Replication of Examples in Chapter 6

Zheng Tian

1 Introduction

This document is to show how to perform hypothesis testing for a single coefficient in a simple linear regression model. I replicate examples that occur in Chapter 6.

2 Scatterplot with two regressors

```
library(AER)
library(foreign)
classdata <- read.dta("caschool.dta")
```

We can draw the scatterplots of *STR* against *TestScr* and *PctEl* against *TestScr*, and arrange the two scatterplot in one frame.

```
# scatterplot
oldpar <- par(mfrow = c(2, 1))

plot(classdata$str, classdata$testscr, col = "red",
    main = "student-teacher ratio vs test scores",
     xlab = "Student-teacher ratio", ylab = "Test scores")

plot(classdata$el_pct, classdata$testscr, col = "blue",
     main = "English learners vs test scores",
     xlab = "Percentage of English learners",
     ylab = "Test scores")

par(oldpar)
```

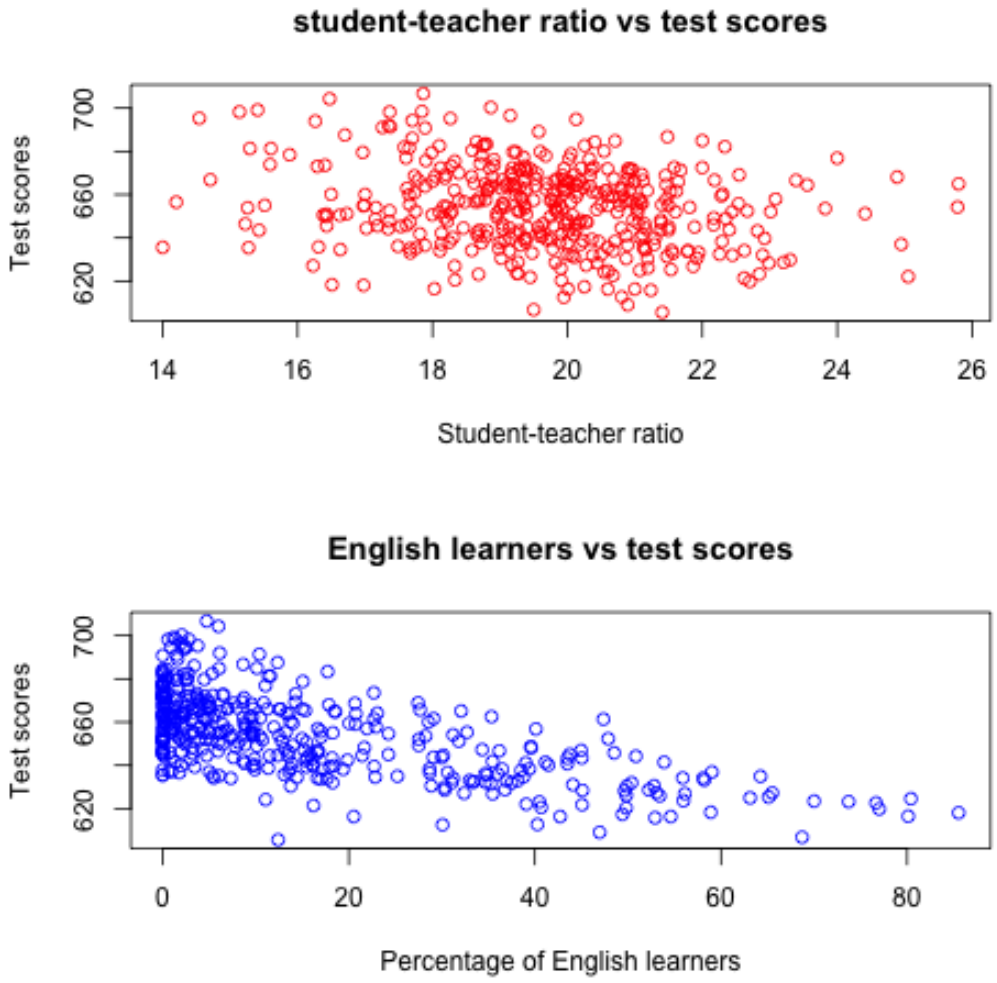

Figure 1: The scatterplots of test scores against student-teacher ratios and the percentage of English learners

