**Lab Assignment 03**

**I. Objectives**: The objectives are to practice constructing multivariate linear models using variances and covariances, to examine omitted variable bias, and to introduce the quantities that go into calculating standard errors.

**II. Datasets**: *caschool.dta*

**III. Packages**: *foreign*, *car*, *Hmisc*

**IV. Preparation**

1) Open RStudio by double-clicking the icon or selecting RStudio from the Windows Start menu.

2) Clear any data in memory by typing rm(list=ls())

3) Download dataset *caschool.dta* and place it in a working directory

4) Download R script “Lab03.R” and place it in a working directory.

5) Open the R script by typing *Ctrl*+*O* or by clicking on File in the upper-left corner, using the dropdown menu, and navigating to the script in your working directory.

6) Run lines 4–6 in the R script to load three packages that you will need. If any of these are not already installed, you need to install them using the **Packages** tab or using the command:

> install.packages("car"); install.packages("Hmisc")

**V. Instructions for Lab 03**

You will use data on school districts in California to explore the factors that determine test scores. The policy question you might consider is whether the influential factors are policies decided by the school districts or demographic factors that are largely outside the control of districts.

Begin by loading the dataset, typing the following line, changing the directory if needed:

> CAdta<-read.dta("C:/Users/Scott/caschool.dta")

If you failed to load the foreign package earlier, then R will not load the Stata dataset. Ensure that the package is both installed and loaded, and then re-run the line of code. Next, you will need to run lines 10–11 of the R script to drop twelve variables from the dataset (by selecting six that we will actually use in the lab. The new data frame that you will work with is named newdata.

You may want to download and view the data description (CaliforniaTestScores.pdf) which lists the variables you will use (on its second page). Some key demographic variables are the average income of families in the district, in thousands of dollars (avginc), the percent of students qualifying for the CalWORKs program, which is the state of California’s name for welfare (calw\_pct), the percent of students qualifying for reduced-price lunch (meal\_pct), and the percent of students who are learning English as a second language (el\_pct). Three policy variables are the number of computers per student (comp\_stu), the district’s expenditures per student (expn\_stu), and the number of students per teacher (str).

You might wish to run the following command in order to see the correlation matrix. The example of movie revenues that we examined in class had the property that IMDB ratings and production budgets were positively correlated to each other, and that both these variables were positively correlated to revenues, but the correlation between production budgets and revenues was quite a bit stronger. Examine the correlations among explanatory variables in this dataset, and then examine their correlations to test scores.

> cor(newdata[,c(2:6)])

**A. Simple Regression Coefficients**

The dependent variable that you will use in your analysis is average test scores in a district (testscr). Run lines 14–15 to find means and variances of test scores and the district’s expenditures (expn\_stu); run line 16 to find the covariance between these variables.

The general command for finding the covariance is cov(); inside the parentheses you would place the data frame and variable names, as shown in the next line of code, or you can attach the data frame and then just list the variable names, as shown in the subsequent line of code.

> cov(newdata$expn\_stu,newdata$testscr)

> attach(newdata); cov(expn\_stu, testscr, use="everything")

Using the means, variances, and covariances, you can calculate the simple regression line for the student-to-teacher ratio’s effect on test scores. This entails filling in this table:

|  |  |  |
| --- | --- | --- |
| Slope () |  |  |
| Intercept () |  |  |

Run lines 17–18 to check the results of the first regression, showing the effects of student-to-teacher ratio: > reg1<-lm(newdata$testscr~newdata$expn\_stu)

> summary(reg1)

The most important deduction from this model is that a one-unit increase in a district’s expenditures causes a \_\_\_\_\_−unit \_\_\_\_\_\_\_\_ (increase? decrease?) in the average test scores.

In lecture, we spent some time talking about competing explanations for the same phenomenon, which might take the following form: “Conventional wisdom holds that how much school districts spend is an important determinant of students’ test scores. However, an alternative theory posits that a school district’s demographics are more important.”

Run lines 20–21 to find the mean level of average household income in the district (avginc), to find the variance in average household incomes across districts, and to find the covariance of this variable with test scores.

