Q2. 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)
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
## -2.4725  -0.1444  -0.0574  -0.0211   3.7245
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept) -1.154e+01  4.348e-01 -26.545  < 2e-16 ***
## income       2.081e-05  4.985e-06   4.174 2.99e-05 ***
## balance      5.647e-03  2.274e-04  24.836  < 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, 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
## -2.3583  -0.1268  -0.0475  -0.0165   3.8116
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept) -1.208e+01  6.658e-01 -18.148   <2e-16 ***
## income       1.858e-05  7.573e-06   2.454   0.0141 *
## balance      6.053e-03  3.467e-04  17.457   <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%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.

Q3. 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
## -2.4725  -0.1444  -0.0574  -0.0211   3.7245
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept) -1.154e+01  4.348e-01 -26.545  < 2e-16 ***
## income       2.081e-05  4.985e-06   4.174 2.99e-05 ***
## balance      5.647e-03  2.274e-04  24.836  < 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 $\beta 0\beta 0$, $\beta 1\beta 1$ *and* $\beta 2\beta 2$ *are respectively 0.4347564, 4.985167210^{-6} and 2.273731410^{-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* -1.154047e+01 -8.008379e-03 4.239273e-01
## t2*  2.080898e-05  5.870933e-08 4.582525e-06
## t3*  5.647103e-03  2.299970e-06 2.267955e-04
```

The bootstrap estimates of the standard errors for the coefficients β0β0, β1β1 *and* β2β2 *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.

Q4. 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 $\hat{\mu}$ μ^.

```
library(MASS)
attach(Boston)
mu.hat <- mean(medv)
mu.hat
## [1] 22.53281
```

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

```
se.hat <- sd(medv) / sqrt(dim(Boston)[1])
se.hat
## [1] 0.4088611
```

c. Now estimate the standard error of $\hat{\mu}$ μ^ 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* 22.53281 0.008517589   0.4119374
```

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 = 55.1111, df = 505, p-value < 2.2e-16
## alternative hypothesis: true mean is not equal to 0
## 95 percent confidence interval:
##  21.72953 23.33608
## sample estimates:
## mean of x
##  22.53281
CI.mu.hat <- c(22.53 - 2 * 0.4119, 22.53 + 2 * 0.4119)
CI.mu.hat
## [1] 21.7062 23.3538
```

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   0.3874004
```

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 $\hat{\mu}_{0.1}$ μ^0.1.

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

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

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

```
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   0.5113487
```

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.