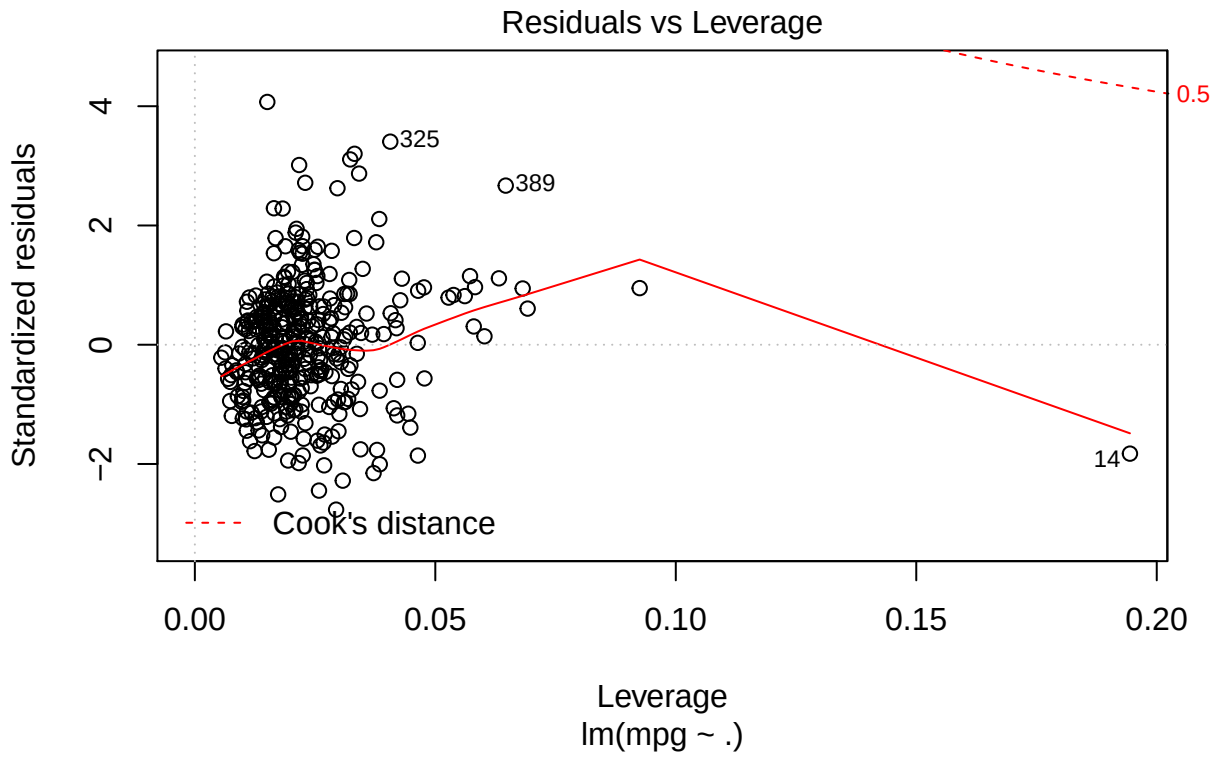
(d)

```
myModelResid = summary(myModel)$residuals
myModelFit   = fitted.values(myModel)
plot(myModel)
```

