Solutions of the exercises from Chapter 5

Conceptual

Q1. Using basic statistical properties of the variance, as well as single variable calculus, derive (5.6). In other words, prove that α given by (5.6) does indeed minimize $\mathrm{Var}(\alpha X + (1-\alpha)Y)$.

We have

$$\mathrm{Var}(\alpha X + (1-\alpha)Y) = \alpha^2\sigma_X^2 + (1-\alpha)^2\sigma_Y^2 + 2\alpha(1-\alpha)\sigma_{XY}.$$

We now take the fist derivative of $\mathrm{Var}(\alpha X + (1-\alpha)Y)$ *relative to* α *and we get*

$$\frac{\partial}{\partial\alpha}\mathrm{Var}(\alpha X + (1-\alpha)Y) = 2\alpha\sigma_X^2 - 2\sigma_Y^2 + 2\alpha\sigma_Y^2 + 2\sigma_{XY} - 4\alpha\sigma_{XY}.$$

We now seek critical points by equalling the last expression to 0*,*

$$2\alpha\sigma_X^2 - 2\sigma_Y^2 + 2\alpha\sigma_Y^2 + 2\sigma_{XY} - 4\alpha\sigma_{XY} = 0,$$

which implies that

$$\alpha = \frac{\sigma_Y^2 - \sigma_{XY}}{\sigma_X^2 + \sigma_Y^2 - 2\sigma_{XY}}.$$

It remains to check that this point is in fact a minimum, this is equivalent to prove that the second derivative is positive,

$$\frac{\partial^2}{\partial\alpha^2}\mathrm{Var}(\alpha X + (1-\alpha)Y) = 2\sigma_X^2 + 2\sigma_Y^2 - 4\sigma_{XY} = 2\mathrm{Var}(X-Y) \geq 0.$$

Q2. We will now derive the probability that a given observation is part of a bootstrap sample. Suppose that we obtain a bootstrap sample from a set of n observations.

(a) What is the probability that the first bootstrap observation is not the jth observation from the original sample ? Justify your answer.

$1 - 1/n$.

(b) What is the probability that the second bootstrap observation is not the jth observation from the original sample ?

$1 - 1/n$.

(c) Argue that the probability that the jth observation is not in the bootstrap sample is $(1-1/n)^n$.

As bootstrapping sample with replacement, we have that the probability that the jth observation is not in the bootstrap sample is the product of the probabilities that each bootstrap observation is not the jth observation from the original sample

$$(1-1/n)\cdots(1-1/n) = (1-1/n)^n$$

as these probabilities are independant.

(d) When $n = 5$, what is the probability that the jth observation is in the bootstrap sample ?

We have

$$P(jth\ obs\ in\ bootstrap\ sample) = 1 - (1 - 1/5)^5 = 0.672.$$

(e) When $n = 100$, what is the probability that the jth observation is in the bootstrap sample ?

We have

$$P(jth\ obs\ in\ bootstrap\ sample) = 1 - (1 - 1/100)^{100} = 0.634.$$

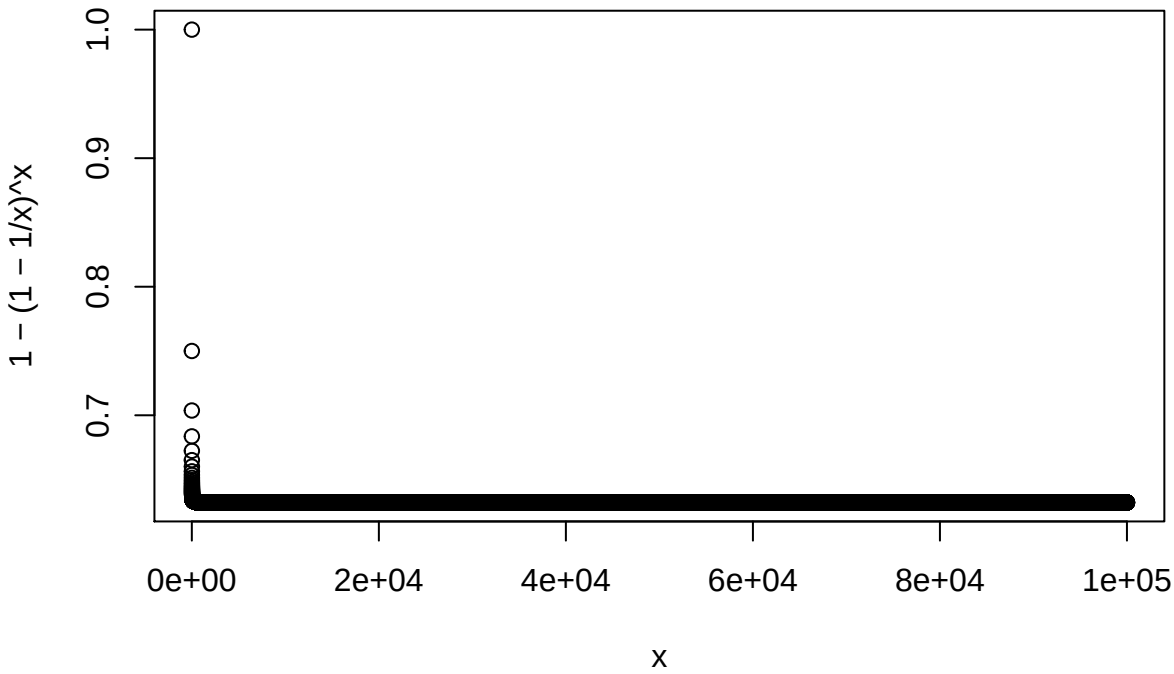
(f) When $n = 10000$, what is the probability that the jth observation is in the bootstrap sample ?

