Replication of Examples in Chapter 5

Zheng Tian

April 11, 2016

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 5.

2 The OLS estimation

The linear model is

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

We first read the data, estimate the linear regression model, and get the regression results.

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

df <- classdata[c("testscr", "str")]
mod1 <- lm(testscr ~ str, data = df)
summary(mod1)
```

```
Loading required package: car
Loading required package: lmtest
Loading required package: zoo

Attaching package: ‘zoo’

The following objects are masked from ‘package:base’:
```

```
    as.Date, as.Date.numeric

Loading required package: sandwich
Loading required package: survival

Call:
lm(formula = testscr ~ str, data = df)

Residuals:
    Min      1Q  Median      3Q     Max
-47.727 -14.251   0.483  12.822  48.540

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) 698.9330     9.4675  73.825  < 2e-16 ***
str          -2.2798     0.4798  -4.751 2.78e-06 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 18.58 on 418 degrees of freedom
Multiple R-squared:  0.05124,Adjusted R-squared:  0.04897
F-statistic: 22.58 on 1 and 418 DF,  p-value: 2.783e-06
```

`summary(mod1)` reports the estimated coefficients, their standard errors, t-statistics, and the p-values. It also reports R^2, SER, and other test statistics that we will learn in the next chapters.

By default, the standard errors reported are computed using the formula of **the homoskedasticity-only standard errors**, which are then used in compute the t-statistics. And the p-values are based on the student-t distribution with 418 degrees of freedom.

3 Hypothesis tests

Now we get into testing the zero hypothesis for β_1, that is,

$$H_0 : \beta_1 = \beta_{1,0} \text{ vs. } H_1 : \beta_1 \neq \beta_{1,0}$$

3.1 Get all the quantities used in the test

We use the t-statistic to test such a hypothesis, which has the following formula,

$$t = \frac{\hat{\beta}_1 - \beta_{1,0}}{SE(\hat{\beta}_1)} \tag{2}$$

Upon computing the t-statistic, we compare it with the critical value at the desired significant level, say 5%, which is 1.96 from the standard normal distribution. Also, we can use the t-statistics to get the p-value.

How can we get all the quantities used in this formula? Of course, you can simply copy the values in the output of `summary(mod1)`. But doing so is cumbersome, and very subject to mistakes because of manual operations. More importantly, we cannot have the **heteroskedasticity-robust standard error** of $\hat{\beta}_1$. Fortunately, you can get all the quantities in Equation eq:t-stat-b1 using R functions.

The coefficients

All estimated coefficients can be extracted using the function of `coef()`, which returns a vector containing all estimated coefficients. By default, the first element in the vector is the estimated intercept. So in our regression, the slope is the second element.

```
b <- coef(mod1)
(b1 <- b[2])
```

```
      str
-2.279808
```

The standard errors.

- The homoskedasticity-only standard errors are reported in the output of `summary()` by default in R. They can also be extracted with the function, `vcov()`, which returns a matrix called the covariance matrix, with the diagonal elements representing the variances of the coefficients. Thus, the standard errors are the square roots of the diagonal elements.

  ```
  V <- vcov(mod1)
  (se_b1 <- sqrt(V[2, 2]))
  ```

```
[1] 0.4798256
```

- The heteroskedasticity-robust standard errors are the square roots of the diagonal elements in the **heteroskedasticity-consistent** covariance matrix, obtained using the function of `vcovHC()` in the `sandwich` package that is loaded by default. There are several versions of the heteroskedasticity-consistent covariance matrix. What we use is the type of `HC1`.

```
htV <- vcovHC(mod1, type = "HC1")
(se_b1.rb <- sqrt(htV[2, 2]))
```

```
[1] 0.5194892
```

The t-statistics, the critical value, and the p-value.

The t-statistics using the heteroskedasticity-robust standard errors is then computed by

```
(t_b1.rb <- b1 / se_b1.rb)
```

```
      str
-4.388557
```

Although we know the critical value at the 5% significant level for a two-sided test is 1.96 with a large sample, we prefer getting the value from a function in R. The critical value at the 5% significance level is in fact the 97.5^{th} percentile of the standard normal distribution, which can be got from the `qnorm()` function.

