Reihaneh Moghisi Assigment 3
RMI 8300

Question 1

a-

```
mean(Boston$medv)
## [1] 22.53281
```

b-

```
sd(Boston$medv) / sqrt(length(Boston$medv))
## [1] 0.4088611
```

c-

They are not identical, but very close to each other as the estimate for SE($\hat{\mu}$) is the same to above: **0.4082** vs **0.4089**.

```
boot.fn <- function(vector, index) {
  mean(vector[index])
}
set.seed(66)

(boot_results <- boot(data = Boston$medv, statistic = boot.fn, R = 1000))
##
## ORDINARY NONPARAMETRIC BOOTSTRAP
##
## Call:
## boot(data = Boston$medv, statistic = boot.fn, R = 1000)
##
## Bootstrap Statistics :
##     original    bias    std. error
## t1* 22.53281 0.0116587   0.4081538
```

d-

calculate the **bootstrap** 95% confidence interval using the hint:

```
boot_results_SE <- sd(boot_results$t)
round(c(mean(Boston$medv) - 2*boot_results_SE, mean(Boston$medv) + 2*boot_res
ults_SE), 4)
## [1] 21.7165 23.3491
```

```
t.test(Boston$medv)
##
##  One Sample t-test
##
## data:  Boston$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
```

The confidence interval estimates are the same to previous part.

e-

We estimate the population median of `medv` by taking the sample median, so $\hat{\mu}_{med}$ is given by:

```
median(Boston$medv)
## [1] 21.2
```

f-

We can use the identical code as in part c), but replacing `mean()` with `median()`:

```
boot.fn <- function(vector, index) {
  median(vector[index])
}

set.seed(77, sample.kind = "Rounding")

(boot_results <- boot(data = Boston$medv, statistic = boot.fn, R = 1000))
##
## ORDINARY NONPARAMETRIC BOOTSTRAP
##
##
## Call:
## boot(data = Boston$medv, statistic = boot.fn, R = 1000)
##
##
## Bootstrap Statistics :
##     original  bias    std. error
## t1*     21.2  0.0094   0.3700318
```

As with the mean, the standard error is quite small relative to the estimate.

g-

As before, we calculate the *sample* tenth percentile as our estimate:

```
quantile(Boston$medv, 0.1)
##   10%
## 12.75
```

h-

```
boot.fn <- function(vector, index) {
  quantile(vector[index], 0.1)
}


set.seed(77, sample.kind = "Rounding")


(boot_results <- boot(data = Boston$medv, statistic = boot.fn, R = 1000))
##
## ORDINARY NONPARAMETRIC BOOTSTRAP
##
##
## Call:
## boot(data = Boston$medv, statistic = boot.fn, R = 1000)
##
##
## Bootstrap Statistics :
##     original  bias    std. error
## t1*    12.75 0.02085    0.488873
```

The standard error is slightly larger relative to $\hat{\mu}_{0.1}$, but it is still small.

Question 2-

a-

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


summary(log_def)$coefficients[, 2]
##   (Intercept)       income      balance
## 4.347564e-01 4.985167e-06 2.273731e-04
```

B –

```
boot.fn <- function(data, index) {
  coef(glm(default ~ income + balance, data = data, subset = index, family =
"binomial"))[-1]

}


boot.fn(Default)
##        income      balance

## 2.080898e-05 5.647103e-03
```

C –

```
set.seed(101, sample.kind = "Rounding")


(boot_results <- boot(data = Default, statistic = boot.fn, R = 1000))
##

## ORDINARY NONPARAMETRIC BOOTSTRAP

## Call:

## boot(data = Default, statistic = boot.fn, R = 1000)

## Bootstrap Statistics :

##          original        bias     std. error

## t1* 2.080898e-05 9.797648e-08 4.710812e-06

## t2* 5.647103e-03 1.345215e-05 2.293757e-04
```

d-

`boot()` Method:

```
sapply(data.frame(income = boot_results$t[ ,1], balance = boot_results$t[ ,2]
), sd)
##        income      balance

## 4.710812e-06 2.293757e-04
```

`glm()` Method:

```
summary(log_def)$coefficients[2:3, 2]
##       income      balance
## 4.985167e-06 2.273731e-04
```

We can see that the standard error estimates for `income` are equal to ones from glm() function. and the estimates for `balance` are equal to glm. This could be an indication that the assumptions of the logistic SE estimates are well-satisfied.

Question 3-

a-

```
log_dir <- glm(Direction ~ Lag1 + Lag2, data = Weekly, family = "binomial")
summary(log_dir)
##
## Call:
## glm(formula = Direction ~ Lag1 + Lag2, family = "binomial", data = Weekly)
##
## Deviance Residuals:
##    Min      1Q  Median      3Q     Max
## -1.623  -1.261   1.001   1.083   1.506
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)  0.22122    0.06147   3.599 0.000319 ***
## Lag1        -0.03872    0.02622  -1.477 0.139672
## Lag2         0.06025    0.02655   2.270 0.023232 *
## ---
## 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: 1494.2
##
## Number of Fisher Scoring iterations: 4
```

b-