We have

$$P(jth\ obs\ in\ bootstrap\ sample) = 1 - (1 - 1/10000)^{10000} = 0.632.$$

(g) Create a plot that displays, for each integer value of n from 1 to 100000, the probability that the jth observation is in the bootstrap sample. Comment on what you observe.

```
x <- 1:100000
plot(x, 1 - (1 - 1/x)^x)
```



We may see that the plot quickly reaches an asymptote at about 0.632.

(h) We will now investigate numerically the probability that a bootstrap sample of size $n = 100$ contains the jth observation. Here $j = 4$. We repeatedly create bootstrap samples, and each time we record whether or not the fourth observation is contained in the bootstrap sample.

```
store <- rep(NA, 10000)
for (i in 1:10000) {
    store[i] <- sum(sample(1:100, rep = TRUE) == 4) > 0
}
mean(store)
```

```
## [1] 0.635
```

Comment on the results obtained.

A known fact from calculus tells us that

$$\lim_{n\to\infty} (1 + x/n)^n = e^x.$$

If we apply this fact to our case, we get that the probability that a bootstrap sample of size n contains the jth observation converges to $1 - 1/e = 0.632$ *as* $n \to \infty$.

Q3. We now review k-fold cross-validation.

(a) Explain how k-fold cross-validation is implemented.

The k-fold cross validation is implemented by taking the n observations and randomly splitting it into k non-overlapping groups of length of (approximately) n/k. These groups acts as a validation set, and the remainder (of length $n - n/k$*) acts as a training set. The test error is then estimated by averaging the k resulting MSE estimates.*

(b) What are the advantages and disadvantages of k-fold cross-validation relative to:

i. The validation set approach ?

The validation set approach has two main drawbacks compared to k-fold cross-validation. First, the validation estimate of the test error rate can be highly variable (depending on precisely which observations are included in the training set and which observations are included in the validation set). Second, only a subset of the observations are used to fit the model. Since statistical methods tend to perform worse when trained on fewer observations, this suggests that the validation set error rate may tend to overestimate the test error rate for the model fit on the entire data set.

ii. LOOCV ?

The LOOCV cross-validation approach is a special case of k-fold cross-validation in which $k = n$*. This approach has two drawbacks compared to k-fold cross-validation. First, it requires fitting the potentially computationally expensive model n times compared to k-fold cross-validation which requires the model to be fitted only k times. Second, the LOOCV cross-validation approach may give approximately unbiased estimates of the test error, since each training set contains* $n - 1$ *observations; however, this approach has higher variance than k-fold cross-validation (since we are averaging the outputs of n fitted models trained on an almost identical set of observations, these outputs are highly correlated, and the mean of highly correlated quantities has higher variance than less correlated ones). So, there is a bias-variance trade-off associated with the choice of k in k-fold cross-validation; typically using* $k = 5$ *or* $k = 10$ *yield test error rate estimates that suffer neither from excessively high bias nor from very high variance.*

Q4. Suppose that we use some statistical learning method to make a prediction for the response Y for a particular value of the predictor X. Carefully describe how we might estimate the standard deviation of our prediction.

We may estimate the standard deviation of our prediction by using the bootstrap method. In this case, rather than obtaining new independant data sets from the population and fitting our model on those data sets, we instead obtain repeated random samples from the original data set. In this case, we perform sampling with replacement B times and then find the corresponding estimates and the standard deviation of those B estimates by using equation (5.8).

