STAT230 HW 3
University of California, Berkeley

Thibault Doutre, Student ID 26980469

February 11, 2016

1

Here is the code to generate the data, using the bivariate normal relationship $Y = \rho X + \sqrt{1-\rho^2}Z$, for Z being standard normal.

```
generate_data = function(n = 100){
  # Define parameters
  rho = 0.7
  mu1=180; s1=40; mu2=66; s2=3

  # Define X, Y and Z with the bivariate normal relationship
  X = rnorm(n)
  Z = rnorm(n)
  eps = sqrt(1-rho^2) * Z
  Y = rho * X + eps

  # Adjust means and variances
  Y = (Y-mean(Y))/sd(Y)*s2+mu2
  X = (X-mean(X))/sd(X)*s1+mu1

  # Adjust rho by transforming Y
  rho_hat = cor(X,Y)
  a = s1^4*(rho^2-1)
  b = 2*rho_hat*s1^3*s2*(rho^2-1)
  c = (rho^2-rho_hat^2)*s2^2*s1^2
  delta = b^2-4*a*c
  correction = (-b-sqrt(delta))/(2*a)
```

```
  Y=Y+correction*X

  # Adjust means and variances
  Y = (Y-mean(Y))/sd(Y)*s2+mu2
  X = (X-mean(X))/sd(X)*s1+mu1

  # Put into data frame
  df = data.frame(WT = Y,
                  HT = X,
                  BMI = 703 * Y / X^2)

  # Output
  return(list(df=df,rho=rho,eps=eps))
}
data = generate_data()

df = data$df
M=df[,1:2]
rho = data$rho
eps = data$eps
```

In order to adjust the correlation of the random variables X and Y, I defined:

$$\hat{\rho} = \frac{cov(X,Y)}{\sigma_X \sigma_Y} \tag{1}$$

$$\tag{2}$$

And a cosntant *correction* such that:

$$\rho = \frac{cov(X, Y + correction * X)}{\sigma_X \sigma_Y} \tag{3}$$

I find the correction coefficient by finding the negative solution of the second order equation:

$$\rho = \frac{cov(X, Y + correction * X)}{\sigma_X \sigma_Y} \tag{4}$$

$$\rho = \frac{cov(X,Y) + correction * \sigma_X^2}{\sigma_X \sqrt{correction^2 \sigma_X^2 + 2 correction * cov(X,Y) + \sigma_Y^2}} \tag{5}$$

Which is equivalent to:

$$(\rho^2 - 1)\sigma_X^4 * correction^2 + 2\hat{\rho}\sigma_X^3 * \sigma_Y^3 * (\rho^2 - 1) * correction + (\rho^2 - \hat{\rho}^2) * \sigma_Y^2 * \sigma_X^2 = 0 \quad (6)$$

Then, we can easily solve this equation and find the corresponding *correction* to adjust the $\hat{\rho}$ to be ρ.

We can see the result by displaying the correlation matrix.

```
# Correlation and covariance matrices
cor(M)

##     WT  HT
## WT 1.0 0.7
## HT 0.7 1.0

cov(M)

##    WT   HT
## WT  9   84
## HT 84 1600

# Mean of variables
mean(df$WT)

## [1] 66

mean(df$HT)

## [1] 180
```