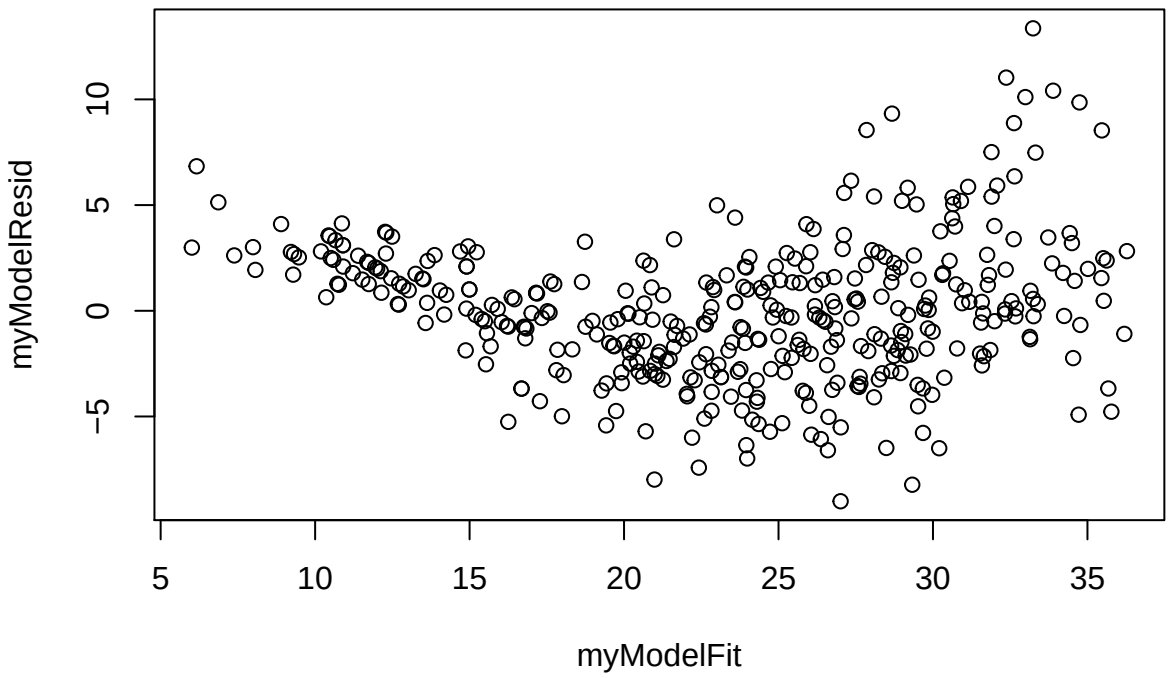

Residuals vs Fitted
Residuals
Fitted values
lm(mpg ~ .)
321
325
324
15
10
5
0
−5
−10
5
10
15
20
25
30
35


Normal Q−Q
Standardized residuals
Theoretical Quantiles
lm(mpg ~ .)
321
325
324
−3
−2
−1
0
1
2
3
−2
0
2
4


Scale−Location
√|Standardized residuals|
0.0
0.5
1.0
1.5
2.0
321
325
324
5
10
15
20
25
30
35
Fitted values
lm(mpg ~ .)



