HW 8 Solutions

A. Again, let's consider the fat data. (You might want to re-read the previous homework assignment.) This time we're going to evaluate how we might answer the question "which model has the best accuracy in making prediction."

a) Last time you fit a model to predict percentage of fat. If you remember that model, write it down here. If not, re-fit to get a reasonable model.
We'll call this "Model 1"

Model 1 (see solutions to HW 7):

predicted.body.fat = -49 + .18(thigh) - .16(weight) + .92 abdomen

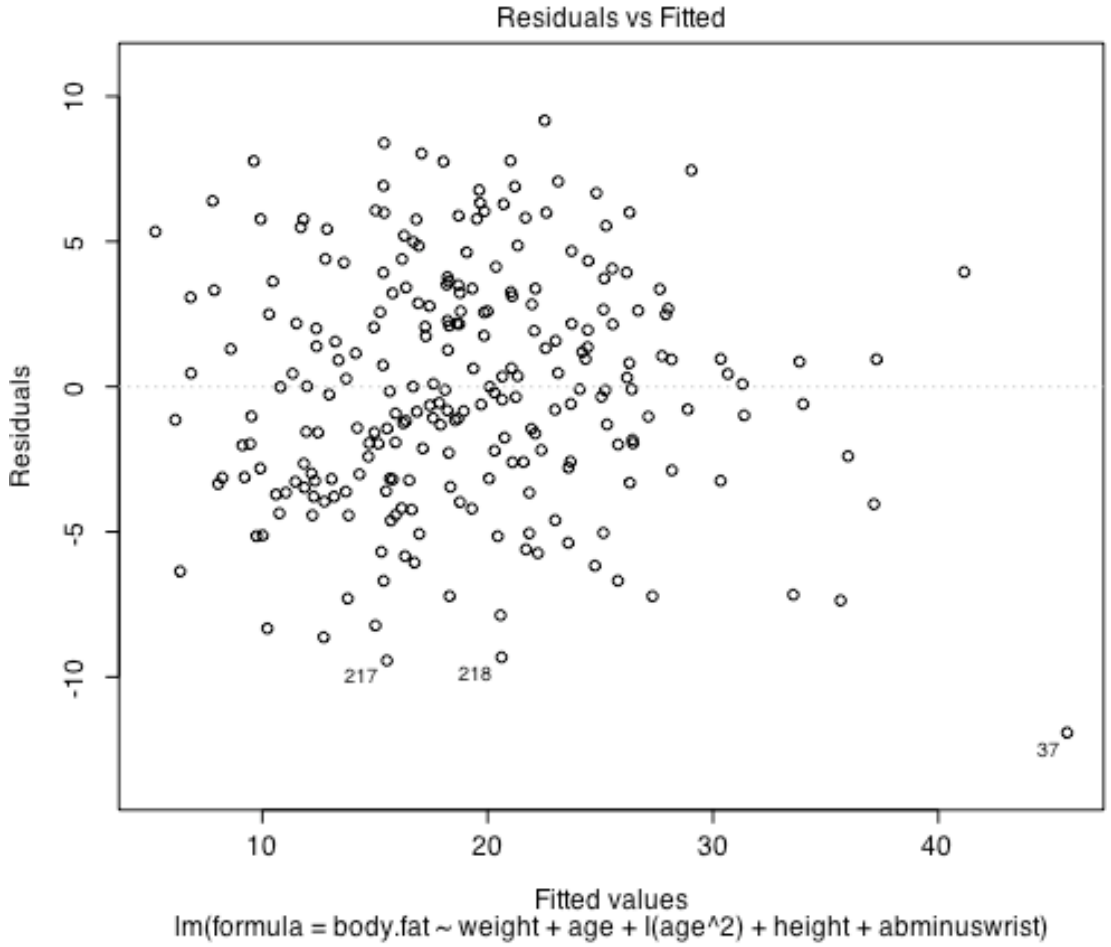
b) Now we'll create "Model 2". Fit a model using (i) weight, (ii) age, (iii) age^2, (iv) height and (v) abdomen minus wrist. No need to take transforms. Just fit the model.