3 The OLS estimation of the multiple regression model

The multiple regression model is

$$TestScore_i = \beta_0 + \beta_1 STR_i + \beta_2 PctEL + u_i \tag{1}$$

Then we define the `formula` object for the multiple regression model and estimate it

```
mod1 <- lm(testscr ~ str + el_pct, data = classdata)
(sum.mod1 <- summary(mod1))
```

```
Call:
lm(formula = testscr ~ str + el_pct, data = classdata)

Residuals:
    Min      1Q  Median      3Q     Max
-48.845 -10.240  -0.308   9.815  43.461

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept) 686.03225    7.41131  92.566  < 2e-16 ***
str          -1.10130    0.38028  -2.896  0.00398 **
el_pct       -0.64978    0.03934 -16.516  < 2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 14.46 on 417 degrees of freedom
Multiple R-squared:  0.4264,Adjusted R-squared:  0.4237
F-statistic:   155 on 2 and 417 DF,  p-value: < 2.2e-16
```

We can get the estimated coefficients, predicted values, residuals, SER, R^2, and the adjusted R^2 using the following commands.

```
# get the compoenents
b <- coef(mod1) # coefficients
y.hat <- predict(mod1) # predicted value of y
u.hat <- resid(mod1) # residuals
SER <- sum.mod1$sigma # standard error of regression
R2 <- sum.mod1$r.squared # R squared
```

```
AR2 <- sum.mod1$adj.r.squared # adjusted R squared
```

So the coefficient on *STR* is `-1.101`, which means that, holding *PctEL* constant, one unit increase in *STR* will lead to a decrease in *TestScr* by `-1.101` units. The R^2 and the adjusted R^2 are round `0.426` and `0.424`, respectively.

The homoskedasticity-only covariance matrix and the heteroskedasticity-consistent covariance matrix of the coefficients can be computed by the command below

```
(vcov.hm <- vcov(mod1)) # homoskedasticity-only covariance matrix
se.hm <- sqrt(diag(vcov.hm)) # homoskedasticity-only standard error
```

```
             (Intercept)         str       el_pct
(Intercept) 54.92755274 -2.79596671  0.030730824
str         -2.79596671  0.14461160 -0.002807340
el_pct       0.03073082 -0.00280734  0.001547836
```

```
(vcov.ht <- vcovHC(mod1, type = "HC1")) # HCCM
se.ht <- sqrt(diag(vcov.ht)) # heterskedasticity-robust se
```

```
             (Intercept)           str        el_pct
(Intercept) 76.18189018 -3.7569802107 -0.0134448546
str         -3.75698021  0.1873566583 -0.0003131024
el_pct      -0.01344485 -0.0003131024  0.0009629703
```

4 An illustration of $TSS = ESS + SSR$

Let's verify the property of the OLS estimator, $TSS = ESS + SSR$. We can compute the three quantities using the following commands.

```
TSS <- with(classdata, sum((testscr - mean(testscr))^2))
ESS <- sum((y.hat - mean(y.hat))^2)
SSR <- sum(u.hat^2)
```

When we directly verify the equality, what we get is `FALSE`.

```
TSS == ESS + SSR
```

```
[1] FALSE
```

This is due to the error of computation with floating point numbers. So instead of directly compare the LHS with the RHS, we can do the following,

```
abs(TSS - ESS - SSR) < 1.0e-9
```

```
[1] TRUE
```

5 An illustration of the FWL theorem

Now let's demonstrate the FWL theorem. Suppose we are interested in the effect of *STR* on *TestScr* controlling for *PctEl*. So according to the FWL theorem, we can follow three steps to estimate the coefficient on *STR*

Step 1 Regress *STR* on *PctEL* and get the residuals;

Step 2 Regress *TestScr* on *PctEl* and get the residuals;

Step 3 Regress the residuals in the second step on the residuals in the first step to get the estimated coefficient.

These steps can be implemented by the following command

```
# step 1
m1 <- lm(str ~ el_pct, data = classdata)
# step 2
m2 <- lm(testscr ~ el_pct, data = classdata)
# step 3
m3 <- lm(resid(m2) ~ resid(m1) - 1)
```

Finally, we compare the estimated coefficient on *STR* following the steps above and that estimated using both *STR* and *PctEl* at a time.

```
abs(coef(m3) - b[2]) < 1.0e-10
```

```
resid(m1)
     TRUE
```

6 An illustration of the dummy variable trap

We define dummy variables for small class, medium class, large class, according to *STR*

$$Small = \begin{cases} 1, & \text{if } STR < 18 \\ 0, & \text{otherwise} \end{cases}, Medium = \begin{cases} 1, & \text{if } 18 \leq STR < 20 \\ 0, & \text{otherwise} \end{cases}, Large = \begin{cases} 1, & \text{if } STR \geq 20 \\ 0, & \text{otherwise} \end{cases}$$

Defining these three dummy variables can be accomplished by the following commands

```
small <- ifelse(classdata$str < 18, 1, 0)
middle <- ifelse(classdata$str >= 18 & classdata$str < 20, 1, 0)
large <- ifelse(classdata$str >= 20, 1, 0)
```

from which we get three vectors consisting of 1 and 0.