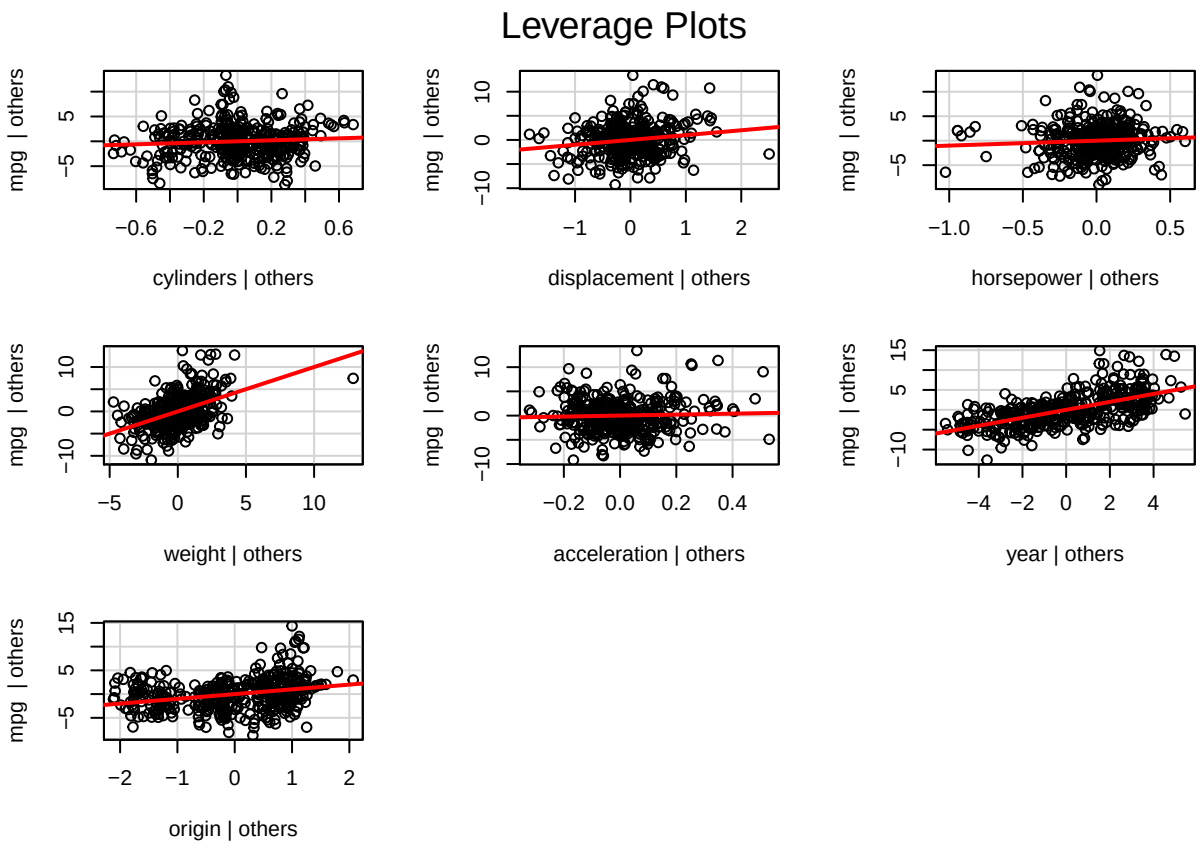

```
plot(myModelFit, myModelResid)
```



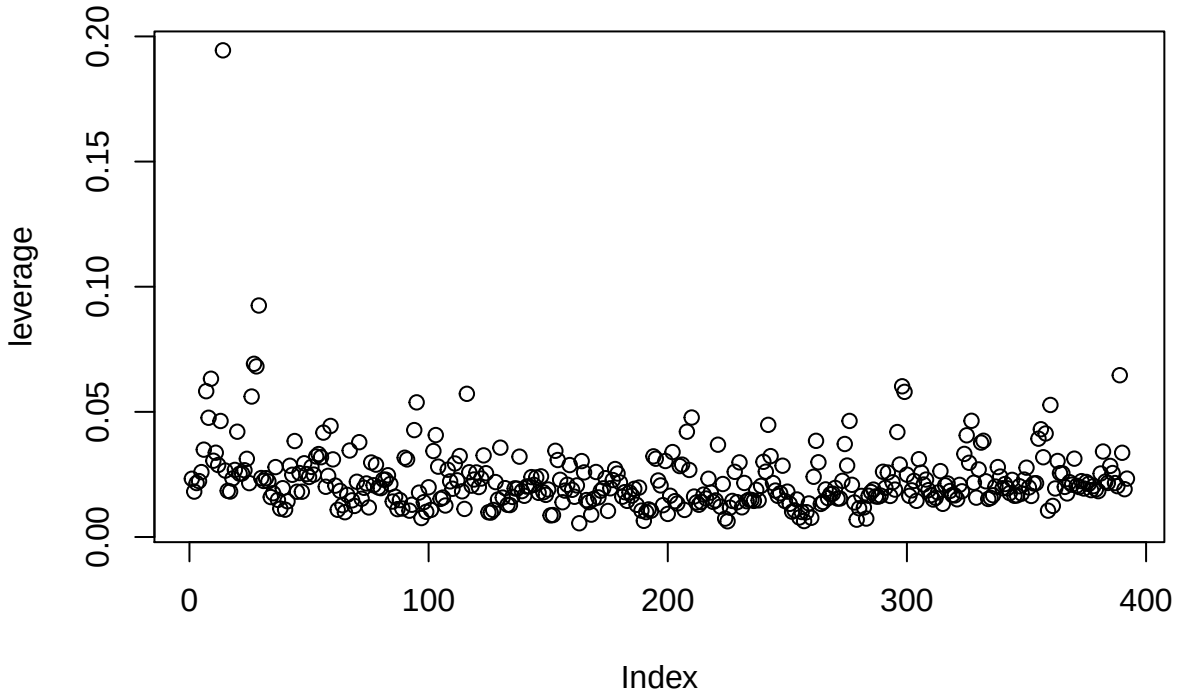

```
outlierTest(myModel)
```

```
##     rstudent unadjusted p-value Bonferonni p
## 321 4.157157          3.981e-05     0.015606
```

```
leveragePlots(myModel)
```




```
leverage = hat(model.matrix(myModel))
plot(leverage)
```




```
bigLevObservation = Auto[leverage > 0.1, ]
bigLevObservation %>% kable()
```

	mpg	cylinders	displacement	horsepower	weight	acceleration	year	origin
14	14	8	455	225	3086	10	70	1