The previous model was fit with some problem-points removed. I removed the same observations from this data set before fitting.

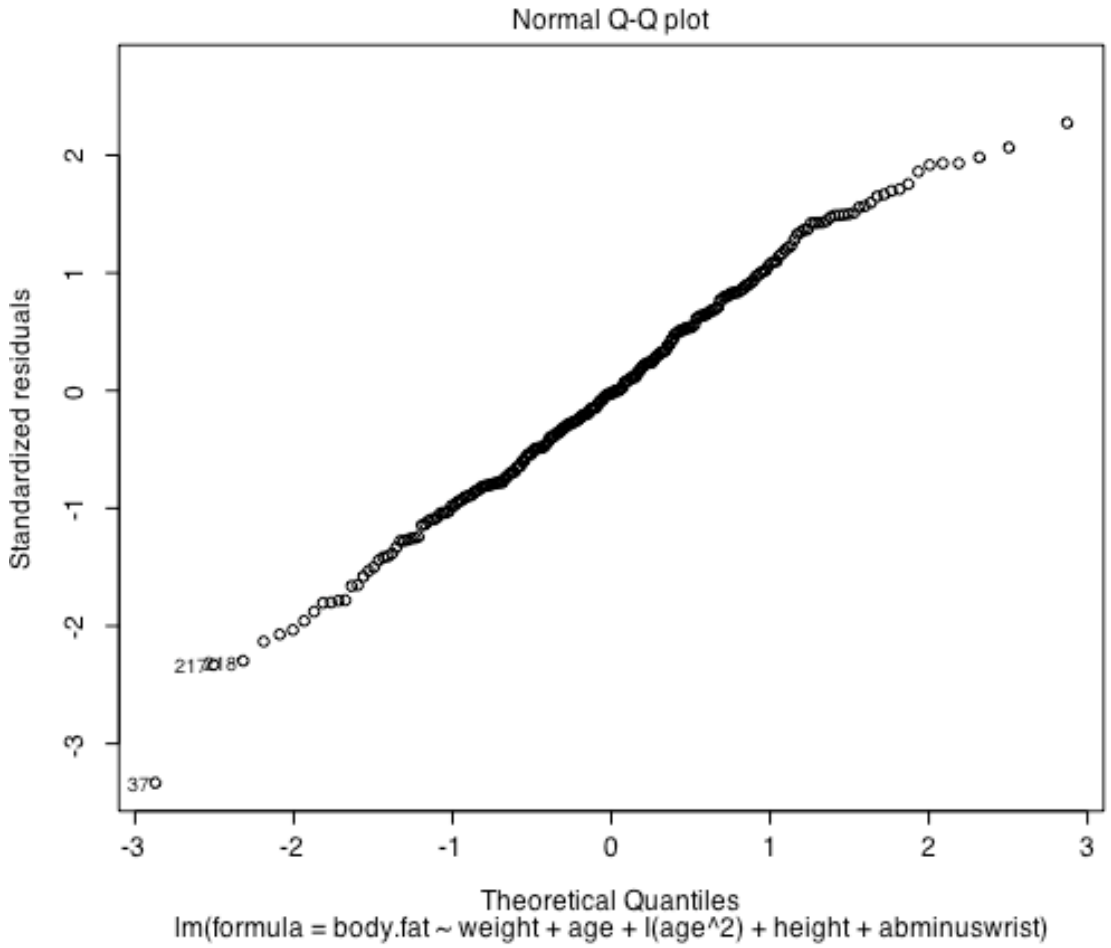
```
Call:
lm(formula = body.fat ~ weight + age + I(age^2) + height + abminuswrist)

Residuals:
     Min       1Q   Median       3Q      Max
-11.91601  -3.01677  -0.09395   2.87555   9.16761

Coefficients:
              Estimate Std. Error t value Pr(>|t|)
(Intercept)  -2.668e+01  1.018e+01  -2.621  0.00932 **
weight       -1.088e-01  2.734e-02  -3.980 9.15e-05 ***
age           2.443e-02  1.275e-01   0.192  0.84824
I(age^2)     -1.231e-04  1.341e-03  -0.092  0.92694
height       -3.191e-02  1.431e-01  -0.223  0.82379
abminuswrist  8.947e-01  7.198e-02  12.429  < 2e-16 ***
---
Signif. codes:  0 `***' 0.001 `**' 0.01 `*' 0.05 `.' 0.1 ` ' 1

Residual standard error: 4.079 on 239 degrees of freedom
Multiple R-Squared: 0.7288,  Adjusted R-squared: 0.7231
F-statistic: 128.4 on 5 and 239 DF,  p-value: < 2.2e-16
```

