Problem 1

This question should be answered using the Default data set in ISLR package. In Chapter 4 on classification, we used logistic regression to predict the probability of default using income and balance. Now we will estimate the test error of this logistic regression model using the validation set approach. Do not forget to set a random seed before beginning your analysis.

(a) Fit a logistic regression model that predicts default using income and balance.

By using the following code to fit the logistic regression:

```
library(ISLR)
data(Default)
names(Default)
summary(Default)
attach(Default)
LG.fit = glm(default~income+balance, data = Default, family=binomial)
summary(LG.fit)
```

(b) Using the validation set approach, estimate the test error of this model. You need to perform the following steps:

i. Split the sample set into a training set and a validation set.

ii. Fit a logistic regression model using only the training data set.

iii. Obtain a prediction of default status for each individual in the validation set using a threshold of 0.5.

iv. Compute the validation set error, which is the fraction of the observations in the validation set that are misclassified.

Using following code to finish the steps:

```
train_set = sample(dim(Default)[1],dim(Default)[1]/1.5)
#i. Split the sample set into a training set and a validation set.
NLG.fit = glm(default ~ income + balance, data = Default, family = "binomial", subset = train_set)
#ii. Fit a logistic regression model using only the training data set.
probs = predict(NLG.fit, newdata = Default[-train_set, ], type = "response")
pred.glm = rep("No", length(probs))
pred.glm[probs > 0.5] = "Yes"
#iii. Obtain a prediction of default status for each individual in the validation set using a threshold of 0.5.
mean(pred.glm != Default[-train_set, ]$default)
```

And we can get the validation set error is 0.02609

```
> mean(pred.glm != Default[-train_set, ]$default)
[1] 0.02609478
```

(c) Repeat the process in (b) three times, using three different splits of the observations into a training set and a validation set. Comment on the results obtained.

```
> mean(pred.glm != Default[-train_set, ]$default)
[1] 0.02729454
> View(Default)
> train_set = sample(dim(Default)[1],dim(Default)[1]/1.5)
> NLG.fit = glm(default ~ income + balance, data = Default, family = "binomial", subset = train_set)
> probs = predict(NLG.fit, newdata = Default[-train_set, ], type = "response")
> pred.glm = rep("No", length(probs))
> pred.glm[probs > 0.5] = "Yes"
> mean(pred.glm != Default[-train_set, ]$default)
[1] 0.02279544
> train_set = sample(dim(Default)[1],dim(Default)[1]/1.5)
> NLG.fit = glm(default ~ income + balance, data = Default, family = "binomial", subset = train_set)
> probs = predict(NLG.fit, newdata = Default[-train_set, ], type = "response")
> pred.glm = rep("No", length(probs))
> pred.glm[probs > 0.5] = "Yes"
> mean(pred.glm != Default[-train_set, ]$default)
[1] 0.02459508
```

Therefore, we can know that the validation estimate of the test error rate can vary based on the specific observations included in the training and validation sets.

(d) Consider another logistic regression model that predicts default using income, balance and student (qualitative). Estimate the test error for this model using the validation set approach. Does including the qualitative variable student lead to a reduction of test error rate?
By using the following coding:

```
#(d)adding one student variable, check the test error
train_set <- sample(dim(Default)[1], dim(Default)[1] / 1.5)
LG.glm <- glm(default ~ income + balance + student, data = Default, family = "binomial", subset = train_set)
pred.glm <- rep("No", length(probs))
probs <- predict(LG.glm, newdata = Default[-train_set, ], type = "response")
pred.glm[probs > 0.5] <- "Yes"
mean(pred.glm != Default[-train_set, ]$default)
```

```
> mean(pred.glm != Default[-train_set, ]$default)
[1] 0.0269946
>
```

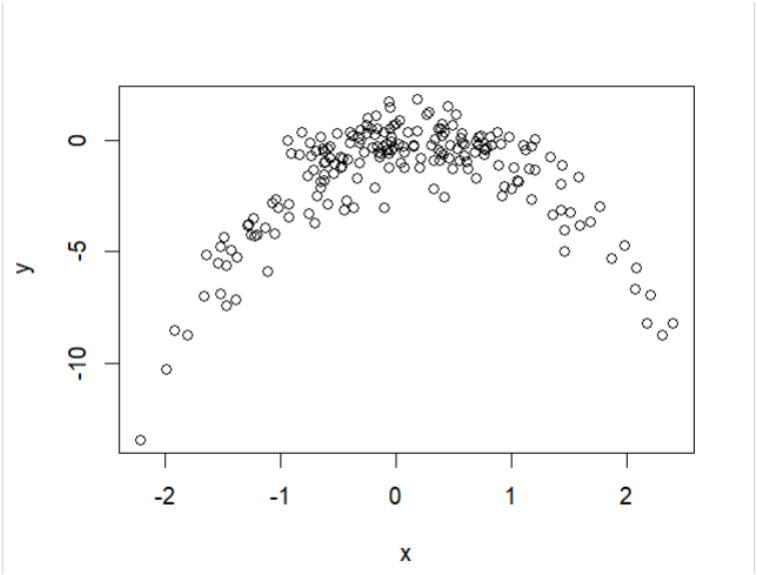
The result gives us 0.0269946, so it's not really reduction the test error rate.