Looking at the fit, it is noticable that the larger fitted values have larger residuals and that this causes a slight cone shape to the residual plot. This entails a non-equal standard deviation across the results. As for outliers a few values have residuals upwards of 10 and some near -10. These observations definitely generate some cause for concern and should be further investigated to determine the appropriate course of action. Using the basic outlierTest from the car package confirms that there observation 321 is a potential outlier. For leverage there is one observation that has a leverage of slightly more than 0.2. Compared to the rest of the leverage measures this is abnormally large. Values like this should be dealt with carefully.

(e)

```
intModel = lm(mpg ~ .*., data = Auto)
summary(intModel) %>% pander(add.significance.stars = TRUE)
```

	Estimate	Std. Error	t value	Pr(>|t|)
(Intercept)	44	51	0.86	0.39
cylinders	3.3	8.2	0.4	0.69
displacement	-0.35	0.2	-1.8	0.075
horsepower	0.53	0.34	1.6	0.12
weight	-0.0033	0.018	-0.18	0.86

	Estimate	Std. Error	t value	Pr(>|t|)	
acceleration	-6	2.1	-2.8	0.0051	* *
year	0.48	0.59	0.82	0.42	
origin2	-35	13	-2.8	0.0055	* *
origin3	-38	14	-2.6	0.0087	* *
cylinders:displacement	-0.0063	0.0071	-0.89	0.37	
cylinders:horsepower	0.015	0.025	0.59	0.56	
cylinders:weight	0.00057	9e-04	0.63	0.53	
cylinders:acceleration	0.37	0.17	2.2	0.029	*
cylinders:year	-0.14	0.097	-1.5	0.13	
cylinders:origin2	-0.72	1.1	-0.66	0.51	
cylinders:origin3	1.2	1	1.2	0.22	
displacement:horsepower	-5.4e-05	0.00029	-0.19	0.85	
displacement:weight	2.7e-05	1.5e-05	1.8	0.068	
displacement:acceleration	-0.0025	0.0034	-0.76	0.45	
displacement:year	0.0045	0.0024	1.9	0.064	
displacement:origin2	-0.034	0.042	-0.8	0.43	
displacement:origin3	0.054	0.041	1.3	0.2	
horsepower:weight	-3.4e-05	3e-05	-1.2	0.25	
horsepower:acceleration	-0.0034	0.0039	-0.88	0.38	
horsepower:year	-0.0064	0.0039	-1.7	0.099	
horsepower:origin2	-0.0049	0.051	-0.096	0.92	
horsepower:origin3	0.023	0.063	0.37	0.71	
weight:acceleration	-6.9e-05	0.00024	-0.29	0.77	
weight:year	-8.1e-05	0.00022	-0.37	0.71	
weight:origin2	0.0023	0.0027	0.85	0.4	
weight:origin3	-0.0045	0.0035	-1.3	0.2	
acceleration:year	0.061	0.025	2.4	0.016	*
acceleration:origin2	0.92	0.26	3.5	0.00053	* * *
acceleration:origin3	0.72	0.33	2.2	0.029	*
year:origin2	0.29	0.14	2	0.043	*
year:origin3	0.31	0.15	2.1	0.035	*

Table 7: Fitting linear model: mpg ~ . * .

Observations	Residual Std. Error	R^2	Adjusted R^2
392	2.6	0.9	0.89

The interactions between acceleration and year, acceleration and origin, as well as year and origin all seem to be useful effects to keep in mind. Cylinders and acceleration also shows promise for predicting as it also has a p value < 0.05.