c) Evaluate how well Model 2 fits the data, using standard

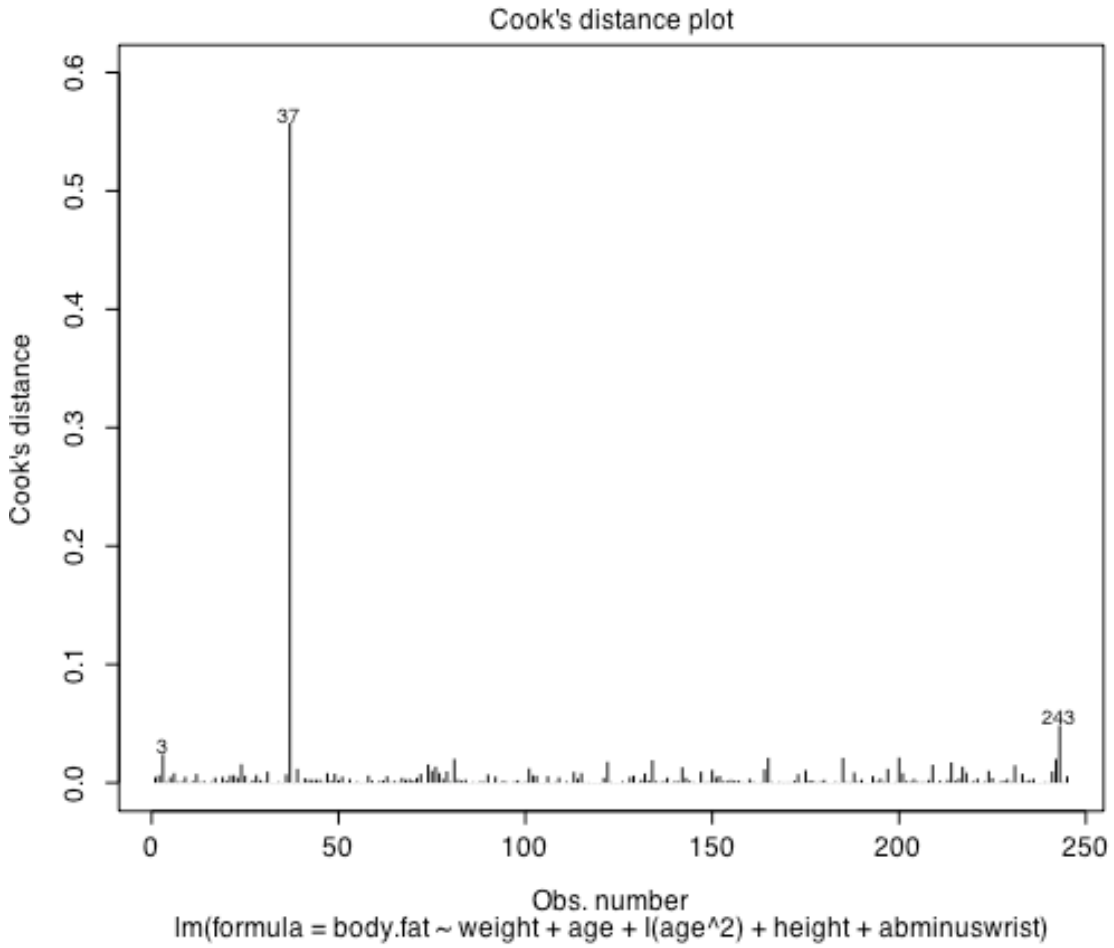
diagnostics.




