Understanding and Plotting Fitted Multiple Regression Models

2018-07-29

Introduction and Research Question

In this set of notes, we will again examine the question of whether education level is related to income using the *riverside.csv* data from C. Lewis-Beck & Lewis-Beck (2016). In particular, we will re-examine the multiple regression model and try to gain a better understanding of *statistical control*. While doing this, you will also learn how to plot the fitted multiple regression model.

Preparation

```r
# Load libraries
library(broom)
library(dplyr)
library(ggplot2)
library(readr)
library(tidyr)

# Read in data
city = read_csv(file = "~/Documents/github/epsy-8251/data/riverside.csv")

# Fit the multiple regression model
lm.1 = lm(income ~ 1 + education + seniority, data = city)

# Coefficient-level information
tidy(lm.1)
```

```
         term  estimate std.error statistic         p.value
1 (Intercept) 6769.1720 5372.8914  1.259875 0.2177593428983
2   education 2251.8456  334.6443  6.729073 0.0000002202903
3   seniority  738.7965  210.0954  3.516481 0.0014597774256
```

Predictions

The fitted equation is,

$$\widehat{\text{Income}} = 6769 + 2252(\text{Education}) + 739(\text{Seniority})$$

Let's predict the average income for employees who have differing education levels,

- Education level = 10 years
- Education level = 11 years
- Education level = 12 years

Let's also assume that these employees all have 10 years of seniority.

Education level	Seniority	Predicted Income
10	10	$\widehat{\text{Income}} = 6769 + 2252(10) + 739(10) = 36,679$
11	10	$\widehat{\text{Income}} = 6769 + 2252(11) + 739(10) = 38,931$
12	10	$\widehat{\text{Income}} = 6769 + 2252(12) + 739(10) = 41,183$

In this example, the value of `seniority` is "constant" across the three employees. Education level differs by one-year between each subsequent employee. The difference in predicte income between these employee is \$2,252. When we hold seniority constant, the predicted difference in income between employees with a one-year difference in education is \$2,252.

What if we had chosen seniority level of 11 years instead?

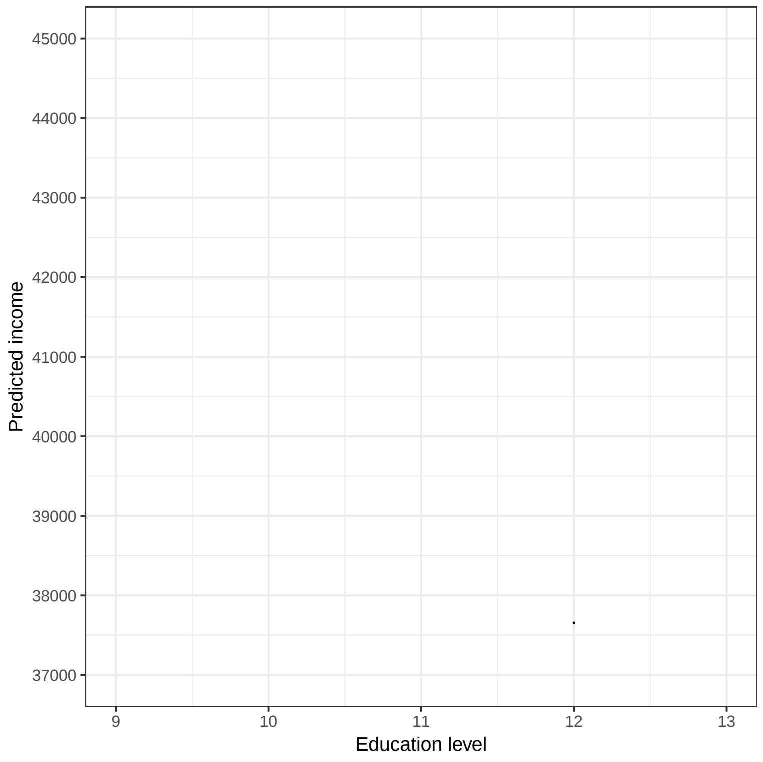
Education level	Seniority	Predicted Income
10	11	$\widehat{\text{Income}} = 6769 + 2252(10) + 739(11) = 37,418$
11	11	$\widehat{\text{Income}} = 6769 + 2252(11) + 739(11) = 39,670$
12	11	$\widehat{\text{Income}} = 6769 + 2252(12) + 739(11) = 41,922$