(f) The work below shows some of these functions considered with respect to the variable weight. What can be found in the results is a show of effect in the log and square root effects. This could perhaps be due to a trend in the data being adjusted into a better fit for linear regression as the log transformation and the square root transformation behave similarly.

```
rootModel = lm(mpg ~ . + I(log(weight))+I((weight)^.5)+I((weight)^2), data = Auto)
summary(rootModel) %>% pander(add.significance.stars = TRUE)
```

	Estimate	Std. Error	t value	Pr(>|t|)	

	Estimate	Std. Error	t value	Pr(>|t|)	
(Intercept)	-5291	2599	-2	0.043	*
cylinders	-0.24	0.32	-0.76	0.44	
displacement	0.019	0.0069	2.7	0.0065	* *
horsepower	-0.029	0.012	-2.4	0.019	*
weight	0.74	0.39	1.9	0.058	
acceleration	0.073	0.089	0.81	0.42	
year	0.82	0.047	18	6e-51	* * *
origin2	1.9	0.52	3.6	0.00031	* * *
origin3	1.6	0.52	3	0.0028	* *
I(log(weight))	1173	580	2	0.044	*
I((weight)^0.5)	-113	57	-2	0.047	*
I((weight)^2)	-1.8e-05	1.1e-05	-1.7	0.096	

Table 9: Fitting linear model: mpg ~ . + I(log(weight)) + I((weight)^0.5) + I((weight)^2)

Observations	Residual Std. Error	R^2	Adjusted R^2
392	3	0.86	0.86

Question 3 (Chapter 4, #4, 7 marks)

(a) On average we should be using 10% of the observations due to the uniform nature on the distribution of X

(b) On average we should be using 1% of the total observations as both X1 and X2 are uniformly distributed and we are looking at 10% ranges for each.

(c) On average we should be using a proportion of (.10)^100 observations due to the uniform nature of all of the variables.

(d) Relative to the number of factors, the quantity of observations to be used quickly diminishes when using fixed ranges for neighbour determination. This is because the domain on which an observation is deemed to be an elligible neighbour is being further contracted with each additional constraint (feature). A solution, or at least improvement, to this is to take the K nearest observations rather than all the observations on a specified domain.

(e) For p = 1, a side has length 0.1 For p = 2, a side has length (0.1)^1/2 For p = 100, a side has length (0.1)^1/100 This answer may seem unintuitive based on previous results. It is sensical though due to the fact that we want to use 10% of the data to train and this data exists in the area or volume enclosed by the hypercube. This would be a measure calculated as a product of all dimension's measurements and therefore would require the aforementioned side lengths.

Question 4 (Chapter 4, #10, 9 marks)

```
data(Weekly)
head(Weekly) %>% kable()
```