> mean(newdata$avginc); var(newdata$avginc)

> cov(newdata$avginc,newdata$testscr)

Using the means, variances, and covariance, you can calculate the simple regression for average household income’s effect on test scores. This entails filling in the following table:

|  |  |  |
| --- | --- | --- |
| Slope () |  |  |
| Intercept () |  |  |

Run lines 22–23 to check the results of the second regression, showing effects of average income:

> reg2<-lm(newdata$testscr~newdata$avginc)

> summary(reg2)

The most important deduction from this model is that a one-unit increase in the value of average income causes a \_\_\_\_\_−unit \_\_\_\_\_\_\_\_ (increase? decrease?) in the average test scores. Average income explains \_\_\_\_\_\_ percent of the variation in test scores across districts.

**B. Multiple Regression Coefficients**

To save time, you could simply run line 25 to find the correlation matrix between the five variables (the code [,c(2:6)] selects only the last five variables, since *district* is not numeric) or run lines 26–27 to find the variance-covariance matrix for four of the variables (the code [,c(2,4:6)] selects the second, third, fourth, and sixth variables only).

Run line 28 to show the mean value for a new variable, the percent of students who are English language learners:

> mean(newdata$el\_pct)

When examining the matrix of variances and covariances, be sure that you can locate the values that you examined above (covariances with the dependent variable are shown in the first column *and* the first row; variances are shown along the main diagonal).

Run lines 31–32 to estimate a multiple regression model using the average household income and the percent of English language learners in a district. These are two variables that are outside the control of school districts. Here is the code for the baseline model:

> base<-lm(newdata$testscr~newdata$avginc+newdata$el\_pct)

> summary(base)

Combined, average income and the percent English language learners explains \_\_\_\_\_\_ percent of variation in test scores. How does this compare to the R2 from the simple regressions of test scores on district income in part A?

The coefficient for district income in the baseline model is \_\_\_\_\_ ; this coefficient’s value in the simple regression (lines 24–25) was \_\_\_\_\_ ; in your own words, what accounts for the difference?

Run line 33 to see the *added variable plots*, which illustrate the partial effect of one explanatory variable after accounting for the partial effect of the other explanatory variable. Since there were two variables in the baseline model, there will be two plots.

> avPlots(base)

Run line 35 to check the first key property, which is that the mean residual equals zero:

> round(mean(base$residuals), digits=4)

Run lines 36–37 to examine the correlations between the baseline model’s residuals and the two included variables. Both correlations ought to equal zero – indeed, this is the assumption that we use to begin the algebraic steps that lead to the equations for the multiple regression coefficients!

> round(cor(base$residuals, newdata$el\_pct), digits=4)

> round(cor(base$residuals, newdata$avginc), digits=4)

Run lines 38–39 to examine the correlation between the baseline model’s residuals and two omitted variables, which in this case are the student-teacher ratio and per-pupil expenditures.

> round(cor(base$residuals, newdata$str), digits=4)

> round(cor(base$residuals, newdata$expn\_stu), digits=4)

Do either of these variables seem to have a relationship to the dependent variable (test scores) that isn’t already accounted for in the included variables? In other words, is either of these variables “relevant”?

**C. Exploring Omitted Variable Bias and Proxies**

Run lines 42–43 to estimate and then examine the results of a model using the two demographic variables plus a policy variable (per-pupil expenditures) that whose “relevance” you just assessed:

> baseplus<-lm(testscr ~ avginc + el\_pct + expn\_stu,newdata)

> summary(baseplus)

Is your R-Studio showing you scientific notation? If it is and you would prefer it not to, then you can run the following line of code, then re-run line 43, to examine the regression estimates with standard notation.

> options(scipen=999)

Examine the results, and please answer three questions:

* Does district expenditures have a significant effect on test scores in the multivariate model that controls for average income and English-language learners?
* Does either the effect of average income or the effect of English-language learners change from the baseline model when the district expenditures variable is added?
* Does the fit of the model improve when the district expenditures variable is added?

If you also run line 44, displaying the added-variable plots, then you ought to observe that the student-to-teacher ratio has no impact.