```
(c.5 <- qnorm(0.975))
```

```
[1] 1.959964
```

The p-value associated with the actual t-statistics is $\Pr\left(|t| > |t^{act}|\right) = 2\Phi(-|t^{act}|)$. We can compute the p-value in R, following this definition and using the `pnorm()` function.

```
(pval <- 2 * pnorm(-abs(t_b1.rb)))
```

```
         str
1.141051e-05
```

3.2 Use `coeftest()`

Since hypothesis testing is a very common work in statistics, many R functions have been developed to do it. Here I introduce a function, `coeftest()`, which is in the package of `lmtest`, which is loaded through the `AER` package.

```
coeftest(mod1)
```

```
t test of coefficients:

             Estimate Std. Error t value  Pr(>|t|)
(Intercept) 698.93295    9.46749 73.8245 < 2.2e-16 ***
str          -2.27981    0.47983 -4.7513 2.783e-06 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

By default, it reports the homoskedasticity-only standard errors, the corresponding t-statistics, and the p-values. To get the heteroskedasticity-robust results, we need to add an argument to this function to specify the heteroskedasticity-consistent covariance matrix, which has been defined above as `htV <- vcovHC(mod1, type = "HC1")`.

```
t.tst <- coeftest(mod1, vcov. = htV)
t.b1 <- t.tst["str", "t value"]
```

```
-4.38855724131497
```

3.3 TODO Confidence interval

Finally, we can construct the 95% confidence interval of β_1 using the function of `confint()`.

TODO Replace confidence interval with the heteroskedasticity-robust SE

```
# confidence interval with the default homoskedasticity-only SE
confint(mod1, "str")
```

```
       2.5 %    97.5 %
str -3.22298 -1.336637
```

Since there is no existing function to report the confidence interval with heteroskedasticity-robust SE, we can write a user-defined function to do that.

```
conf_interval_robust <- function(lm_obj, param, vcov_ = vcov(lm_obj),
 level = 0.05){
    ## This function generates a two-sided confidence interval for a
    ## parameter in the linear regression model with a specified
    ## covariance matrix.  The inputs The output

    ## get all the parameters' names and select one based on param
    all_param <- attr(lm_obj$coefficients, "names")
    which_param <- grep(param, all_param)

    ## get the estimated parameter and its standard error
    bhat_param <- coef(lm_obj)[which_param]
    sd_param <- sqrt(vcov_[which_param, which_param])

    ## get the critical value
    cv <- qnorm(1 - level/2)

    ## calculate the confidence interval
    lower <- bhat_param - cv * sd_param
    upper <- bhat_param + cv * sd_param

    conf_interval <- c(lower, upper)
    names(conf_interval) <- c("lower", "upper")
    return(conf_interval)
}
```

4 TODO Dummy variable

A dummy variable can be represented using a `factor` object in R. There are many ways to create a dummy variable. Here I will create a dummy variable

$$D_i = \begin{cases} 1, & \text{if } str < 20 \\ 0, & \text{if } str \geq 20 \end{cases}$$

The R command to create such a dummy variable is as follows

```
D <- factor(ifelse(df$str < 20, 1, 0))
```

The function `ifelse()` creates a vector consisting of 1 and 0. The first argument in this function is a condition, `df$str < 20`. If the condition is satisfied for an element in `df$str`, the corresponding element in `D` is 1, otherwise 0. The function `factor()` converts the `numeric` vector to a `factor` vector.

4.1 TODO Add a scatterplot with the dummy variable

Then we can estimate the linear regression of test scores against the dummy variable, and do the zero hypothesis test.

```
mod2 <- lm(testscr ~ D, data = df)
coeftest(mod2, vcov. = vcovHC(mod2, type = "HC1"))


t test of coefficients:

            Estimate Std. Error  t value  Pr(>|t|)
(Intercept) 649.9788     1.3229 491.3317 < 2.2e-16 ***
D1            7.3724     1.8236   4.0428 6.288e-05 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```