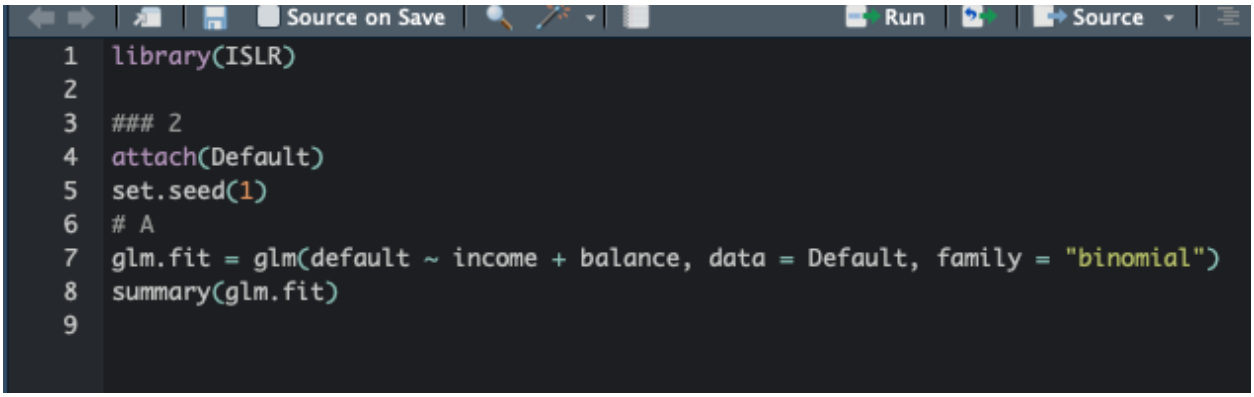
Jingtang Ma
CS465
Lab4

2. (40 points total) In Chapter 4, we used logistic 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) (10 points) Fit a logistic regression model that uses income and balance to predict default.

CODE:



OUTPUT:

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

(i) (2.5 points) Split the sample set into a training set and a validation set.

(ii) (2.5 points) Fit a multiple logistic regression model using only the training observations.

(iii) (2.5 points) 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.

(iv) (2.5 points) Compute the validation set error, which is the fraction of the observations in the validation set that are misclassified.

CODE:

```
## B
# (i)
train = sample(dim(Default)[1], dim(Default)[1]*0.8)
#test =

# (ii)
glm.fit = glm(default ~ income + balance, family = "binomial", subset = train)
summary(glm.fit)

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

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

OUTOUT:

```
> mean(glm.pred != Default[-train, ]$default)
[1] 0.026
>
```

(c) (10 points) 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.

Ans: When train:validation is 9:1, the test error is 3.9%. When train:validation is 7:3, the test error is 2.5%. When train:validation is 5:5, the test error is 2.7%. Test errors depends on the different splits.

CODE:

```
## C
# Train:Validation = 9:1
train = sample(dim(Default)[1], dim(Default)[1]*0.9)
glm.fit = glm(default ~ income + balance, family = "binomial", subset = train)
prob = predict(glm.fit, newdata = Default[-train, ], type = "response")
glm.pred = rep("No", length(prob))
glm.pred[prob > 0.5] = "Yes"
mean(glm.pred != Default[-train, ]$default)
# Train:Validation = 7:3
train = sample(dim(Default)[1], dim(Default)[1]*0.7)
glm.fit = glm(default ~ income + balance, family = "binomial", subset = train)
prob = predict(glm.fit, newdata = Default[-train, ], type = "response")
glm.pred = rep("No", length(prob))
glm.pred[prob > 0.5] = "Yes"
mean(glm.pred != Default[-train, ]$default)
# Train:Validation = 5:5
train = sample(dim(Default)[1], dim(Default)[1]*0.5)
glm.fit = glm(default ~ income + balance, family = "binomial", subset = train)
prob = predict(glm.fit, newdata = Default[-train, ], type = "response")
glm.pred = rep("No", length(prob))
glm.pred[prob > 0.5] = "Yes"
mean(glm.pred != Default[-train, ]$default)
```

OUTPUT:

```
> mean(glm.pred != Default[-train, ]$default)
[1] 0.039
> # Train:Validation = 7:3
> train = sample(dim(Default)[1], dim(Default)[1]*0.7)
> glm.fit = glm(default ~ income + balance, family = "binomial", subset = train)
> prob = predict(glm.fit, newdata = Default[-train, ], type = "response")
> glm.pred = rep("No", length(prob))
> glm.pred[prob > 0.5] = "Yes"
> mean(glm.pred != Default[-train, ]$default)
[1] 0.02533333
> # Train:Validation = 5:5
> train = sample(dim(Default)[1], dim(Default)[1]*0.5)
> glm.fit = glm(default ~ income + balance, family = "binomial", subset = train)
> prob = predict(glm.fit, newdata = Default[-train, ], type = "response")
> glm.pred = rep("No", length(prob))
> glm.pred[prob > 0.5] = "Yes"
> mean(glm.pred != Default[-train, ]$default)
[1] 0.0266
>
```

(d) (10 points) 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.

Ans: As we tested before, when train:test is 8:2, the test error is 2.6% without the dummy variable. However, when the same train:test is 8:2, the test error is 2.2% including the dummy variable. There is a little different when we added the dummy variable.

CODE:

```
## D
train = sample(dim(Default)[1], dim(Default)[1]*0.8)
glm.fit = glm(default ~ income + balance + student, family = "binomial", subset =
prob = predict(glm.fit, newdata = Default[-train, ], type = "response")
glm.pred = rep("No", length(prob))
glm.pred[prob > 0.5] = "Yes"
mean(glm.pred != Default[-train, ]$default)
```

OUTPUT:

```
> mean(glm.pred != Default[-train, ]$default)
[1] 0.0215
```

3. (40 points) 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 compute 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) (10 points) 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.

CODE:

```
### Q3 ###
## A
set.seed(1)
glm.fit = glm(default ~ income + balance, data = Default, family = "binomial")
summary(glm.fit)
```

OUTPUT:

```
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) (10 points)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.