2

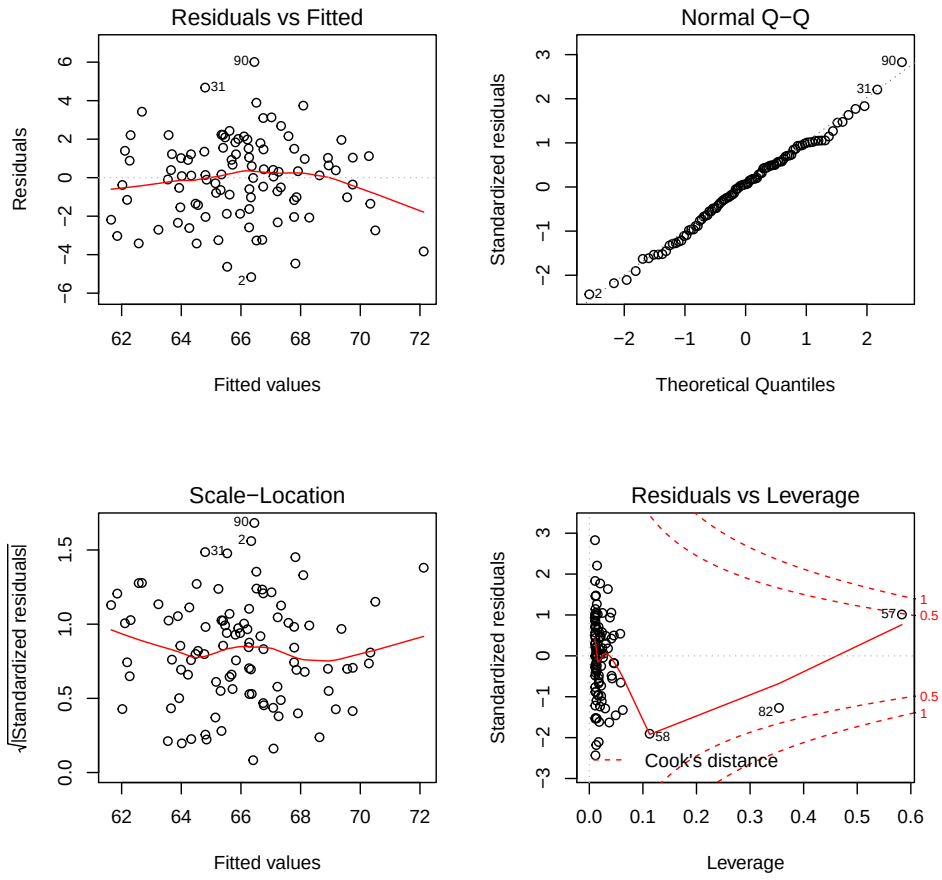
```
lm.fit = lm(WT ~ ., data = df)
beta = lm.fit$coefficients
```

False assumptions:

First X and Y are random variables, not observed values. Here we know β and do observe ϵ. The model is not linear itself, i.e. BMI is not a linear

function of the columns of X. The resuiduals are not gaussian. We can see it by plotting the object $lm.fit$.

```
par(mfrow=c(2,2))
plot(lm.fit)
```




```
par(mfrow=c(1,1))
```

In particular, the sd of the residuals is not equal to 1:

```
sd(lm.fit$residuals)

## [1] 2.11417
```

3

The true value of β_1, the coefficient associated with the height HT, is $\rho * \frac{sd(WT)}{sd(HT)}$.

```
# True value of beta
beta_true = rho*sd(df$WT)/sd(df$HT)
beta_true

## [1] 0.0525

# The simulated value of £\beta_1£ is
beta1 = beta["HT"]
beta1

##         HT
## 0.06237115
```

4

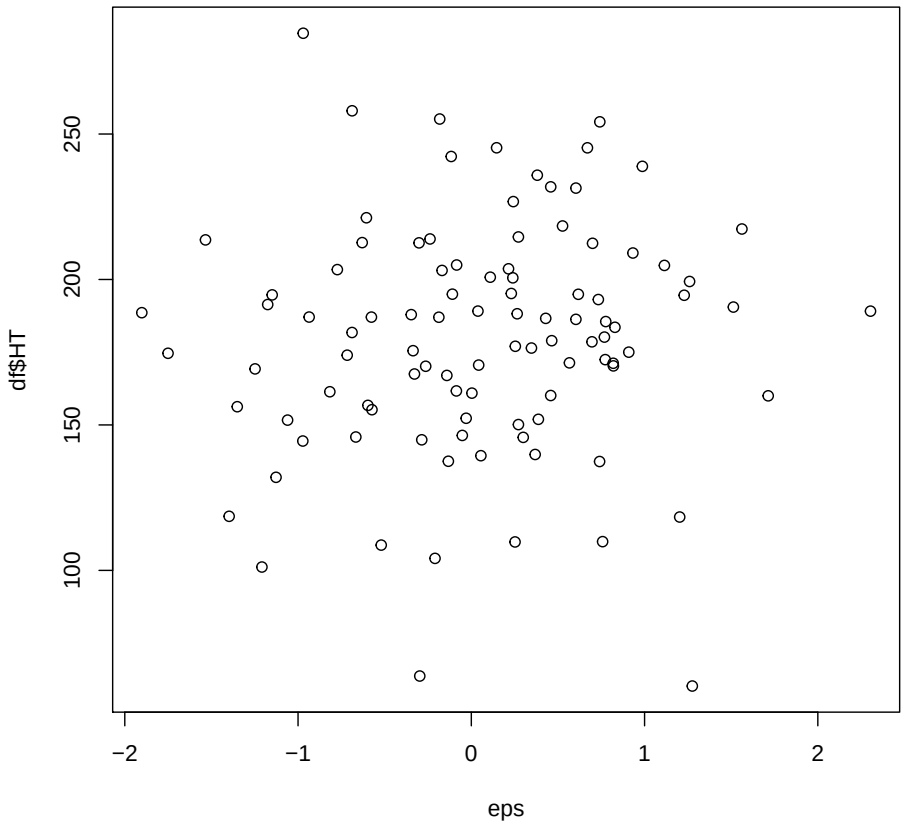
According to Theorem 2, page 43 of Freedman, OLS is conditionally unbiased, that is, $E(\hat{\beta}|X) = \beta$. Therefore, we should have $E(E(\hat{\beta}|X)) = \beta$ i.e. $E(\hat{\beta}) = \beta$. However here some assumptions for the OLS are not true. Therefore, the estimator could be biased.