The predicted incomes are higher for these employees because they have a higher seniority level, but again, when we hold seniority constant, the predicted difference in income between employees with a one-year difference in education is \$2,252.

This relationship between education level and income will be true, regardless of which value we pick for seniority.

Plotting this Relationship

- Sketch the scatterplot to display the ordered pairs (`education`, `predicted income`) for those employess whose seniority value is 10.



- Add the ordered pairs (`education`, `predicted income`) for those employees whose seniority value is 11 to the plot, but use a different symbol.

Holding Education Level Constant

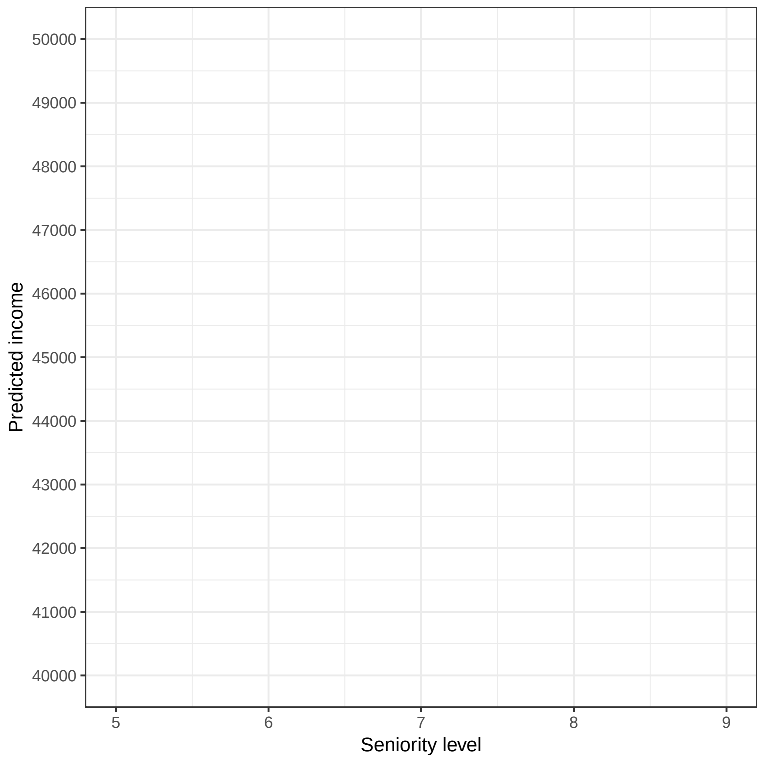
What happens if we pick a fixed education level and look at how the predicted income varies for different values of seniority?

$$\widehat{\text{Income}} = 6769 + 2252(\text{Education}) + 739(\text{Seniority})$$

Education level	Seniority	Predicted Income
10	11	$\widehat{\text{Income}} = 6769 + 2252(16) + 739(6) = ???$
11	11	$\widehat{\text{Income}} = 6769 + 2252(16) + 739(7) = ???$
12	11	$\widehat{\text{Income}} = 6769 + 2252(16) + 739(8) = ???$

Try it and convince yourself!

- Sketch the predicted regression lines showing the effect of seniority on income for employees with 16 years of education



- Add the predicted regression lines showing the effect of seniority on income for employees with 14 years of education.

