# W203 Supplementary Exercise 4

*Mohammad Jawad Habib*

*April 20, 2016*

## Code copied from assignment

```
# getwd()
# setwd("W203 Week 14")
library(MASS)
```

```
## Warning: package 'MASS' was built under R version 3.2.4
```

```
N = 100
#creating independent variables
#first, the covariance matrix for our multi-variate normal
# this determines the correlation of our independent variables.
sig = matrix(c(2,.5,.25,.5,1,0,.25,0,1) , nrow=3)
# now the variables:
M = mvrnorm(n = N, mu = rep(1,3), Sigma = sig )
y.cont = 1 + 2* M[,1] - 5 * M[,2] + M[,3] + rnorm(N)
y.bin = as.numeric ( y.cont > 0 )
# include the intercept in your independent variables
X = cbind (1, M )
y = y.cont
# or y = y.bin , depending on the exercise
```

## Part 1. OLS using a recursive function

```
# Somehting is wrong with this function
# It returns Null for column length after first iteration
reg <- function(y, x) {
  print(ncol(x))
  if (ncol(x) == 1) {
    return(sum(x*y) / sum(x^2))
  } else {
      # x = ifelse(ncol(x) > 1, x[, -1], x[, 1])
      return(reg(y, ifelse(ncol(x) > 0, x[, -1], x[, 1])))
  }
}

# does not work

# xdf <- data.frame(X)
#
# reg(y.cont, xdf)
# reg(y.cont, X[, 2:4])
```

## Part 2. OLS using numeric optimization

```r
# Generate heteroskedastic data
x1 <- runif(100, -1, 1)# rep(c(-1, 1), 50)
x2 <- rnorm(100)
err <- rnorm(100, sd = 1:10)
y1 <- 1 + x1 + x2 + err # not sure if this is a better method

library(lmtest)
```

```
## Loading required package: zoo
```