5

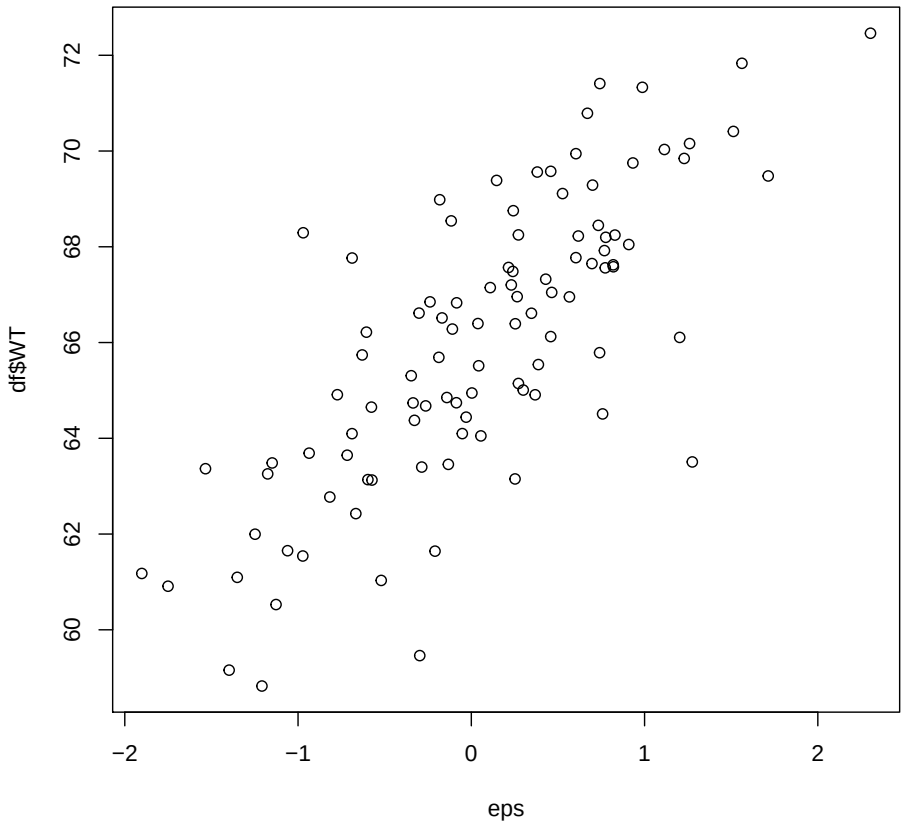
```
WT_hat = predict(lm.fit)
e = df$WT-WT_hat
```

To see if the variables are correlated, wee look for some pattern in the plot of one against the other.