Applied

Q5. In Chapter 4, we used logisitc regression to predict the probability of "default" using "income" and "balance" on the "Default" data set. We will now 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 uses "income" and "balance" to predict "default".

```
library(ISLR)
```

```
##
## Attaching package: 'ISLR'
##
## The following object is masked _by_ '.GlobalEnv':
##
##     Auto
```

```
attach(Default)
set.seed(1)
fit.glm <- glm(default ~ income + balance, data = Default, family = "binomial")
summary(fit.glm)
```

```
##
## Call:
## glm(formula = default ~ income + balance, family = "binomial",
##     data = Default)
##
## Deviance Residuals:
##    Min      1Q  Median      3Q     Max
## -2.473  -0.144  -0.057  -0.021   3.724
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept) -1.15e+01   4.35e-01  -26.54   <2e-16 ***
## income       2.08e-05   4.99e-06    4.17    3e-05 ***
## balance      5.65e-03   2.27e-04   24.84   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 2920.6  on 9999  degrees of freedom
```

```
## Residual deviance: 1579.0  on 9997  degrees of freedom
## AIC: 1585
##
## Number of Fisher Scoring iterations: 8
```

(b) Using the validation set approach, estimate the test error of this model. In order to do this, you must perform the following steps:

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

```
train <- sample(dim(Default)[1], dim(Default)[1] / 2)
```

ii. Fit a multiple logistic regression model using only the training observations.

```
fit.glm <- glm(default ~ income + balance, data = Default, family = "binomial", subset = train)
summary(fit.glm)
```

```
##
## Call:
## glm(formula = default ~ income + balance, family = "binomial",
##     data = Default, subset = train)
##
## Deviance Residuals:
##    Min      1Q  Median      3Q     Max
## -2.358  -0.127  -0.047  -0.016   3.812
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept) -1.21e+01   6.66e-01  -18.15   <2e-16 ***
## income       1.86e-05   7.57e-06    2.45    0.014 *
## balance      6.05e-03   3.47e-04   17.46   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1457.0  on 4999  degrees of freedom
## Residual deviance:  734.4  on 4997  degrees of freedom
## AIC: 740.4
##
## Number of Fisher Scoring iterations: 8
```

iii. Obtain a prediction of default status for each individual in the validation set by computing the posterior probability of default for that individual, and classifying the individual to the "default" category if the posterior probability is greater than 0.5.

```
probs <- predict(fit.glm, newdata = Default[-train, ], type = "response")
pred.glm <- rep("No", length(probs))
pred.glm[probs > 0.5] <- "Yes"
```

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

```
mean(pred.glm != Default[-train, ]$default)
```

```
## [1] 0.0286
```

We have a 2.86% *test error rate with the validation set approach.*

(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.

```
train <- sample(dim(Default)[1], dim(Default)[1] / 2)
fit.glm <- glm(default ~ income + balance, data = Default, family = "binomial", subset = train)
probs <- predict(fit.glm, newdata = Default[-train, ], type = "response")
pred.glm <- rep("No", length(probs))
pred.glm[probs > 0.5] <- "Yes"
mean(pred.glm != Default[-train, ]$default)
```

```
## [1] 0.0236
```

```
train <- sample(dim(Default)[1], dim(Default)[1] / 2)
fit.glm <- glm(default ~ income + balance, data = Default, family = "binomial", subset = train)
probs <- predict(fit.glm, newdata = Default[-train, ], type = "response")
pred.glm <- rep("No", length(probs))
pred.glm[probs > 0.5] <- "Yes"
mean(pred.glm != Default[-train, ]$default)
```

```
## [1] 0.028
```

```
train <- sample(dim(Default)[1], dim(Default)[1] / 2)
fit.glm <- glm(default ~ income + balance, data = Default, family = "binomial", subset = train)
probs <- predict(fit.glm, newdata = Default[-train, ], type = "response")
pred.glm <- rep("No", length(probs))
pred.glm[probs > 0.5] <- "Yes"
mean(pred.glm != Default[-train, ]$default)
```

```
## [1] 0.0268
```

We see that the validation estimate of the test error rate can be variable, depending on precisely which observations are included in the training set and which observations are included in the validation set.

(d) Now consider a logistic regression model that predicts the probability of "default" using "income", "balance", and a dummy variable for "student". Estimate the test error for this model using the validation set approach. Comment on whether or not including a dummy variable for "student" leads to a reduction in the test error rate.

```
train <- sample(dim(Default)[1], dim(Default)[1] / 2)
fit.glm <- glm(default ~ income + balance + student, data = Default, family = "binomial", subset = train
pred.glm <- rep("No", length(probs))
probs <- predict(fit.glm, newdata = Default[-train, ], type = "response")
pred.glm[probs > 0.5] <- "Yes"
mean(pred.glm != Default[-train, ]$default)
```

```
## [1] 0.0264
```

It doesn't seem that adding the "student" dummy variable leads to a reduction in the validation set estimate of the test error rate.