CODE:

```
## B
boot.fn = function(dataset, i) {
  fit = glm(default ~ income + balance, data = dataset, family = "binomial", subset = i)
  return (coef(fit))
}
```

(c) (10 points) Use the boot() function together with your boot.fn() function to estimate the standard errors of the logistic regression coefficients for income and balance.

CODE:

```
## C
library(boot)
boot(Default, boot.fn, 100)
```

OUTPUT:

```
> boot(Default, boot.fn, 100)

ORDINARY NONPARAMETRIC BOOTSTRAP


Call:
boot(data = Default, statistic = boot.fn, R = 100)


Bootstrap Statistics :
        original        bias     std. error
t1* -1.154047e+01  1.412716e-02 4.028164e-01
t2*  2.080898e-05 -1.245422e-07 4.903698e-06
t3*  5.647103e-03 -1.096080e-05 2.231952e-04
>
```

(d) (10 points) Comment on the estimated standard errors obtained using the glm() function and using your bootstrap function.

Ans: There are almost same betws these two models by using R = 100.

4. (40 points) We will now consider the Boston housing data set, from the MASS library.

(a) (5 points) Based on this data set, provide an estimate for the population mean of medv. Call this estimate μˆ.

CODE:

```
### Q4 ###
library(MASS)
attach(Boston)
## A
u = mean(medv)
u
```

OUTPUT:

```
> u
[1] 22.53281
>
```

(b) Provide an estimate of the standard error of μˆ. Interpret this result.
Hint: We can compute the standard error of the sample mean by dividing the sample standard deviation by the square root of the number of observations.

CODE;

```
## B
standardError = sd(medv) / sqrt(dim(Boston)[1])
standardError
```

OUTPUT:

```
> standardError
[1] 0.4088611
```

(c) (5 points) Now estimate the standard error of $\hat{\mu}$ using the bootstrap. How does this compare to your answer from (b)?

Ans: The standard error of $\hat{\mu}$ using the bootstrap is 0.43, the standard error of $\hat{\mu}$ without using the bootstrap is 0.41. Pretty close.

CODE:

```
## C
boot.fn = function(data, i) {
  u = mean(data[i])
  return (u)
}
boot(medv, boot.fn, 100)
```

OUTPUT:

```
Bootstrap Statistics :
    original      bias    std. error
t1* 22.53281 -0.01845257   0.4313303
>
```

(d) (5 points) 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). Hint: You can approximate a 95% confidence interval using the formula $[\hat{\mu} - 2SE(\hat{\mu}), \hat{\mu} + 2SE(\hat{\mu})]$.

Ans: Depends on the results. We can see the t.test() result is very close to the confidenceIn.

CODE:

```
## D
t.test(medv)
confidenceIn = c(22.53281 - 2 * 0.4313303, 22.53281 + 2 * 0.4313303)
confidenceIn
```

OUTPUT:

```
> t.test(medv)

        One Sample t-test

data:  medv
t = 55.111, 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

> confidenceIn = c(22.53281 - 2 * 0.4313303, 22.53281 + 2 * 0.4313303)
> confidenceIn
[1] 21.67015 23.39547
>
```

(e) (5 points) Based on this data set, provide an estimate, μˆ of medv in the population.

CODE:

```
## E
med = median(medv)
med
```

OUTPUT:

```
> med
[1] 21.2
>
```

(f) (5 points) We now would like to estimate the standard error of μˆ Unfortunately, there is no simple formula for computing the standard error of the median. In- stead, estimate the standard error of the median using the bootstrap. Comment on your findings.

Ans: The median value is 21.2 with the std.error 0.387.

CODE:

```
## F
boot.fn = function(data, i) {
  u = median(data[i])
  return (u)
}
boot(medv, boot.fn, 100)
```

OUTPUT:

```
Bootstrap Statistics :
    original  bias    std. error
t1*     21.2   0.018   0.3948865
>
```

(g) Based on this data set, provide an estimate for the tenth percentile of medv in Boston suburbs. Call this quantity μˆ (You can use the quantile() function.)

CODE:

```
## G
percent = quantile(medv, c(0.1))
percent
```

OUTPUT:

```
> percent
  10%
12.75
```

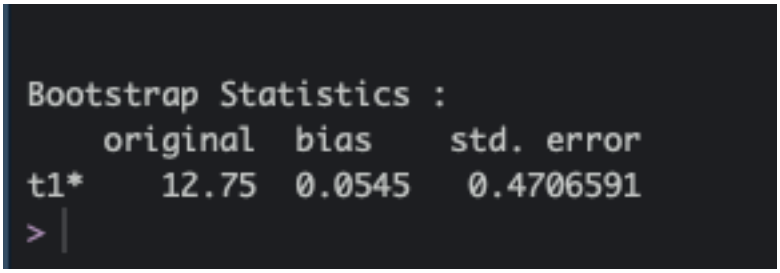
(h) (5 points) Use the bootstrap to estimate the standard error of μˆ . Comment med on your findings.

Ans: The percentile value is 12.75 with the std.error 0.47.

CODE:

```
## H
boot.fn = function(data, i) {
  u = quantile(data[i], c(0.1))
  return (u)
}
boot(medv, boot.fn, 100)
```

OUTPUT:



##################################END##
###

ALL OF CODE IS ON MY GITHUB:
https://github.com/arthurmjt/CS465_Introduction-to-Statistical-Learning.git