Problem 2
This question requires performing cross validation on a simulated data set.
(a) Generate a simulated data set as follows:
set.seed(1)
x=rnorm(200)
y=x-2*x^2+rnorm(200)
In this data set, what is n and what is p? Write out the model used to generate the data in equation form (i.e., the true model of the data).

From the coding we can see that there are 200 observations, so n = 200, and p=2 since there are two independent variables in the model(the model used to generate the data includes two terms of the independent variable x (linear and quadratic).).

(b) Create a scatter plot of Y vs X. Comment on what you find.

```
> set.seed(1)
> x=rnorm(200)
> y=x-2*x^2+rnorm(200)
> plot(x,y)
>
```



From the above graph, we can see it's a clearly carve down relationship.

(c) Consider the following four models for the data set, Compute the LOOCV errors that result from fitting these models:

i. $Y=\beta 0+\beta 1X+\epsilon$

```
> library(boot)
> Data = data.frame(x,y)
> set.seed(1)
> LG.fit = glm(y~x)
> cv.glm(Data, LG.fit)$delta
[1] 6.037638 6.036993
```

ii. $Y=\beta 0+\beta 1X+\beta 2X2+\epsilon$

```
> LG.fit = glm(y~poly(x,2))
> cv.glm(Data, LG.fit)$delta
[1] 1.040922 1.040840
```

iii. $Y=\beta 0+\beta 1X+\beta 2X2+\beta 3X3+\epsilon$

```
> LG.fit = glm(y~poly(x,3))
> cv.glm(Data, LG.fit)$delta
[1] 1.039049 1.038940
```

iv. $Y=\beta 0+\beta 1X+\beta 2X2+\beta 3X3+\beta 4X4+\epsilon$

```
> LG.fit = glm(y~poly(x,4))
> cv.glm(Data, LG.fit)$delta
[1] 1.028604 1.028482
```

Compute the LOOCV errors that result from fitting these models.

(d) Repeat (c) using another random seed, and report your results. Are your results the same as what you got in (c)? Why?

```
> set.seed(3)
> LG.fit = glm(y~x)
> cv.glm(Data, LG.fit)$delta
[1] 6.037638 6.036993
>
> LG.fit = glm(y~poly(x,2))
> cv.glm(Data, LG.fit)$delta
[1] 1.040922 1.040840
>
> LG.fit = glm(y~poly(x,3))
> cv.glm(Data, LG.fit)$delta
[1] 1.039049 1.038940
>
> LG.fit = glm(y~poly(x,4))
> cv.glm(Data, LG.fit)$delta
[1] 1.028604 1.028482
>
```

So, we can see, by changing the seed, we still get the same result. Although the seed used in LOOCV may vary, the results remain the same because all n folds containing a single observation will be evaluated.

(e) Which of the models in (c) had the smallest LOOCV error? Is this what you expected? Explain your answer.

The models had the smallest LOOCV is the polynomial of degree 4 (i.e., x^4) had the lowest LOOCV test error rate. This was expected because it matches the true form of Y.

(f) Now we use 5-fold CV for the model selection. Compute the CV errors that result from fitting the four models. Which model has the smallest CV error? Are the results consistent with LOOCV?

```
> # Print the cross-validation errors
> cat("CV error for model 1:", cv.error1, "\n")
CV error for model 1: 5.783126
> cat("CV error for model 2:", cv.error2, "\n")
CV error for model 2: 1.008758
> cat("CV error for model 3:", cv.error3, "\n")
CV error for model 3: 0.9971667
> cat("CV error for model 4:", cv.error4, "\n")
CV error for model 4: 0.9818643
```

So, by using k-fold k = 5, we can see the smallest CV error is still polynomial of degree 4. It is consistent with LOOCV.

(g) Repeat (f) using 10-fold CV. Are the results the same as 5-fold CV?

```
> # Print the cross-validation errors
> cat("CV error for model 1:", cv.error1_10, "\n")
CV error for model 1: 6.97686
> cat("CV error for model 2:", cv.error2_10, "\n")
CV error for model 2: 1.054148
> cat("CV error for model 3:", cv.error3_10, "\n")
CV error for model 3: 1.053641
> cat("CV error for model 4:", cv.error4_10, "\n")
CV error for model 4: 1.036868
```

From above result, we can see it is consistent with k = 5.