Q6. We continue to consider the use of a logistic regression model to predict the probability of "default" using "income" and "balance" on the "Default" data set. In particular, we will now computes estimates for the standard errors of the "income" and "balance" logistic regression coefficients in two different ways : (1) using the bootstrap, and (2) using the standard formula for computing the standard errors in the glm() function. Do not forget to set a random seed before beginning your analysis.

(a) Using the summary() and glm() functions, determine the estimated standard errors for the coefficients associated with "income" and "balance" in a multiple logistic regression model that uses both predictors.

```
set.seed(1)
attach(Default)
```

```
## The following objects are masked from Default (pos = 3):
##
##     balance, default, income, student
```

```
fit.glm <- glm(default ~ income + balance, data = Default, family = "binomial")
summary(fit.glm)
```

```
##
## Call:
## glm(formula = default ~ income + balance, family = "binomial",
##     data = Default)
##
## Deviance Residuals:
##    Min      1Q  Median      3Q     Max
## -2.473  -0.144  -0.057  -0.021   3.724
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept) -1.15e+01   4.35e-01  -26.54   <2e-16 ***
## income       2.08e-05   4.99e-06    4.17    3e-05 ***
## balance      5.65e-03   2.27e-04   24.84   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 2920.6  on 9999  degrees of freedom
## Residual deviance: 1579.0  on 9997  degrees of freedom
## AIC: 1585
##
## Number of Fisher Scoring iterations: 8
```

The glm() estimates of the standard errors for the coefficients β_0, β_1 and β_2 are respectively 0.4348, 4.9852 × 10-6 and 2.2737 × 10-4.

(b) Write a function, boot.fn(), that takes as input the "Default" data set as well as an index of the observations, and that outputs the coefficient estimates for "income" and "balance" in the multiple logistic regression model.

```
boot.fn <- function(data, index) {
    fit <- glm(default ~ income + balance, data = data, family = "binomial", subset = index)
    return (coef(fit))
}
```

(c) Use the boot() function together with your boot.fn() function to estimate the standard errors of the logistic regression coefficients for "income" and "balance".

```
library(boot)
boot(Default, boot.fn, 1000)
```

```
##
## ORDINARY NONPARAMETRIC BOOTSTRAP
##
##
## Call:
## boot(data = Default, statistic = boot.fn, R = 1000)
##
##
## Bootstrap Statistics :
##       original     bias    std. error
## t1* -1.154e+01 -8.008e-03   4.239e-01
## t2*  2.081e-05  5.871e-08   4.583e-06
## t3*  5.647e-03  2.300e-06   2.268e-04
```

The bootstrap estimates of the standard errors for the coefficients β_0, β_1 and β_2 are respectively 0.4239, 4.583 x 10^(-6) and 2.268 x 10^(-4).

(d) Comment on the estimated standard errors obtained using the glm() function and using your bootstrap function.

The estimated standard errors obtained by the two methods are pretty close.

Q7. In sections 5.3.2 and 5.3.3, we saw that the cv.glm() function can be used in order to compute the LOOCV test error estimate. Alternatively, one could compute those quantities using just the glm() and predict.glm() functions, and a for loop. You will now take this approach in order to compute the LOOCV error for a simple logistic regression model on the "Weekly" data set. Recall that in the context of classification problems, the LOOCV error is given in (5.4).

(a) Fit a logistic regression model that predicts "Direction" using "Lag1" and "Lag2".

```
set.seed(1)
attach(Weekly)
fit.glm <- glm(Direction ~ Lag1 + Lag2, data = Weekly, family = "binomial")
summary(fit.glm)
```

```
##
## Call:
## glm(formula = Direction ~ Lag1 + Lag2, family = "binomial", data = Weekly)
##
## Deviance Residuals:
##    Min      1Q  Median      3Q     Max
##  -1.62   -1.26    1.00    1.08    1.51
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)   0.2212     0.0615    3.60  0.00032 ***
## Lag1         -0.0387     0.0262   -1.48  0.13967
## Lag2          0.0602     0.0265    2.27  0.02323 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1496.2  on 1088  degrees of freedom
## Residual deviance: 1488.2  on 1086  degrees of freedom
## AIC: 1494
##
## Number of Fisher Scoring iterations: 4
```

(b) Fit a logistic regression model that predicts "Direction" using "Lag1" and "Lag2" using all but the first observation.

```
fit.glm.1 <- glm(Direction ~ Lag1 + Lag2, data = Weekly[-1, ], family = "binomial")
summary(fit.glm.1)
```

```
##
## Call:
## glm(formula = Direction ~ Lag1 + Lag2, family = "binomial", data = Weekly[-1,
##     ])
##
## Deviance Residuals:
##    Min      1Q  Median      3Q     Max
##  -1.63   -1.26    1.00    1.08    1.51
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)   0.2232     0.0615    3.63  0.00028 ***
## Lag1         -0.0384     0.0262   -1.47  0.14268
## Lag2          0.0608     0.0266    2.29  0.02197 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1494.6  on 1087  degrees of freedom
## Residual deviance: 1486.5  on 1085  degrees of freedom
## AIC: 1493
##
## Number of Fisher Scoring iterations: 4
```