Using R to Compute Predicted Values

We can use the `predict()` function to obtain predicted values once we have fitted an `lm()` object. To do this, we need to have a data frame of predictor values. The columns are the predictor variables, and these need to have the EXACT same names as the predictors used in the `lm()` function, in our case, `education` and `seniority`.

```
employees = data.frame(
  education =  c(10, 11, 12),
  seniority =  c(10, 10, 10)
  )

# View data frame
employees
```

```
  education seniority
1        10        10
2        11        10
3        12        10
```

Now we can use the `predict()` function to do the actual prediction. This function takes two arguments. The first is the name of the `lm()` object. The second argument is `newdata=` and takes the name of the data frame you want to predict from.

```
predict(lm.1, newdata = employees)
```

```
       1        2        3 
36675.59 38927.44 41179.28 
```

We could also append the predicted values as a new column on the data frame to see which set of predictor values go with which predicted values.

```
employees %>%
  mutate(
    yhat = predict(lm.1, newdata = employees)
    )
```

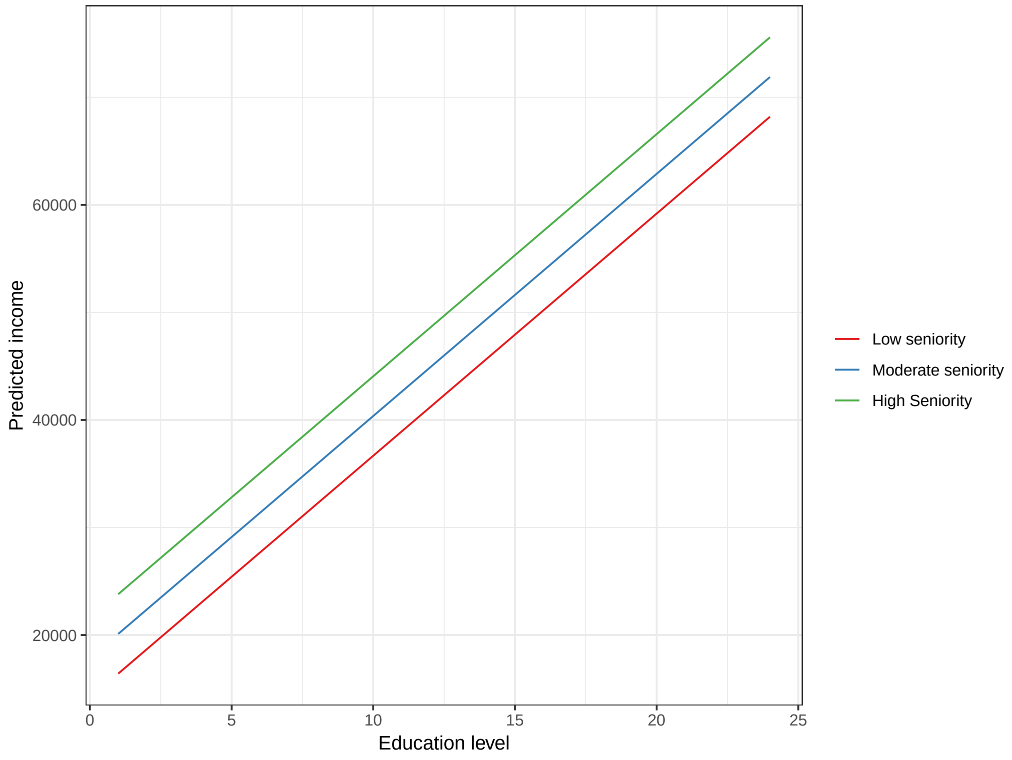
```
  education seniority     yhat
1        10        10 36675.59
2        11        10 38927.44
3        12        10 41179.28
```

YOUR TURN: Try using R to compute the predicted values for income for the following employees:

- Education level = 16; Seniority = 6
- Education level = 16; Seniority = 7
- Education level = 16; Seniority = 8

Plotting the Results

With two (or more) effects in the model we have more than one potential way to display the fitted results. In general, we will display one predictor through the slope of the line plotted (same as we did with only one predictor), and EVERY OTHER predictor will be shown through one or more lines. For example, below, we show the effect of education level through the slope of the lines, and the effect of seniority through the three different lines.