```
log_dir_2 <- glm(Direction ~ Lag1 + Lag2, data = Weekly[-1, ], family = "bino
mial")
summary(log_dir_2)
```

```
##
## Call:
## glm(formula = Direction ~ Lag1 + Lag2, family = "binomial", data = Weekly[
-1,
##     ])
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.6258  -1.2617   0.9999   1.0819   1.5071
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)  0.22324    0.06150   3.630 0.000283 ***
## Lag1        -0.03843    0.02622  -1.466 0.142683
## Lag2         0.06085    0.02656   2.291 0.021971 *
## ---
## 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: 1492.5
```

```
##
## Number of Fisher Scoring iterations: 4
```

C-

The prediction for this first observation:

```
ifelse(predict(log_dir_2, newdata = Weekly[1, ], type = "response") > 0.5, "U
p", "Down")
##    1
## "Up"
```

The *actual* `Direction`:

```
as.character(Weekly[1, "Direction"])
## [1] "Down"
```

Therefore the observation was *not* correctly classified.

d-

```
error <- c()


for (i in 1:nrow(Weekly)) {
  log_dir <- glm(Direction ~ Lag1 + Lag2, data = Weekly[-i, ], family = "bino
mial") # i.
  prediction <- ifelse(predict(log_dir, newdata = Weekly[i, ], type = "respon
se") > 0.5, "Up", "Down") # ii. & iii.
  error[i] <- as.numeric(prediction != Weekly[i, "Direction"]) # iv.
}


error[1:10]
##  [1] 1 1 0 1 0 1 0 0 0 1
```

e-

```
mean(error)
## [1] 0.4499541
```

The LOOCV estimate for the test error is ≈ **0.45**. Looking at the split of `Direction` between `Up` and `Down`:

```
prop.table(table(Weekly$Direction))
##
##      Down        Up
## 0.4444444 0.5555556
```

We can see that a classifier that predicted that the `Direction` of the S&P 500 will go `Up` every week between 1990 and 2010 would have achieved an error of **0.444**. therefore using LOOCV estimate will not result in good classification outcome.

Question 4-

a-

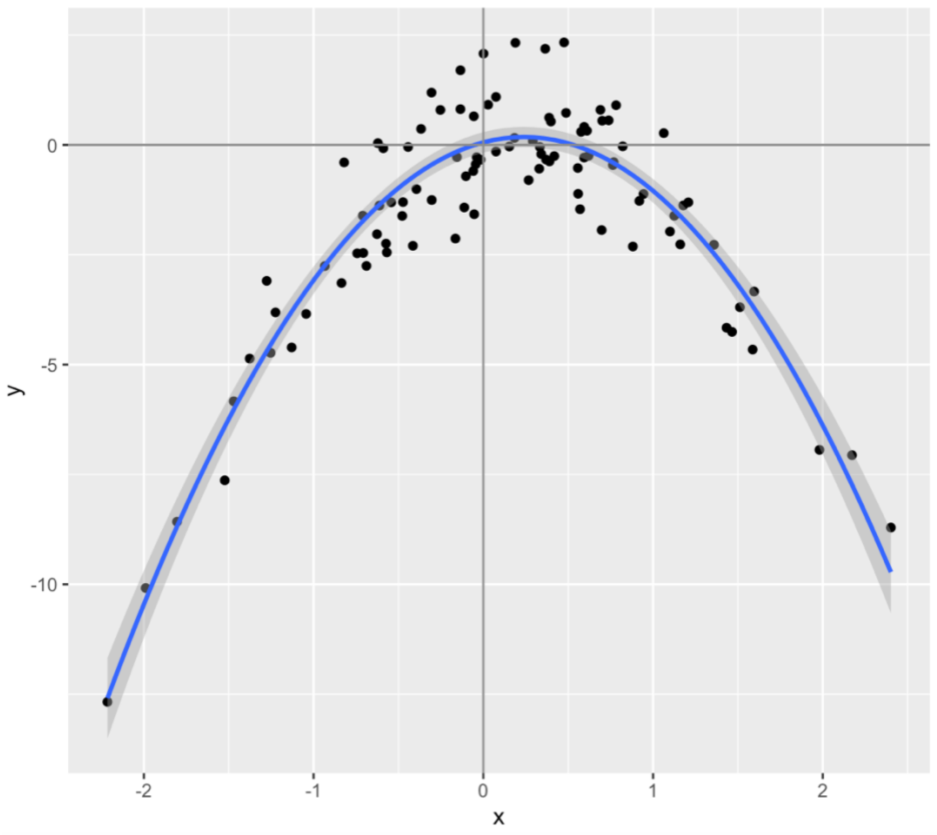
We have n = 100 and p = 2.
The model used to generate the data is:

$$Y=X-2X_2+\epsilon\epsilon\sim\square(0,1)$$

b-

This means that, since $\epsilon\sim\square(0,1)$ is centered at 0, we should expect the data to show a quadratic relationship between X and Y, with a maximum at $\approx X=0.25$, and crossing the x-axis at $\approx X=0$ and $X=0.5$.
We can see that the fitted model which uses least squares linear model, fits the data.

Reihaneh Moghisi Assigment 3
RMI 8300



C-

The LOOCV estimate using the `cv.glm()` function:

i. $Y=\beta_0+\beta_1X+\epsilon$