(c) Use the model from (b) to predict the direction of the first observation. You can do this by predicting that the first observation will go up if $P(\text{direction} = \text{"Up"}|Lag1, Lag2) > 0.5$. Was this observation correctly classified ?

```
predict.glm(fit.glm.1, Weekly[1, ], type = "response") > 0.5
```

```
##    1
## TRUE
```

We may conclude that the prediction for the first observation is "Up". This observation was not correctly classified as the true direction is "Down".

(d) Write a loop from $i = 1$ to $i = n$, where n is the number of observations in the data set, that performs each of the following steps :

i. Fit a logistic regression model using all but the ith observation to predict "Direction" using "Lag1" and "Lag2".

ii. Compute the posterior probability of the market moving up for the ith observation.

iii. Use the posterior probability for the ith observation in order to predict whether or not the market moves up.

iv. Determine whether or not an error was made in predicting the direction for the ith observation. If an error was made, then indicate this as a 1, and otherwise indicate it as a 0.

```
error <- rep(0, dim(Weekly)[1])
for (i in 1:dim(Weekly)[1]) {
    fit.glm <- glm(Direction ~ Lag1 + Lag2, data = Weekly[-i, ],  family = "binomial")
    pred.up <- predict.glm(fit.glm, Weekly[i, ], type = "response") > 0.5
    true.up <- Weekly[i, ]$Direction == "Up"
    if (pred.up != true.up)
        error[i] <- 1
}
error
```

```
##    [1] 1 1 0 1 0 1 0 0 0 1 1 0 1 0 1 0 1 0 1 0 0 0 1 1 1 1 1 1 0 1 1 1 1 0
##   [35] 1 0 0 0 1 0 1 0 0 1 0 1 1 1 0 1 0 0 0 1 0 0 1 1 0 0 0 0 1 0 1 1 0 0
##   [69] 1 0 1 1 0 0 0 1 0 1 1 0 0 1 1 0 1 1 0 0 1 0 0 1 1 1 0 0 0 0 0 1 0 1
##  [103] 1 0 0 1 0 1 0 0 1 1 0 0 1 0 0 1 0 0 1 1 1 1 0 0 0 1 0 1 0 1 1 0 0 0
##  [137] 1 1 1 0 0 0 1 0 0 0 0 0 0 1 1 1 0 1 0 0 1 1 0 1 0 0 1 1 0 0 1 0 0 1
##  [171] 0 0 1 1 1 0 1 0 1 0 0 0 0 0 0 0 0 1 1 0 1 0 1 0 1 0 1 0 0 1 0 0 1 0
##  [205] 0 1 0 1 0 1 1 1 0 0 1 1 0 1 0 0 1 1 0 0 0 1 1 1 0 1 0 1 0 