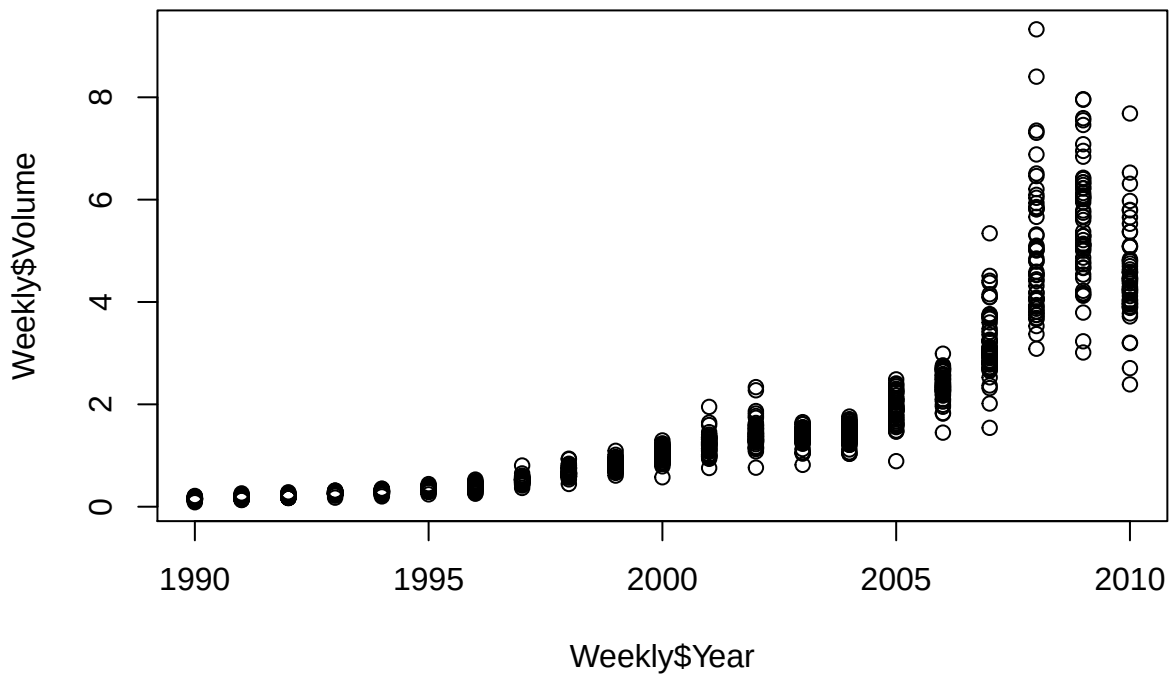
Year	Lag1	Lag2	Lag3	Lag4	Lag5	Volume	Today	Direction
1990	0.816	1.572	-3.936	-0.229	-3.484	0.1549760	-0.270	Down
1990	-0.270	0.816	1.572	-3.936	-0.229	0.1485740	-2.576	Down
1990	-2.576	-0.270	0.816	1.572	-3.936	0.1598375	3.514	Up
1990	3.514	-2.576	-0.270	0.816	1.572	0.1616300	0.712	Up
1990	0.712	3.514	-2.576	-0.270	0.816	0.1537280	1.178	Up
1990	1.178	0.712	3.514	-2.576	-0.270	0.1544440	-1.372	Down

```
names(Weekly) %>% pander()
```

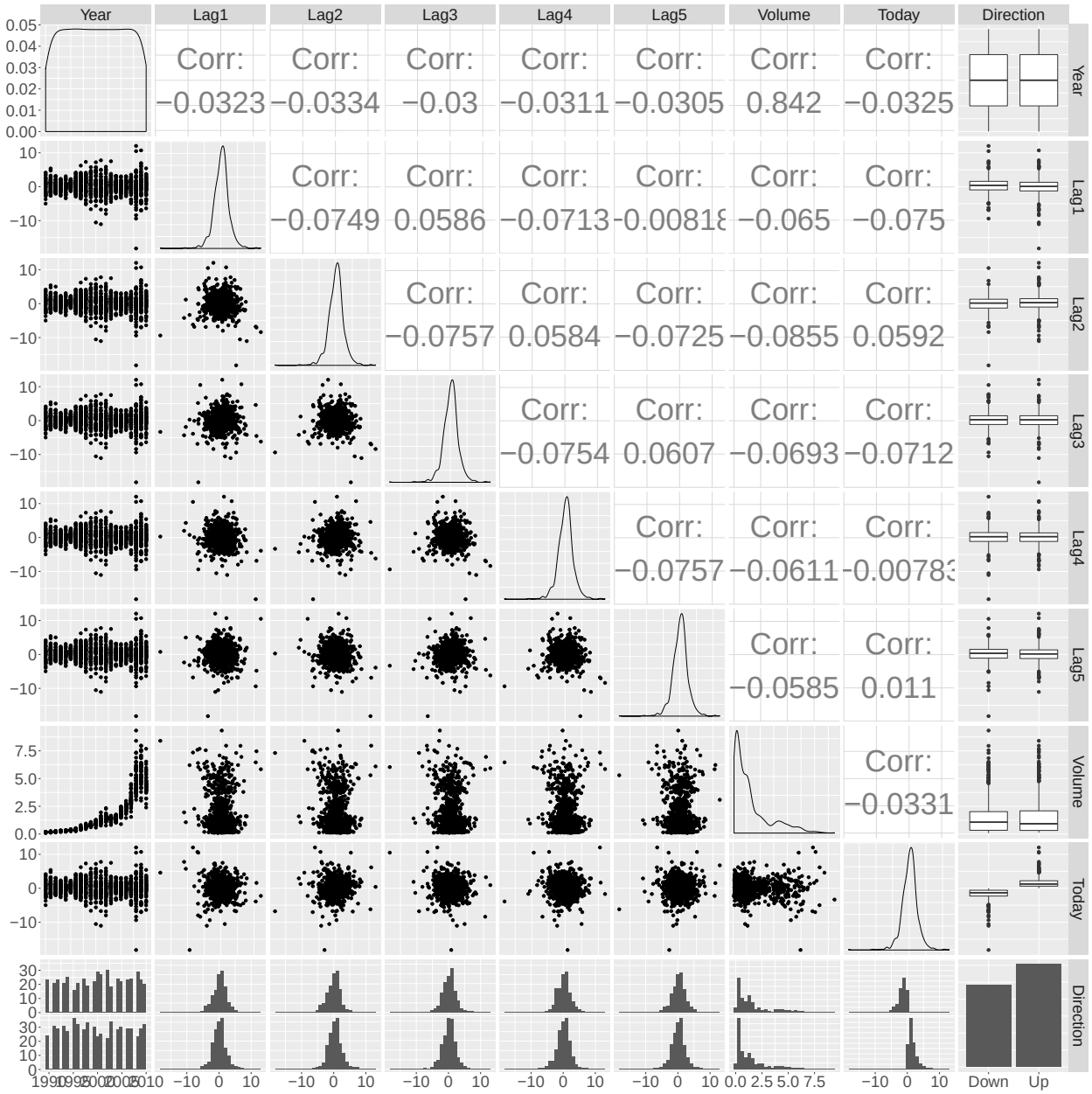
Year, *Lag1*, *Lag2*, *Lag3*, *Lag4*, *Lag5*, *Volume*, *Today* and *Direction*