The residual plot suggests that (a) the mean body fat does seem to depend on the predictors in a linear fashion (since there are no obvious trends in the residuals) and (b) that the variance of the residuals is constant across the fitted values. That said, there is a bit of a "tail" for those with predicted fat over 30% that suggests that the model may not fit as well for these people. But there are so few data points it is hard to tell.




The qqnorm plot suggests the residuals follow a normal distribution.




Observations 37 (the 350 pound person) is still influential.

Removing this point gives a slightly different model. In particualr, weight becomes a less important factor in determining body fat:

```
Call:
lm(formula = body.fat ~ weight + age + I(age^2) + height + abminuswrist2)

Residuals:
    Min      1Q  Median      3Q     Max
-9.5993 -2.8156 -0.1987  3.0225  8.9719

Coefficients:
              Estimate Std. Error t value Pr(>|t|)
(Intercept)   -2.023e+01  1.014e+01  -1.995   0.0472 *
weight        -7.395e-02  2.864e-02  -2.582   0.0104 *
age            5.564e-02  1.251e-01   0.445   0.6570
I(age^2)      -4.187e-04  1.315e-03  -0.318   0.7506
height        -1.681e-01  1.457e-01  -1.154   0.2496
abminuswrist2  8.434e-01  7.203e-02  11.709   <2e-16 ***
---
Signif. codes:  0 `***' 0.001 `**' 0.01 `*' 0.05 `.' 0.1 ` ' 1
```

```
Residual standard error: 3.992 on 238 degrees of freedom
Multiple R-Squared: 0.7374,  Adjusted R-squared: 0.7319
F-statistic: 133.7 on 5 and 238 DF,  p-value: < 2.2e-16
```

## d) Calculate AIC and BIC for both models. According to AIC which model is best? Which model does BIC prefer?

For model 1:

```
> anova(model1)
Analysis of Variance Table

Response: body.fat
          Df Sum Sq Mean Sq F value    Pr(>F)
thigh      1 4532.9  4532.9 269.191 < 2.2e-16 ***
weight     1 1006.7  1006.7  59.781 2.873e-13 ***
abdomen    1 5065.1  5065.1 300.794 < 2.2e-16 ***
Residuals 241 4058.2    16.8
---
Signif. codes:  0 `***' 0.001 `**' 0.01 `*' 0.05 `.' 0.1 ` ' 1
> n <- nrow(fat)
> p1 <- 3  #there are three predictors
> aic1 <- n*log(2048.2/n) + 2*p1
> bic1 <- n*log(2048.2/n) + p1*log(n)
> aic1
[1] 526.2473
> bic1
[1] 536.7511
```

For model 2:

```
> anova(model2)
Analysis of Variance Table

Response: body.fat
            Df Sum Sq Mean Sq  F value    Pr(>F)
weight       1 5506.3  5506.3 330.8926 < 2.2e-16 ***
age          1 1326.5  1326.5  79.7155 < 2.2e-16 ***
I(age^2)     1    0.1     0.1   0.0067    0.9348
height       1 1282.1  1282.1  77.0444 3.251e-16 ***
abminuswrist 1 2570.7  2570.7 154.4785 < 2.2e-16 ***
Residuals  239 3977.2    16.6
---
Signif. codes:  0 `***' 0.001 `**' 0.01 `*' 0.05 `.' 0.1 ` ' 1
> aic2 <- n*log(3977.2/n) + 2*p2
> bic2 <- n*log(3977.2/n) + p2*log(n)
> aic2
```

```
[1] 692.8334
> bic2
[1] 710.3397
```

And for model2.a (the one with observation 37 removed):

```
> bic2.a <- n*log(3792.4/n) + p2*log(n)
> bic2.a
[1] 698.6828
> aic2.a <- n*log(3792.4/n) + 2*p2
> aic2.a
[1] 681.1766
```