1 0 0 0 1
##  [239] 1 0 1 0 1 0 1 0 1 0 1 1 0 1 0 0 1 0 0 1 0 0 0 0 0 1 0 0 0 1 0 0 1 0
##  [273] 0 0 1 0 0 1 0 0 1 0 0 1 0 1 1 0 0 0 0 0 1 0 1 0 0 1 0 0 0 1 0 0 1 1
##  [307] 0 0 1 0 0 0 0 1 0 1 1 0 0 1 0 1 0 1 1 0 0 0 1 0 1 0 0 1 1 1 1 0 1 0
##  [341] 0 1 0 0 0 1 0 1 0 1 0 0 0 0 0 0 1 1 0 0 1 0 0 1 0 0 0 1 1 0 1 1 1 1 1
##  [375] 0 0 0 1 0 0 0 0 0 0 1 0 1 1 0 0 1 1 0 0 0 0 0 1 0 0 1 1 1 0 1 0 1 0
##  [409] 1 1 1 0 1 0 0 0 0 0 0 0 0 0 0 1 0 1 0 1 0 1 0 0 1 0 1 0 0 0 0 0 1 1
##  [443] 1 1 0 1 1 0 1 0 1 1 0 1 0 0 1 0 0 1 1 0 0 0 0 1 1 0 0 1 0 1 0 0 0 1
##  [477] 0 0 1 0 0 0 1 1 1 0 1 0 0 0 1 0 1 1 1 0 0 0 0 1 1 1 0 1 1 0 1 0 0 0
##  [511] 1 0 1 0 0 0 1 0 1 1 0 0 1 1 0 0 0 1 1 0 1 0 1 1 1 1 1 0 0 0 1 0 0 0
##  [545] 1 1 0 1 0 0 0 1 1 1 1 1 1 0 1 0 1 0 0 1 0 0 1 1 1 0 0 0 1 1 1 1 1 1
```

```
##  [579] 1 1 1 1 0 1 0 0 1 0 0 1 0 1 0 0 1 0 0 1 0 1 1 0 1 1 1 0 1 0 1 0 1 0
##  [613] 0 0 1 0 1 0 1 0 1 1 0 1 1 0 1 0 1 0 1 1 1 1 0 1 1 0 0 0 1 1 1 1 0 1
##  [647] 1 1 0 1 0 0 0 1 1 1 1 1 1 0 1 0 0 1 0 0 0 1 1 0 1 0 1 1 1 1 0 0 0 1
##  [681] 1 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 1 0 0 1 0 1 1 0 0 0 0 1 0 1 0 1 0 1
##  [715] 0 0 1 1 0 0 0 0 0 0 0 0 0 1 0 0 1 0 0 1 1 1 0 1 1 0 1 1 1 1 0 0 0 1
##  [749] 1 1 1 1 1 0 1 0 0 0 0 0 0 1 0 1 1 1 0 0 0 1 0 0 1 0 0 0 1 1 1 0 0 0
##  [783] 1 0 0 1 1 1 0 0 1 0 1 0 1 0 0 1 0 0 1 0 0 0 0 0 1 0 1 1 0 0 1 1 0 1
##  [817] 1 1 0 0 0 0 0 1 1 0 0 1 0 0 1 0 1 0 0 0 1 1 0 1 1 0 1 0 1 0 1 1 0 0
##  [851] 1 1 1 0 1 1 0 0 0 1 0 1 0 1 0 1 0 0 0 0 0 1 0 0 1 1 0 0 1 0 1 0 1 1
##  [885] 0 1 0 1 1 0 0 0 1 0 0 0 0 0 0 0 1 0 1 0 1 0 0 0 1 1 1 1 1 0 1 1 0 0
##  [919] 0 0 0 1 0 0 1 0 0 0 0 1 0 1 1 1 0 0 1 1 0 1 1 1 1 0 1 0 1 0 1 0 1 0
##  [953] 1 0 0 1 1 1 1 1 0 1 0 0 0 1 1 1 0 1 1 1 1 0 0 0 0 1 1 0 0 0 0 0 1 0 0
##  [987] 1 1 1 0 0 1 1 1 0 1 0 0 0 0 1 0 0 1 0 1 0 0 1 1 1 1 0 1 0 0 1 0 0 1
## [1021] 0 0 1 1 0 1 1 1 0 1 1 0 0 0 1 0 1 0 1 1 1 1 0 0 1 0 0 0 0 0 0 1 0 1
## [1055] 1 0 0 0 0 0 0 1 1 1 0 0 0 1 0 1 1 1 0 0 0 0 1 0 0 0 0 0 0 1 0 1 0 0 0 0
## [1089] 0
```