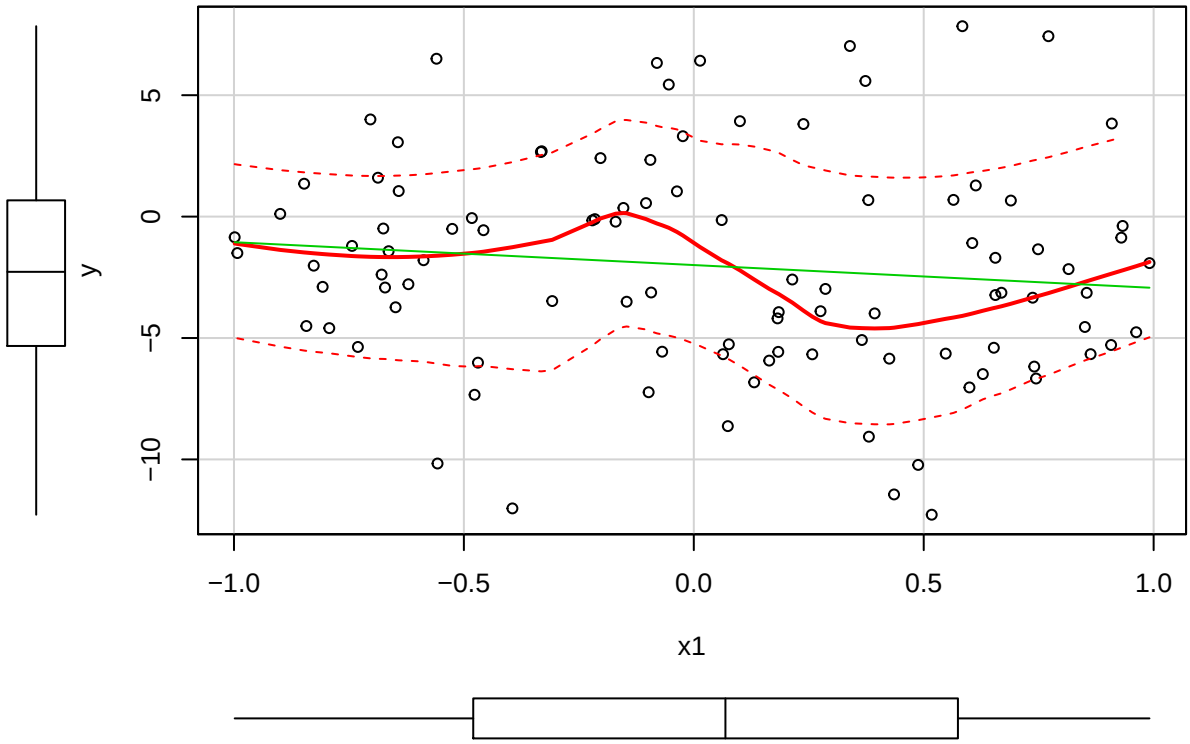
```
##
## Attaching package: 'zoo'
```

```
## The following objects are masked from 'package:base':
##
##     as.Date, as.Date.numeric
```

```r
bptest(y1 ~ x1) # reject homoskedasticity (p > 0.05)
```

```
##
##  studentized Breusch-Pagan test
##
## data:  y1 ~ x1
## BP = 0.85683, df = 1, p-value = 0.3546
```

```r
library(car)
scatterplot(x1, y)
```




```
# create a data.frame for analysis
d <- data.frame(y1, x1, x2)

# use bootstrap to estimate the model
library(boot)
```

```
##
## Attaching package: 'boot'

## The following object is masked from 'package:car':
##
##     logit
```

```
# copy bootReg from DSUR book
bootReg <- function (formula, data, i)
{
  d <- data [i,]
  fit <- lm(formula, data = d)
  return(coef(fit))
}

boot.r <- boot(statistic = bootReg, formula = y ~ x1 + x2, data = d, R = 2000)
boot.r
```

```
##
```

```
## ORDINARY NONPARAMETRIC BOOTSTRAP
##
##
## Call:
## boot(data = d, statistic = bootReg, R = 2000, formula = y ~ x1 +
##     x2)
##
##
## Bootstrap Statistics :
##       original      bias    std. error
## t1* -2.0057166 -0.02763536  0.07809893
## t2* -0.9090379  0.89048335  0.77740519
## t3*  0.1614231 -0.15500675  0.47584571
```

```
summary(boot.r)
```

```
##      R original  bootBias   bootSE   bootMed
## 1 2000 -2.00572 -0.027635 0.078099 -2.035964
## 2 2000 -0.90904  0.890483 0.777405 -0.025160
## 3 2000  0.16142 -0.155007 0.475846  0.016295
```

```
boot.ci(boot.r, type = "bca", index = 1)
```

```
## BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
## Based on 2000 bootstrap replicates
##
## CALL :
## boot.ci(boot.out = boot.r, type = "bca", index = 1)
##
## Intervals :
## Level       BCa
## 95%   (-2.092, -1.713 )
## Calculations and Intervals on Original Scale
## Some BCa intervals may be unstable
```

```
boot.ci(boot.r, type = "bca", index = 2)
```

```
## BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
## Based on 2000 bootstrap replicates
##
## CALL :
## boot.ci(boot.out = boot.r, type = "bca", index = 2)
##
## Intervals :
## Level       BCa
## 95%   (-2.5029, -0.2384 )
## Calculations and Intervals on Original Scale
## Warning : BCa Intervals used Extreme Quantiles
## Some BCa intervals may be unstable
```

```r
boot.ci(boot.r, type = "bca", index = 3)
```

```
## BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
## Based on 2000 bootstrap replicates
##
## CALL :
## boot.ci(boot.out = boot.r, type = "bca", index = 3)
##
## Intervals :
## Level       BCa
## 95%   (-0.6141,  1.1884 )
## Calculations and Intervals on Original Scale
## Some BCa intervals may be unstable
```

```r
# linear model without bootstrap
mod <- lm(y ~ x1 + x2, data = d)
summary(mod)
```

```
##
## Call:
## lm(formula = y ~ x1 + x2, data = d)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -10.4336  -3.0837  -0.3767   2.6214  10.3272
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  -2.0057     0.4436  -4.521 1.74e-05 ***
## x1           -0.9090     0.7833  -1.161    0.249
## x2            0.1614     0.4656   0.347    0.730
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 4.412 on 97 degrees of freedom
## Multiple R-squared:  0.01607,    Adjusted R-squared:  -0.004218
## F-statistic: 0.7921 on 2 and 97 DF,  p-value: 0.4558
```

```r
confint(mod)
```

```
##                  2.5 %     97.5 %
## (Intercept) -2.8861748 -1.1252583
## x1          -2.4636127  0.6455368
## x2          -0.7626366  1.0854829
```

```r
library(sandwich)
library(lmtest)

coefs.sand <- coeftest(mod, vcov = sandwich)

# results of lm and boot.r look similar
cbind("boot" = boot.r$t0, "lm" = coefs.sand[, 1])
```

```
##                   boot         lm
## (Intercept) -2.0057166 -2.0057166
## x1          -0.9090379 -0.9090379
## x2           0.1614231  0.1614231
```

## Part 3. Logit

```
# create the logit function based on equation 3 from the link below
# http://faculty.smu.edu/tfomby/eco6352/Notes/Logit%20and%20Probit%20Notes.pdf
logit.mine <- function(y, X, b) {
  return(
    -sum(y*log(1 / (1 + exp(-X %*% b)))
         + (1 - y)*log(1 - 1 / (1 + exp(-X %*% b))))
    )
}

# initialize the values of our coefficients (b)
b.new <- c(0, 0, 0, 0) # one for each independent variable in X
model.logit <- optim(b.new, logit.mine, X = X, y = y.bin,
                     method = "BFGS", hessian = TRUE)

X.new <- cbind(y.bin, X)
model.glm <- glm(X.new[, 1] ~ X.new[, 3] + X.new[, 4] + X.new[, 5],
                 data = data.frame(X.new), family = binomial, x = TRUE)

# compare the results
cbind("logit.coef" = model.logit$par, "glm.coef" = model.glm$coefficients)
```

```
##              logit.coef    glm.coef
## (Intercept)   0.8629345   0.8629099
## X.new[, 3]    3.1023325   3.1023504
## X.new[, 4]  -10.0209954 -10.0210360
## X.new[, 5]    2.8579124   2.8579347
```