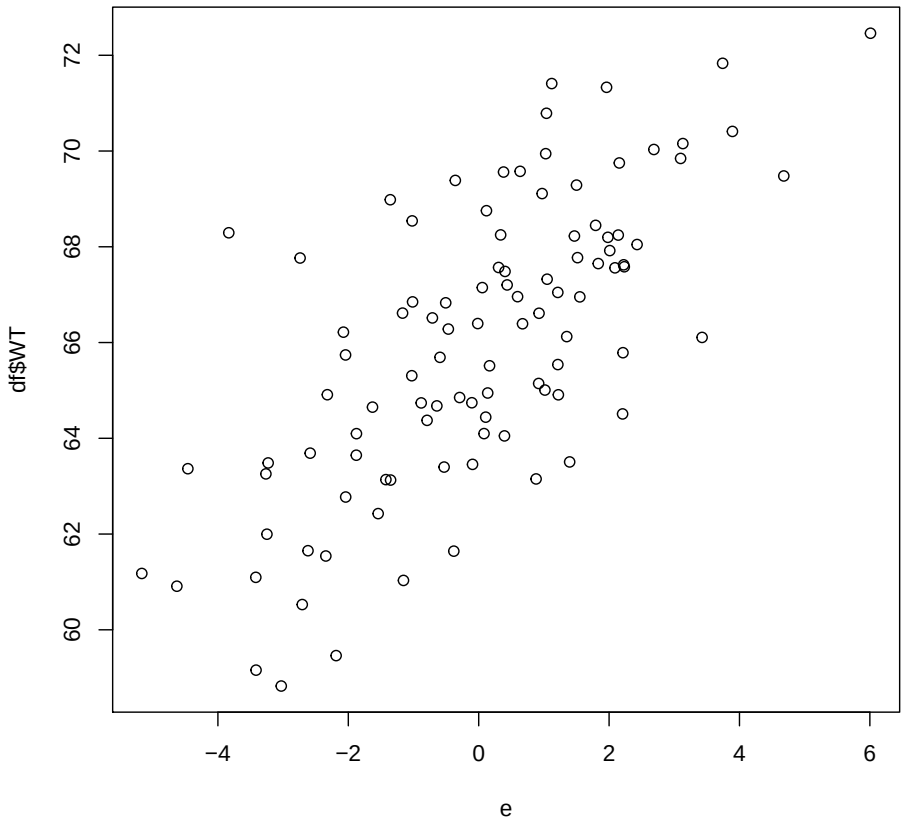
```
plot(eps,df$HT) # not correlated, seem to be independent
```



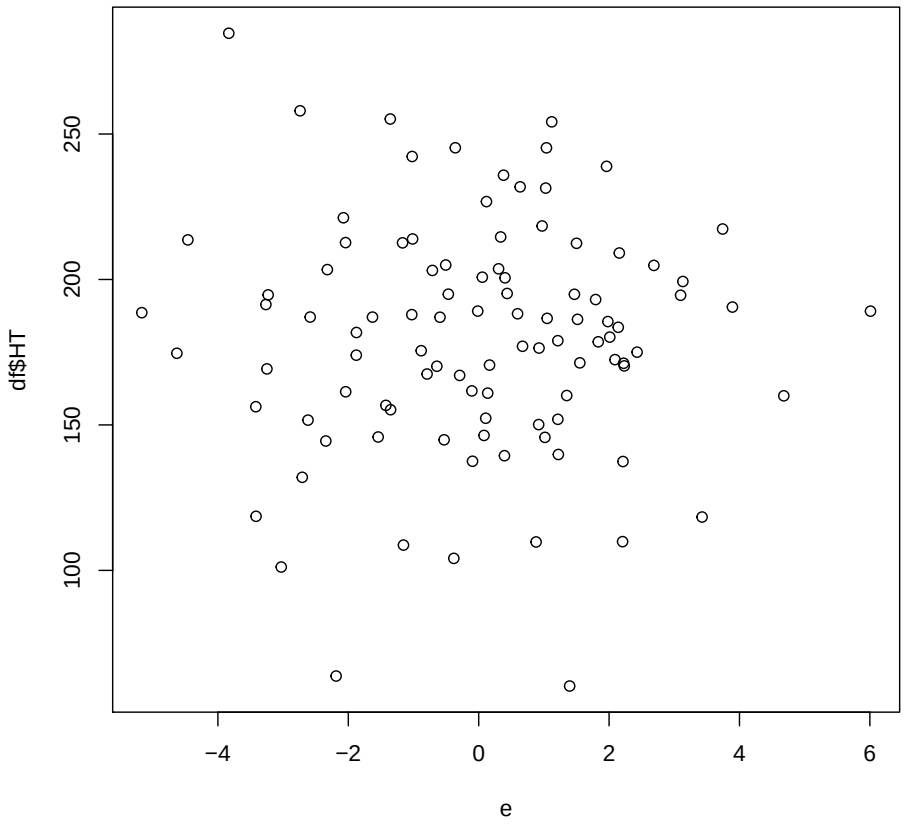

```
plot(eps,df$WT) # positively correlated -> dependent
```




```
plot(e,df$WT) # positively correlated -> dependent
```




```
plot(e,df$HT) # not correlated, seem to be independent
```