In thinking about the plot you want to create, answer the questions:

- Which predictor do you want to display on the x-axis?
- How many levels of the remaining predictors do you want to display?

Above we put education level on the x-axis and displayed three (3) different levels of seniority.

After you have made those decisions, then you can determine:

- What range of values should we use for the predictor on the x-axis?
- Which discrete values should we choose for the the remaining predictors?

For the range of values for education level, I used the range of the data.

```r
summary(city$education)
```

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
      8      12      16      16      20      24 
```

This suggested that a range of 8–24 might be reasonable.

What about for the values to choose for seniority? Again the data might suggest reasonable values:

```r
summary(city$seniority)
```

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
   1.00    9.75   15.00   14.81   20.25   27.00 
```

Here I picked the first quartile, the median, and the third quartile; 10, 15, and 20. I also rounded to the nearest whole number to make interpretation easier.

Now that we have made these decisions, we are basically ready. The general process for plotting is as follows:

- Set up a data set to predict from. It should include a sequence of values for the predictor you are plotting on the x-axis for EACH value of the other variable you have selected to show different lines for.
- Predict the y-hat values using the fitted model and the dat set you just created
- Turn every predictor variable that is not on the x-axis into a factor. This we do for better plotting and labels.
- Create a line plot of y-hat values vs the X values. Use the `group=` aesthetic to separate by the other predictors.

To start this process, we need to basically set up data that includes the range of values (8–24) for education for ALL three values of seniority. To do this we will use the `crossing()` function from the **tidyr** package to create a tibble (data frame) of all combinations of the declared values in the `education` and `seniority` variables.

```r
plot_data = crossing(
  education = seq(from = 1, to = 24, by = 1),
  seniority = c(10, 15, 20)
  )

# View data
plot_data
```

```
# A tibble: 72 x 2
   education seniority
       <dbl>     <dbl>
 1         1        10
 2         1        15
 3         1        20
 4         2        10
 5         2        15
 6         2        20
 7         3        10
 8         3        15
 9         3        20
10         4        10
# ... with 62 more rows
```

Now that we have the predictor values, we can use the `predict()` function to obtain a column of y-hats.

```
plot_data %>%
  mutate(
    yhat = predict(lm.1, newdata = plot_data)
    )
```

```
# A tibble: 72 x 3
   education seniority   yhat
       <dbl>     <dbl>  <dbl>
 1         1        10 16409.
 2         1        15 20103.
 3         1        20 23797.
 4         2        10 18661.
 5         2        15 22355.
 6         2        20 26049.
 7         3        10 20913.
 8         3        15 24607.
 9         3        20 28301.
10         4        10 23165.
# ... with 62 more rows
```

We are almost ready to plot. One last step that will make the plotting look better is that we need to turn the predictor that defines the different lines (in our case `seniority`) into a factor. This will give us a discrete color palette, and give better labels.

```
plot_data %>%
  mutate(
    yhat = predict(lm.1, newdata = plot_data),
    seniority2 = factor(seniority,
      levels = c(10, 15, 20),
      labels = c("Low seniority", "Moderate seniority", "High seniority")
      )
    )
```

```
# A tibble: 72 x 4
   education seniority   yhat seniority2
       <dbl>     <dbl>  <dbl> <fct>
 1         1        10 16409. Low seniority
 2         1        15 20103. Moderate seniority
 3         1        20 23797. High seniority
 4         2        10 18661. Low seniority
 5         2        15 22355. Moderate seniority
 6         2        20 26049. High seniority
 7         3        10 20913. Low seniority
 8         3        15 24607. Moderate seniority
 9         3        20 28301. High seniority
10         4        10 23165. Low seniority
# ... with 62 more rows
```