We can more easily define dummy variables in R using a `factor` object as follows

```
classsize <- ifelse(classdata$str < 18, "small",
     ifelse(classdata$str >= 18 & classdata$str < 20, "medium", "large"))
classsize <- as.factor(classsize)
```

Let's first try to estimate a model with an intercept and all three dummy variables, which is an example of the dummy variable trap.

```
mod3 <- lm(testscr ~ small + middle + large, data = classdata)
summary(mod3)
```

```
Call:
lm(formula = testscr ~ small + middle + large, data = classdata)

Residuals:
    Min      1Q  Median      3Q     Max
-48.441 -14.354   0.534  13.749  45.109

Coefficients: (1 not defined because of singularities)
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  649.979      1.378 471.721  < 2e-16 ***
small         12.067      2.551   4.731 3.06e-06 ***
middle         5.212      2.005   2.600  0.00965 **
large             NA         NA      NA       NA
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 18.59 on 417 degrees of freedom
Multiple R-squared:  0.05272,Adjusted R-squared:  0.04818
F-statistic:  11.6 on 2 and 417 DF,  p-value: 1.247e-05
```

We can see that R automatically drop the dummy variable for large classes in estimation, resulting in `NA` for `large` and a warning message saying that `Coefficients: (1 not defined`

because of singularities). So we should drop a dummy variable to set up a correct model.

```
mod3.a <- lm(testscr ~ small + middle, data = classdata)
summary(mod3.a)

Call:
lm(formula = testscr ~ small + middle, data = classdata)

Residuals:
    Min      1Q  Median      3Q     Max
-48.441 -14.354   0.534  13.749  45.109

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  649.979      1.378 471.721  < 2e-16 ***
small         12.067      2.551   4.731 3.06e-06 ***
middle         5.212      2.005   2.600  0.00965 **
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 18.59 on 417 degrees of freedom
Multiple R-squared:  0.05272,Adjusted R-squared:  0.04818
F-statistic:  11.6 on 2 and 417 DF,  p-value: 1.247e-05
```

Equivalently, we can drop the intercept term.

```
mod4 <- lm(testscr ~ small + middle + large - 1, data = classdata)
summary(mod4)

Call:
lm(formula = testscr ~ small + middle + large - 1, data = classdata)

Residuals:
    Min      1Q  Median      3Q     Max
-48.441 -14.354   0.534  13.749  45.109

Coefficients:
       Estimate Std. Error t value Pr(>|t|)
small   662.046      2.146   308.4   <2e-16 ***
middle  655.191      1.456   450.0   <2e-16 ***
```

```
large   649.979      1.378   471.7   <2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 18.59 on 417 degrees of freedom
Multiple R-squared:  0.9992,Adjusted R-squared:  0.9992
F-statistic: 1.734e+05 on 3 and 417 DF,  p-value: < 2.2e-16
```

In fact, when we use the `factor` object, `classsize`, the formula get easier as follows,

```
mod5 <- lm(testscr ~ classsize, data = classdata)
summary(mod5)

Call:
lm(formula = testscr ~ classsize, data = classdata)

Residuals:
    Min      1Q  Median      3Q     Max
-48.441 -14.354   0.534  13.749  45.109

Coefficients:
                Estimate Std. Error t value Pr(>|t|)
(Intercept)      649.979      1.378 471.721  < 2e-16 ***
classsizemedium    5.212      2.005   2.600  0.00965 **
classsizesmall    12.067      2.551   4.731 3.06e-06 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 18.59 on 417 degrees of freedom
Multiple R-squared:  0.05272,Adjusted R-squared:  0.04818
F-statistic:  11.6 on 2 and 417 DF,  p-value: 1.247e-05
```

which yields the same estimation as specifying two dummy variables explicitly.