> avPlots(baseplus)

Line 47 will run a new model that removes average income from the specification, while leaving in district expenditures. In other words, you are running a regression that you should strongly suspect will have an omitted variable bias. Run line 48 to examine the coefficients.

> proxy<-lm(testscr ~ el\_pct + expn\_stu, newdata)

> summary(proxy)

Compared to the base and baseplus models, the coefficient for el\_pct should have changed (base \_\_\_\_\_ ; baseplus \_\_\_\_\_ ; proxy \_\_\_\_\_) and the R-squared should be lower (base \_\_\_\_\_ ; baseplus \_\_\_\_\_ ; proxy \_\_\_\_\_) in the model using the proxy variable.

The coefficient for expn\_stu also should have changed (baseplus \_\_\_\_\_ ; proxy \_\_\_\_\_) when we omit average income.

Run lines 50–52 to verify two properties of the residuals: that their mean equals zero, and that they are uncorrelated to the included regressors:

> round(mean(resid(proxy)), digits=4)

> round(cor(resid(proxy), newdata$el\_pct), digits=4)

> round(cor(resid(proxy), newdata$expn\_stu), digits=4)

Next run line 53 to examine whether the residual is uncorrelated to the omitted regressor:

> round(cor(resid(proxy), newdata$avginc), digits=4)

When you ran the baseline model earlier, the residual was uncorrelated to avginc, because residuals are forced to be uncorrelated to all included regressors. Now, when you run the model *without* avginc, the residual *is correlated* to this omitted variable!

Additionally, the coefficients on el\_pct and expn\_stu change when avginc is omitted, because the regression model gives some of the credit that average income deserves to other variables with which it covaries.

**D. Standard Errors for Simple (Bivariate) Regressions**

Now let us turn our attention to standard errors. We are interested in two types of standard errors: the standard error of the regression, and the standard error of a regression coefficient. For the moment, let’s focus on the second simple regression you estimated, exploring how school districts’ average income impacts test scores.

To find the correlation between average income and average test scores in a district, run line 56:

> r = cor(newdata$testscr, newdata$avginc); r; r^2

This line of code retrieves the correlation coefficient for the variables you specify, and names it *r*, and then displays the correlation coefficient and the squared correlation coefficient.; the computed value of *r*2 = .

Now double-check that *r*2 = *R*2 by regressing testscr on avginc as shown in line 57.

> summary(reg2)[8]

R lists the root mean square error (a.k.a. *sigma*) among the regression results when you use the summary() command, but it calls it **Residual standard error**. You can run line 59:

> summary(reg2)$sigma

Root MSE = \_\_\_\_\_\_\_ ; recall that *sigma* estimates the standard deviation of the disturbances, and that *sigma*2 is our sample estimate of the error variance.

You can obtain Root MSE by running lines 61–65 in the R script:

> reg2.res<-resid(reg2)

> res2<-reg2.res^2

> ssr<-sum(res2)

> mse<-ssr/(length(res2)-2)

> rmse<-sqrt(mse)

The first three lines generate the sum of squared residuals (SSR = \_\_\_\_\_\_ ). In your own words, describe what steps are being taken when you these three lines of code.

The next two lines of code shown above generate the mean squared error, which is an estimate of the variance of the disturbances (MSE = \_\_\_\_\_\_ ), and the root mean squared error, also known as *sigma* (root MSE = \_\_\_\_\_\_ ).

Earlier in the semester, I claimed that the standard error of a simple regression coefficient equaled the square root of a ratio: *sigma*2 divided by the sum: ; this latter term equals *n*–1 times the variance of *x*. Run line 67 to display the coefficients and standard errors; focus on the standard error of avginc , which we are treating as *x* at the moment. Then run lines 69–71 to calculate the denominator (*n*–1 times the variance of *x*), to calculate the ratio, and to take the root.

Do the values match?

Over the next couple lectures we will explore how changing the model specification impacts the standard error, and the equations will be slightly more complicated. But, hopefully this gives you some insight into the inputs for computing standard errors.

To clear the **Environment**, type rm(list=ls()) or click on the broom icon.

To clear the **Console** window, type Ctrl-*l*