To see if two vectors are orthogonal, I compute their scalar product.

```
sum(e*df$WT) # not orthogonal
## [1] 442.5016
sum(e*df$HT) # orthogonal
## [1] 2.131237e-10
sum(eps*df$WT) # not orthogonal
## [1] 542.8491
sum(eps*df$HT) # not orthogonal
## [1] 1239.723
```

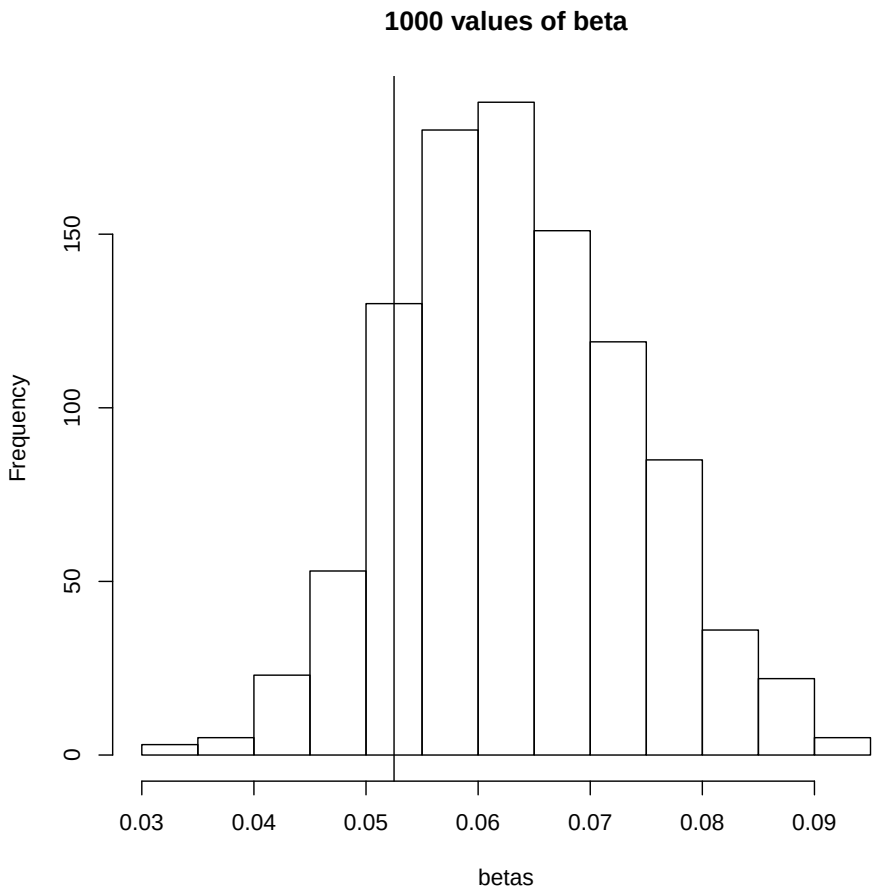
6

```r
# Generate beta
beta = function(n=100){
  data0 = generate_data(n)
  df0 = data0$df
  lm.fit0 = lm(WT ~ HT+BMI, data = df0)
  lm.fit0$coefficients
}

# Replication
betas = replicate(1000,beta()["HT"])
```

7

```r
hist(betas,main = "1000 values of beta")
abline(v = beta_true)
```



The estimate looks biased here. We can have an estimate of the bias with:

```
mean(betas) - beta_true

## [1] 0.011039
```