Both AIC and BIC prefer the first model.

```
n <- nrow(fat)
rand <- sample(1:n)%%3 + 1
group1.index <- (1:n)[rand==1]
group2.index <- (1:n)[rand==2]
group3.index <- (1:n)[rand==3]
fat.1 <- fat[group1.index,]
fat.2 <- fat[group2.index,]
fat.3 <- fat[group3.index,]
```

ii) Fit a model using weight, age, age^2, height, and abdomen minus wrist using just the first two groups. (Don't worry about transforming the data.) You'll need to combine the groups together: training <- fat[c(group1.index, group2.index),]

iii) Find the predicted values using group 3. Calculate the residual sum of squares (use the "residual" function). Calcualte the mean square error (RSS/n) where n is the sample size of the training group.

iv) Repeat steps (ii) and (iii) using Group3 as the test data and then with Group2 as the test data. What are the values for the mean square error? What are the value sfor the RSS (residual sum of squares)?

v) You can estimate the over-all mean square error by adding the residual sum of squares for each test set, and dividing by the total sample size. What is this estimate?

In principle, you could then repeat the procedure for Model 1 and use this to help you decide which makes the better predictions. (No need to do that here.)

In practice, using k=3 gives a crude (typically too large) measure of the error. You can improve by taking k larger. In fact, one method (requires that you write a program) uses k = n-1. This is also called the "leave one out" method or the "jacknife".

You can do all of this with the cv command:
> cv.lm(df=fat,form.lm = model2$call,m=3)

The Sums of squares and mean squared error are, for each fold (a "fold" is the third of the data randomly set aside to serve as the test data):
fold 1: SS = 2773  MS = 34.2
fold2: SS = 3016, MS = 36.8
fold3 : SS = 3518, MS = 42.9

Overall MS = 38

## B) Bootstrapping

When the residuals are not normal, then our estimates of confidence intervals are only approximations. While these approximations improve as the sample size increases, how do we know when the sample size is good enough?

An alternative method is to use bootstrapping. Bootstrapping can be used to provide confidence intervals for the slope (and intercept, but we'll focus on the slope) that do not require an assumption of normal distribution. In this exercise, we'll calculate a bootstrap confidence interval for the slope in a simple linear regression.

a) Download the movie data. Fit a model using the log of Friday gross to predict the log of the ultimate gross. What is the model?

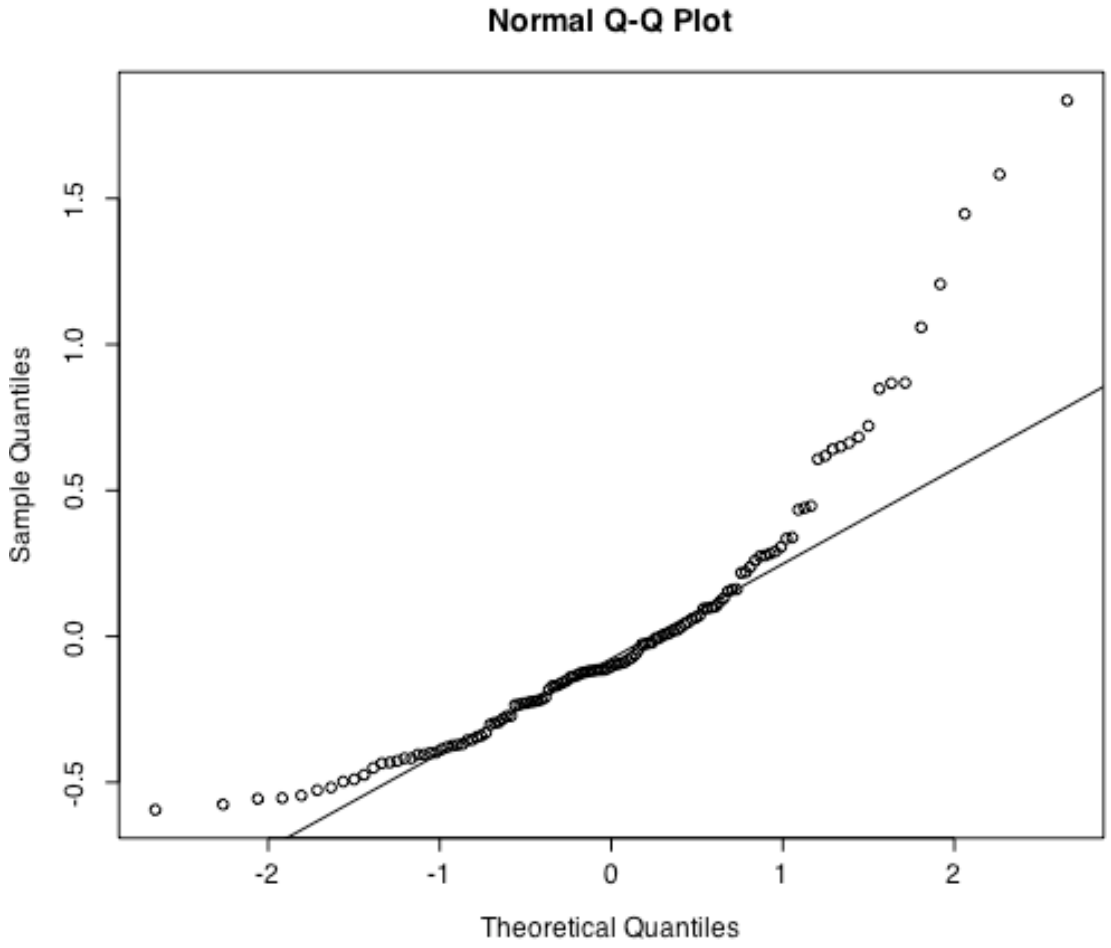
Log(gross) = 1.52 + 1.06*log(Friday)