(a) The first noticeable pattern is the increase in volume as the years go by. The only time this trend subsides is shortly after the market crash of 2008. We should also note that there should be some collinearity in the lags as stocks go on runs or losses and this information would likely be captured by these variables.

```
plot(Weekly$Year, Weekly$Volume)
```




```
ggpairs(Weekly, upper= list(continuous = GGally::wrap("cor", size = 14))) +
  theme(text = element_text(size = 26))
```



(b) The model suggests that only Lag2 is statistically significant in this prediction.

```
str(Weekly)
```

```
## 'data.frame':    1089 obs. of  9 variables:
##  $ Year     : num  1990 1990 1990 1990 1990 1990 1990 1990 1990 1990 ...
##  $ Lag1     : num  0.816 -0.27 -2.576 3.514 0.712 ...
##  $ Lag2     : num  1.572 0.816 -0.27 -2.576 3.514 ...
##  $ Lag3     : num  -3.936 1.572 0.816 -0.27 -2.576 ...
##  $ Lag4     : num  -0.229 -3.936 1.572 0.816 -0.27 ...
##  $ Lag5     : num  -3.484 -0.229 -3.936 1.572 0.816 ...
##  $ Volume   : num  0.155 0.149 0.16 0.162 0.154 ...
##  $ Today    : num  -0.27 -2.576 3.514 0.712 1.178 ...
##  $ Direction: Factor w/ 2 levels "Down","Up": 1 1 2 2 2 1 2 2 2 1 ...
```

```
names(Weekly) %>% pander()
```

Year, *Lag1*, *Lag2*, *Lag3*, *Lag4*, *Lag5*, *Volume*, *Today* and *Direction*

```
logReg = glm(data = Weekly, Direction ~ . - Year - Today, family = binomial(link = "logit"))
summary(logReg) %>% pander()
```

	Estimate	Std. Error	z value	Pr(>|z|)
(Intercept)	0.27	0.086	3.1	0.0019
Lag1	-0.041	0.026	-1.6	0.12
Lag2	0.058	0.027	2.2	0.03
Lag3	-0.016	0.027	-0.6	0.55
Lag4	-0.028	0.026	-1.1	0.29
Lag5	-0.014	0.026	-0.55	0.58
Volume	-0.023	0.037	-0.62	0.54

(Dispersion parameter for binomial family taken to be 1)