(e) Take the average of the n numbers obtained in (d)iv in order to obtain the LOOCV estimate for the test error. Comment on the results.

```
mean(error)
```

```
## [1] 0.45
```

The LOOCV estimate for the test error rate is 44.9954%.

Q8. We will now perform cross-validation on a simulated data set.

(a) Generate a simulated data set as follows :

```
set.seed(1)
y <- rnorm(100)
x <- rnorm(100)
y <- x - 2 * x^2 + rnorm(100)
```

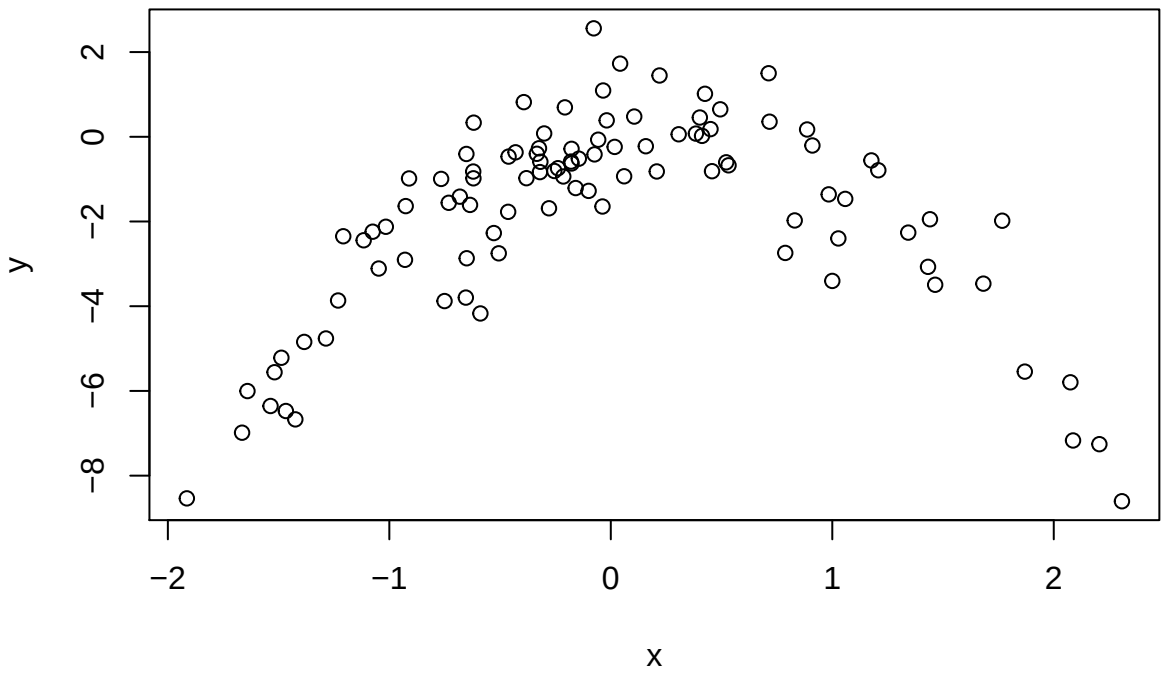
In this data set, what is n and what is p ? Write out the model used to generate the data in equation form.

Here we have that $n = 100$ *and* $p = 2$*, the model used is*

$$Y = X - 2X^2 + \varepsilon.$$

(b) Create a scatterplot of X against Y. Comment on what you find.

```
plot(x, y)
```



The data obviously suggests a curved relationship.

(c) Set a random seed, and then compute the LOOCV errors that result from fitting the following four models using least squares :

i. $Y = \beta_0 + \beta_1 X + \varepsilon$

```
library(boot)
set.seed(1)
Data <- data.frame(x, y)
fit.glm.1 <- glm(y ~ x)
cv.glm(Data, fit.glm.1)$delta[1]
```

```
## [1] 5.891
```

ii. $Y = \beta_0 + \beta_1 X + \beta_2 X^2 + \varepsilon$

```
fit.glm.2 <- glm(y ~ poly(x, 2))
cv.glm(Data, fit.glm.2)$delta[1]
```

```
## [1] 1.087
```

iii. $Y = \beta_0 + \beta_1 X + \beta_2 X^2 + \beta_3 X^3 + \varepsilon$

```
fit.glm.3 <- glm(y ~ poly(x, 3))
cv.glm(Data, fit.glm.3)$delta[1]
```

```
## [1] 1.103
```

iv. $Y = \beta_0 + \beta_1 X + \beta_2 X^2 + \beta_3 X^3 + \beta_4 X^4 + \varepsilon$

```
fit.glm.4 <- glm(y ~ poly(x, 4))
cv.glm(Data, fit.glm.4)$delta[1]
```

```
## [1] 1.115
```

(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(10)
fit.glm.1 <- glm(y ~ x)
cv.glm(Data, fit.glm.1)$delta[1]
```

```
## [1] 5.891
```

```
fit.glm.2 <- glm(y ~ poly(x, 2))
cv.glm(Data, fit.glm.2)$delta[1]
```

```
## [1] 1.087
```

```
fit.glm.3 <- glm(y ~ poly(x, 3))
cv.glm(Data, fit.glm.3)$delta[1]
```

```
## [1] 1.103
```

```
fit.glm.4 <- glm(y ~ poly(x, 4))
cv.glm(Data, fit.glm.4)$delta[1]
```

```
## [1] 1.115
```

The results above are identical to the results obtained in (c) since LOOCV evaluates n folds of a single observation.

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

We may see that the LOOCV estimate for the test MSE is minimum for "fit.glm.2", this is not surprising since we saw clearly in (b) that the relation between "x" and "y" is quadratic.

(f) Comment on the statistical significance of the coefficient estimates that results from fitting each of the models in (c) using least squares. Do these results agree with the conclusions drawn based on the cross-validation results ?