```
glm_1 <- glm(y ~ x , data = data)
cv_glm_1 <- cv.glm(data, glm_1)
LOOCV_MSE <- cv_glm_1$delta[1]
LOOCV_MSE
## [1] 7.288
```

ii. $Y=\beta_0+\beta_1X+\beta_2X_2+\epsilon$

```
glm_1 <- glm(y ~ x + I(x^2), data = data)
cv_glm_1 <- cv.glm(data, glm_1)
```

```
LOOCV_MSE <- cv_glm_1$delta[1]
LOOCV_MSE
## [1] 0.937
```

iii. $Y=\beta_0+\beta_1X+\beta_2X_2+\beta_3X_3+\epsilon$

```
glm_1 <- glm(y ~ x + I(x^2)+ I(x^3), data = data)
cv_glm_1 <- cv.glm(data, glm_1)
LOOCV_MSE <- cv_glm_1$delta[1]
LOOCV_MSE
## [1] 0.957
```

iv. $Y=\beta_0+\beta_1X+\beta_2X_2+\beta_3X_3+\beta_4X_4+\epsilon$

```
glm_1 <- glm(y ~ x + I(x^2)+ I(x^3)+ I(x^4), data = data)
cv_glm_1 <- cv.glm(data, glm_1)
LOOCV_MSE <- cv_glm_1$delta[1]
LOOCV_MSE
## [1] 0.954
```

d- LOOCV and Random Seeds

We get the same results. This is due to the absence of any randomness in LOOCV. This means that there is only one error statistics in LOOCV. Its different from k-fold CV in that the initial partition is not affected by random state. The LOOCV error can only be calculated one way, and if it was impacted by different seed the LOOCV MSE estimate would not have been reduced to a simple computation from a single least square model fit.

```
set.seed(2, sample.kind = "Rounding")
## [1] 7.288
```

```
## [1] 0.937
## [1] 0.957
```

```
## [1] 0.954
```

e- LOOCV Error Comparison

The second order displays the smallest LOOCV error, despite the fact that the second, third, and fourth order polynomial fits exhibit nearly identical performance. This outcome is expected, as observed in part 1. It is reasonable to expect the benefit of upgrading from a first to second - order outweigh the penalty of going from second to third or fourth-order.

f- Coefficient Significance & Cross-Validation MSE

At 5% significance level there is no proof that the coefficients for X3 and X4 are not zero. This aligns with LOOCV error outcome, indicating that selecting the second-order fit is reasonable and there is no justification for selecting third or foruth-order.

i. $Y=\beta_0+\beta_1X+\epsilon$

```
summary(glm(y ~ x, data = data))$coefficients
##               Estimate Std. Error   t value Pr(>|t|)
## (Intercept) -1.625427   0.261937 -6.205420 0.000000
## x            0.692497   0.290942  2.380191 0.019238
```

ii. $Y=\beta_0+\beta_1X+\beta_2X_2+\epsilon$

```
summary(glm(y ~ x + I(x^2), data = data))$coefficients
##               Estimate Std. Error    t value Pr(>|t|)
## (Intercept)  0.056715   0.117655   0.482043 0.630861
## x            1.017161   0.107983   9.419666 0.000000
## I(x^2)       -2.118921   0.084766 -24.997388 0.000000
```

iii. $Y=\beta_0+\beta_1X+\beta_2X_2+\beta_3X_3+\epsilon$

```
summary(glm(y ~ x + I(x^2) + I(x^3), data = data))$coefficients
##               Estimate Std. Error    t value Pr(>|t|)
## (Intercept)   0.061507   0.119504   0.514688 0.607954
## x             0.975280   0.187281   5.207564 0.000001
## I(x^2)       -2.123791   0.087003 -24.410686 0.000000
## I(x^3)        0.017639   0.064290   0.274358 0.784399
```

iv. $Y=\beta_0+\beta_1X+\beta_2X_2+\beta_3X_3+\beta_4X_4+\epsilon$

```
summary(glm(y ~ x + I(x^2) + I(x^3) + I(x^4), data = data))$coefficients
##               Estimate Std. Error    t value Pr(>|t|)
## (Intercept)   0.156703   0.139462   1.123625 0.264003
## x             1.030826   0.191337   5.387500 0.000001
## I(x^2)       -2.409898   0.234855 -10.261215 0.000000
## I(x^3)       -0.009133   0.067229  -0.135848 0.892229
## I(x^4)        0.069785   0.053240   1.310769 0.193096
```