Null deviance:	1496 on 1088 degrees of freedom
Residual deviance:	1486 on 1082 degrees of freedom

```
names(logReg) %>% pander()
```

coefficients, *residuals*, *fitted.values*, *effects*, *R*, *rank*, *qr*, *family*, *linear.predictors*, *deviance*, *aic*, *null.deviance*, *iter*, *weights*, *prior.weights*, *df.residual*, *df.null*, *y*, *converged*, *boundary*, *model*, *call*, *formula*, *terms*, *data*, *offset*, *control*, *method*, *contrasts* and *xlevels*

```
Guess = ifelse(predict(logReg, Weekly, type = "response") > 0.5, "Up", "Down")
str(Weekly$Direction)
```

```
##  Factor w/ 2 levels "Down","Up": 1 1 2 2 2 1 2 2 2 1 ...
```

```
confusion = xtabs(~as.factor(Guess) + Weekly$Direction)
confusion %>% kable()
```

	Down	Up
Down	54	48
Up	430	557

```
correctPerc = sum(diag(confusion))/sum(confusion) * 100
correctPerc %>% kable()
```

56.10652

(c) By the above work, the model correctly predicts 56% of the data that it was originally fit against. The confusion matrix shows that observations that are correctly classified vs. those incorrectly classified by class are roughly equal. In example, 54 Down observations are correctly identified as such though 48 Down observations are classified incorrectly as Up.

(d)

```
train = Weekly[Weekly$Year <= 2008, c("Direction", "Lag2")]
test = Weekly[Weekly$Year > 2008, c("Direction", "Lag2")]
testClasses = test$Direction
trainMod = glm(data = train, Direction ~ Lag2, family = binomial(link = "logit"))
guessFit = ifelse(predict(trainMod, test, type = "response") > 0.5, "Up", "Down")
confusionTest = xtabs(~as.factor(guessFit) + testClasses)
confusionTest %>% pander()
```

	Down	Up
Down	9	5
Up	34	56

```
accuracyGLM = sum(diag(confusionTest)) / sum(confusionTest) * 100
```

(e)

```
lda.fit = lda(Direction ~ Lag2, data = train)
lda.fit
```

```
## Call:
## lda(Direction ~ Lag2, data = train)
##
## Prior probabilities of groups:
##      Down        Up
## 0.4477157 0.5522843
##
## Group means:
##              Lag2
## Down -0.03568254
## Up    0.26036581
##
## Coefficients of linear discriminants:
##            LD1
## Lag2 0.4414162
```

```
lda.pred = predict(lda.fit, test)
confusionLDA = xtabs(~testClasses + lda.pred$class)
confusionLDA %>% pander()
```

	Down	Up
Down	9	34
Up	5	56

```
accuracyLDA = sum(diag(confusionLDA)) / sum(confusionLDA) * 100
```

(f)

```
qda.fit = qda(Direction ~ Lag2, data = train)
qda.fit
```

```
## Call:
## qda(Direction ~ Lag2, data = train)
##
```

```
## Prior probabilities of groups:
##      Down        Up
## 0.4477157 0.5522843
##
## Group means:
##             Lag2
## Down -0.03568254
## Up    0.26036581
```

```
qda.class = predict(qda.fit, test)$class
confusionQDA = xtabs(~ testClasses + qda.class)
confusionQDA %>% pander()
```

	Down	Up
Down	0	43
Up	0	61

```
accuracyQDA = sum(diag(confusionQDA)) / sum(confusionQDA) * 100
```

(g)

```
set.seed(1234)
# Literally just needed them as matrices/vectors, not data frames
knn.pred = knn(cbind(train$Lag2), cbind(test$Lag2), train$Direction, k = 1)
confusionKNN = xtabs(~ testClasses + knn.pred)
confusionKNN %>% pander()
```

	Down	Up
Down	21	22
Up	29	32

```
accuracyKNN = sum(diag(confusionKNN)) / sum(confusionKNN) * 100
```

(h) According to the following results, the GLM and LDA approaches are most accurate for this data when only considering Lag2 as a predictor.

```
# Confusion measures
accuracyGLM %>% kable()
```

62.5

```
accuracyLDA %>% kable()
```

62.5

```
accuracyQDA %>% kable()
```

58.65385

```
accuracyKNN %>% kable()
```

50.96154

(i) DO NOT HAND IN