```r
summary(fit.glm.4)
```

```
##
## Call:
## glm(formula = y ~ poly(x, 4))
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.8913  -0.5244   0.0749   0.5932   2.7796
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   -1.828      0.104  -17.55   <2e-16 ***
## poly(x, 4)1    2.316      1.041    2.22    0.029 *
## poly(x, 4)2  -21.059      1.041  -20.22   <2e-16 ***
## poly(x, 4)3   -0.305      1.041   -0.29    0.770
## poly(x, 4)4   -0.493      1.041   -0.47    0.637
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for gaussian family taken to be 1.085)
##
##     Null deviance: 552.21  on 99  degrees of freedom
## Residual deviance: 103.04  on 95  degrees of freedom
## AIC: 298.8
##
## Number of Fisher Scoring iterations: 2
```

The p-values show that the linear and quadratic terms are statistically significants and that the cubic and 4th degree terms are not statistically significants. This agree strongly with our cross-validation results which were minimum for the quadratic model.

Q9. We will now consider the "Boston" housing data set, from the "MASS" library.

(a) Based on this data set, provide an estimate for the population mean of "medv". Call this estimate $\hat{\mu}$.

```r
library(MASS)
```

```
##
## Attaching package: 'MASS'
##
## The following object is masked _by_ '.GlobalEnv':
##
##     Boston
```

```r
attach(Boston)
```

```
## The following objects are masked _by_ .GlobalEnv:
##
##     chas, crim01
```

```
mu.hat <- mean(medv)
mu.hat
```

```
## [1] 22.53
```

(b) Provide an estimate of the standard error of $\hat{\mu}$. Interpret this result.

```
se.hat <- sd(medv) / sqrt(dim(Boston)[1])
se.hat
```

```
## [1] 0.4089
```

(c) Now estimate the standard error of $\hat{\mu}$ using the bootstrap. How does this compare to your answer from (b) ?

```
set.seed(1)
boot.fn <- function(data, index) {
    mu <- mean(data[index])
    return (mu)
}
boot(medv, boot.fn, 1000)
```

```
##
## ORDINARY NONPARAMETRIC BOOTSTRAP
##
##
## Call:
## boot(data = medv, statistic = boot.fn, R = 1000)
##
##
## Bootstrap Statistics :
##     original   bias    std. error
## t1*    22.53 0.008518      0.4119
```

The bootstrap estimated standard error of $\hat{\mu}$ of 0.4119 is very close to the estimate found in (b) of 0.4089.

(d) Based on your bootstrap estimate from (c), provide a 95% confidence interval for the mean of "medv". Compare it to the results obtained using t.test(Boston$medv).

```
t.test(medv)
```

```
##
##  One Sample t-test
##
## data:  medv
## t = 55.11, df = 505, p-value < 2.2e-16
## alternative hypothesis: true mean is not equal to 0
## 95 percent confidence interval:
##  21.73 23.34
## sample estimates:
## mean of x
##     22.53
```

```
CI.mu.hat <- c(22.53 - 2 * 0.4119, 22.53 + 2 * 0.4119)
CI.mu.hat
```

```
## [1] 21.71 23.35
```

The bootstrap confidence interval is very close to the one provided by the t.test() function.

(e) Based on this data set, provide an estimate, $\hat{\mu}_{med}$, for the median value of "medv" in the population.

```
med.hat <- median(medv)
med.hat
```

```
## [1] 21.2
```

(f) We now would like to estimate the standard error of $\hat{\mu}_{med}$. Unfortunately, there is no simple formula for computing the standard error of the median. Instead, estimate the standard error of the median using the bootstrap. Comment on your findings.

```
boot.fn <- function(data, index) {
    mu <- median(data[index])
    return (mu)
}
boot(medv, boot.fn, 1000)
```

```
##
## ORDINARY NONPARAMETRIC BOOTSTRAP
##
##
## Call:
## boot(data = medv, statistic = boot.fn, R = 1000)
##
##
## Bootstrap Statistics :
##     original  bias    std. error
## t1*     21.2 -0.0098      0.3874
```

We get an estimated median value of 21.2 which is equal to the value obtained in (e), with a standard error of 0.3874 which is relatively small compared to median value.

(g) Based on this data set, provide an estimate for the tenth percentile of "medv" in Boston suburbs. Call this quantity $\hat{\mu}_{0.1}$.

```
percent.hat <- quantile(medv, c(0.1))
percent.hat
```

```
##   10%
## 12.75
```

(h) Use the bootstrap to estimate the standard error of $\hat{\mu}_{0.1}$. Comment on your findings.

```r
boot.fn <- function(data, index) {
    mu <- quantile(data[index], c(0.1))
    return (mu)
}
boot(medv, boot.fn, 1000)
```

```
##
## ORDINARY NONPARAMETRIC BOOTSTRAP
##
##
## Call:
## boot(data = medv, statistic = boot.fn, R = 1000)
##
##
## Bootstrap Statistics :
##     original  bias    std. error
## t1*    12.75 0.00515      0.5113
```

We get an estimated tenth percentile value of 12.75 which is again equal to the value obtained in (g), with a standard error of 0.5113 which is relatively small compared to percentile value.