b) What is a 95% confidence interval for the slope in your model? Does it include 0?
A 95% CI is (.978, 1.14)

```
lm(formula = lgross ~ lfriday)

Residuals:
   Min     1Q Median     3Q    Max
-0.594 -0.295 -0.103  0.142  1.836

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)   1.5248     0.6112    2.49    0.014 *
lfriday       1.0588     0.0414   25.59   <2e-16 ***
---
Signif. codes:  0 `***' 0.001 `**' 0.01 `*' 0.05 `.' 0.1 ` ' 1

Residual standard error: 0.444 on 125 degrees of freedom
Multiple R-Squared: 0.84,       Adjusted R-squared: 0.838
F-statistic:  655 on 1 and 125 DF,  p-value: <2e-16

> me <- 1.96*.0414
> 1.0588-me
[1] 0.978
> 1.0588+me
[1] 1.14
```

## c) Are the residuals normally distributed? Explain.

Far from it:




d) The first step is to take "bootstrap sample 1" from the original data. To do this, take a random sample WITH replacement from the original data:

```
original.data <- data.frame(logfriday, loggross)  #creates a data frame with just the two variables of interest
n <- nrow(original.data) #saves the number of observations
sample.index <- sample(1:n, n, replace=T)  # randomly selects n integers between 1 and n, with replacement
bs1 <- original.data[sample.index,]   #creates a subset of the original data, randomly chosen
```

"Document" this code -- explain what each line does.

e) Now fit the model using bs1 (don't do transformations or diagnostics.) What is the slope? save this slope in a variable named slope.1

```
> original.data <- data.frame(lfriday, lgross)
> n <- nrow(original.data)
> sample.index <- sample(1:n, n, replace=T)
> bs1 <- original.data[sample.index,]
> fit.bs1 <- lm(bs1$lgross~bs1$lfriday)
> coef(fit.bs1)[2]
bs1$lfriday
      1.05
> slope.l <- coef(fit.bs1)[2]
```

