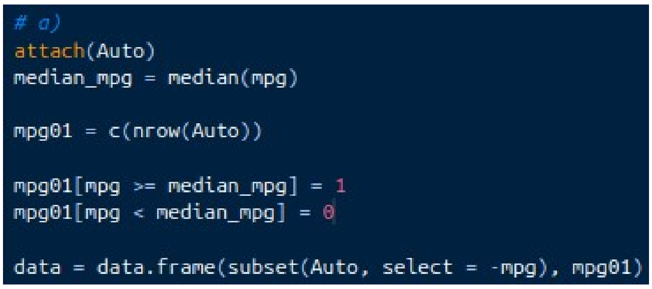
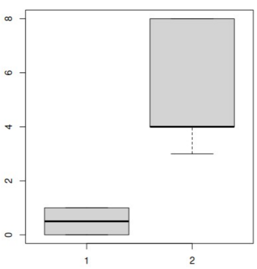
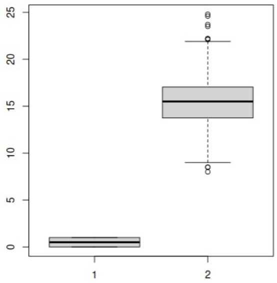
a) I used the median function to pull out the median from the Auto$mpg column, and then used two lines to create a vector of 0s and 1s based on if the mpg is > median_mpg



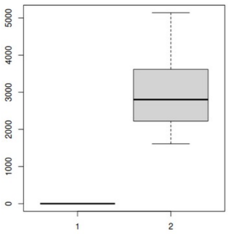
```
# a)
attach(Auto)
median_mpg = median(mpg)

mpg01 = c(nrow(Auto))

mpg01[mpg >= median_mpg] = 1
mpg01[mpg < median_mpg] = 0

data = data.frame(subset(Auto, select = -mpg), mpg01)
```

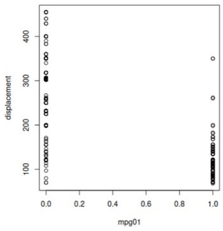
b) Here are some plots in order of how I checked some relationships: Box between mpg01 and # of cylinders, Box between mpg01 and acceleration, Box between mpg01 and weight, and scatterplot between mpg01 and displacement. The box plots all show a positive relationship between mpg01 and the corresponding measure, while the plot shows a negative relationship.



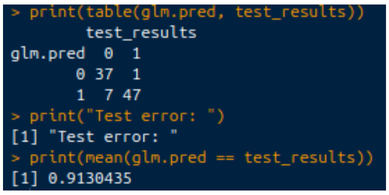


c) I created an index column in a set of the randomized data, and then for the training set took the first 300 values, and test set the last 92 values





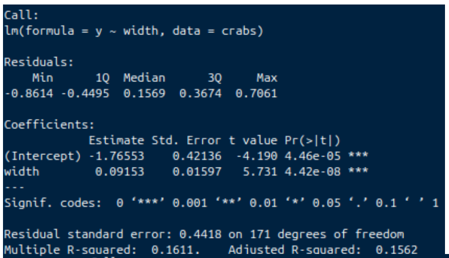
f) After fitting a logic model, I created a confusion matrix and found the mean of correct guesses for mpo01 based on cylinders, displacement, weight, and acceleration. Pretty decent test results!



```
> print(table(glm.pred, test_results))
        test_results
glm.pred  0  1
       0 37  1
       1  7 47
> print("Test error: ")
[1] "Test error: "
> print(mean(glm.pred == test_results))
[1] 0.9130435
```

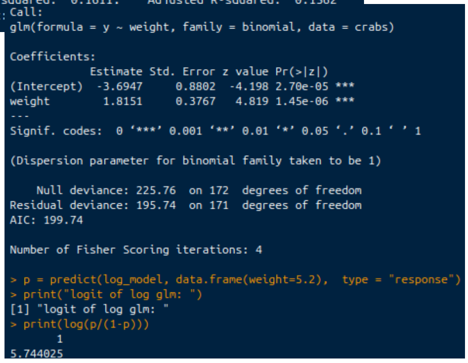
Question AG 3.7

a) The linear model performed very poorly, with an R^2 of 1.562. Both coefficients were considered significant, but overall the model was not good.



```
Call:
lm(formula = y ~ width, data = crabs)

Residuals:
    Min      1Q  Median      3Q     Max
-0.8614 -0.4495  0.1569  0.3674  0.7061

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) -1.76553    0.42136  -4.190 4.46e-05 ***
width        0.09153    0.01597   5.731 4.42e-08 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.4418 on 171 degrees of freedom
Multiple R-squared:  0.1611,    Adjusted R-squared:  0.1562
F-statistic:
```

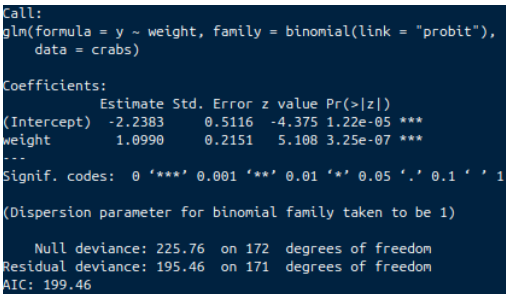
c) By fitting the model to a logistic regression, both coefficients are still considered significant. By manipulating the p(x) we can get the logit of weight = 5.2 as shown:



```
Call:
glm(formula = y ~ weight, family = binomial, data = crabs)

Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept)  -3.6947     0.8802  -4.198 2.70e-05 ***
weight        1.8151     0.3767   4.819 1.45e-06 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 225.76  on 172  degrees of freedom
Residual deviance: 195.74  on 171  degrees of freedom
AIC: 199.74

Number of Fisher Scoring iterations: 4

> p = predict(log_model, data.frame(weight=5.2),  type = "response")
> print("logit of log glm: ")
[1] "logit of log glm: "
> print(log(p/(1-p)))
       1
5.744025
```

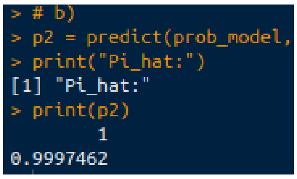
Question AG 3.8

a) The probit model performed just as well as the regular logistic regression model did.



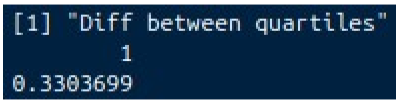
```
Call:
glm(formula = y ~ weight, family = binomial(link = "probit"),
    data = crabs)

Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept)  -2.2383     0.5116  -4.375 1.22e-05 ***
weight        1.0990     0.2151   5.108 3.25e-07 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 225.76  on 172  degrees of freedom
Residual deviance: 195.46  on 171  degrees of freedom
AIC: 199.46
```

b) Our prediction for pi_hat is very confidently close to 1



```
> # b)
> p2 = predict(prob_model,
> print("Pi_hat:")
[1] "Pi_hat:"
> print(p2)
        1
0.9997462
```

c) The difference between quartiles accounts for a 33% change in probability that the crab has a satellite



```
[1] "Diff between quartiles"
        1
0.3303699
```