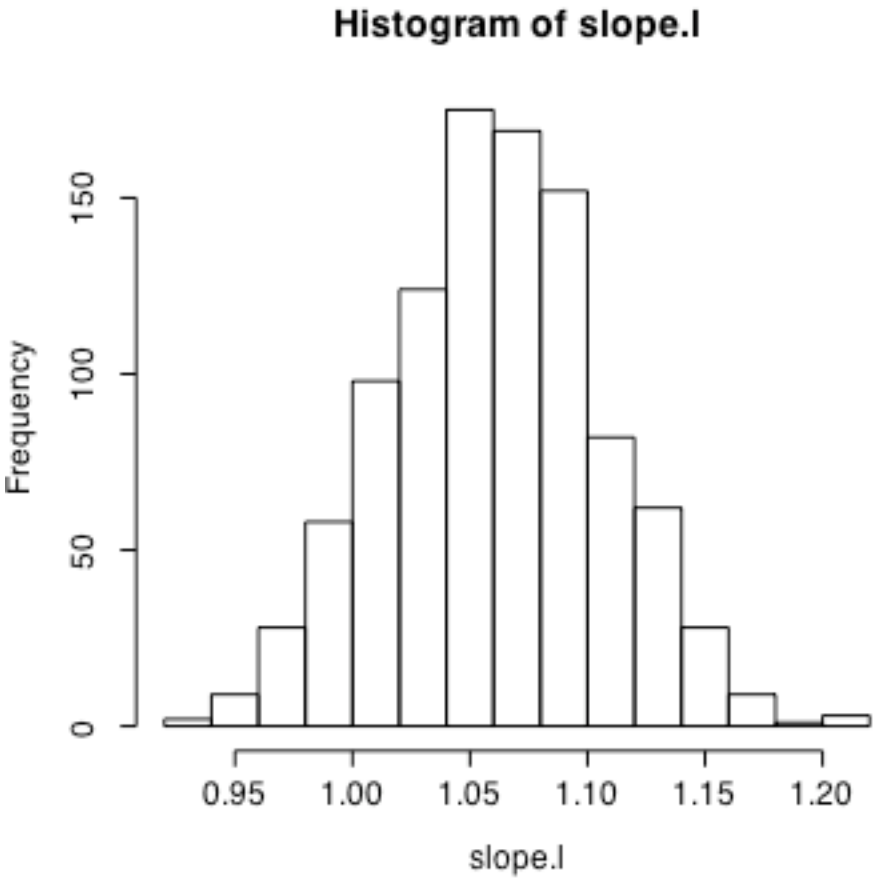
f) Repeat steps (d) and (e). Call this slope slope.2. You now have two different estimates of the slope

```
> sample.index <- sample(1:n, n, replace=T)
> bs2 <- original.data[sample.index,]
> fit.bs2 <- lm(bs2$lgross~bs2$lfriday)
> coef(fit.bs2)[2]
bs2$lfriday
      1.12
> slope.l <- c(slope.l, coef(fit.bs2)[2])
```

g) Repeat (d) and (e)998 times. You'll now have 1000 estimates of the slope and can use this to understand the variability in the estimate. I don't expect you to actually repeat this 1000 times of course. Instead, write a function that will do this. The input to the function should be the data frame (what we called original.data) and a number called B which will represent the number of repetitions. (In this example, B = 1000). The output should be a vector of B estimates of the slope.

Here's one way of doing it that's a little different than the directions ask for. This is not a function, but serves the same purpose.

```
> for (i in 1:998){
+ sample.index <- sample(1:n,n,replace=T)
+ bs <- original.data[sample.index,]
+ fit <- lm(bs$lgross~bs$lfriday)
+ slope.l <- c(slope.l, coef(fit)[2])}
> hist(slope.l)
```




Note that the sampling distribution of the slope is approximately normal, as the Central Limit Theorem predicts.

The function below assumes that data is a data frame that contains variables named lfriday and lgross.

```
bs.fun <- function(data, B){
slopes <- c()
n <- nrow(data)
for (i in 1:B){
        sample.index <- sample(1:n, n, replace=T)
        bs <- data[sample.index, ]
        fit <- lm(bs$lgross~bs$lfriday)
        slopes <- c(slopes, coef(fit)[2])}
slopes}
```

h) One crude way of getting a pretty good 95% confidence interval is to take as your left end point the .025 quartile of the output from your function in (g). The right end point is the .975 quantile. So let's say that you have a vector called *boostraps* that consists of 1000 estimates of the slope. Your estimate of the 95% confidence interval could be found using quantile(boostraps, c(.025, .975)).

What is this interval? How does it compare to the interval in (b)?

```
> quantile(slope.l, c(.025, .975))
 2.5% 97.5%
0.971 1.147
```

(.971, 1.147) is pretty close to what we got assuming normality!