Finally, we are ready to plot.

```r
# Store the plotting data into an object
plot_data_2 = plot_data %>%
  mutate(
    yhat = predict(lm.1, newdata = plot_data),
    seniority2 = factor(seniority,
      levels = c(10, 15, 20),
      labels = c("Low seniority", "Moderate seniority", "High seniority")
      )
    )

# Create the plot
ggplot(data = plot_data_2, aes(x = education, y = yhat, group = seniority2, color = seniority2)) +
  geom_line() +
  theme_bw() +
  xlab("Education level") +
  ylab("Predicted income") +
  scale_color_brewer(name = "", palette = "Set1")
```

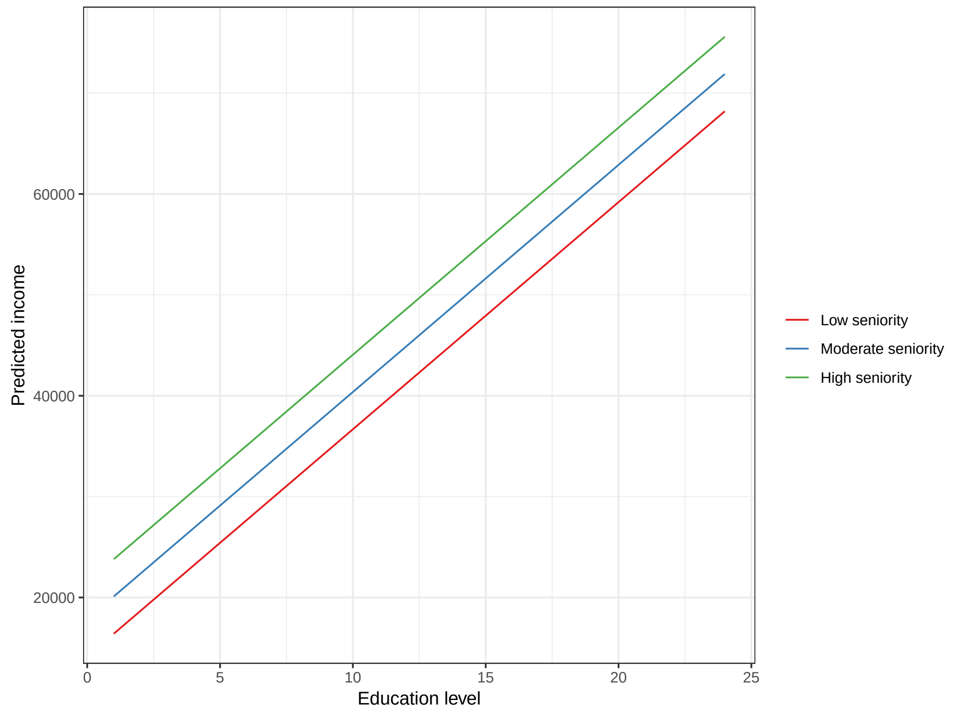


Figure 1: Predicted income as a function of education level for employees with 10 (low), 15 (moderate), and 20 (high) years of seniority.

Seniority on the x-Axis

Here we go through the process again, but put seniority on the x-axis rather than education.

- Which predictor do you want to display on the x-axis? *Seniority*
- How many levels of the remaining predictors do you want to display? *2*
- What range of values should we use for the predictor on the x-axis? *1–27*
- Which discrete values should we choose for the the remaining predictors? *12, 16*

Rather than create different objects, this time I am going to use **dplyr**'s piping syntax from start to finish.

```r
crossing(
  seniority = seq(from = 1, to = 27, by = 1),
  education = c(12, 16)
  ) %>%
  mutate(
    yhat = predict(lm.1, newdata = .)
    ) %>%
  mutate(
    education2 = factor(education,
                        levels = c(12, 16),
                        labels = c("High school", "College")
                        )
    ) %>%
  ggplot(aes(x = seniority, y = yhat, group = education2, color = education2)) +
    geom_line() +
    theme_bw() +
    xlab("Seniority level") +
    ylab("Predicted income") +
    scale_color_brewer(name = "", palette = "Set1")
```

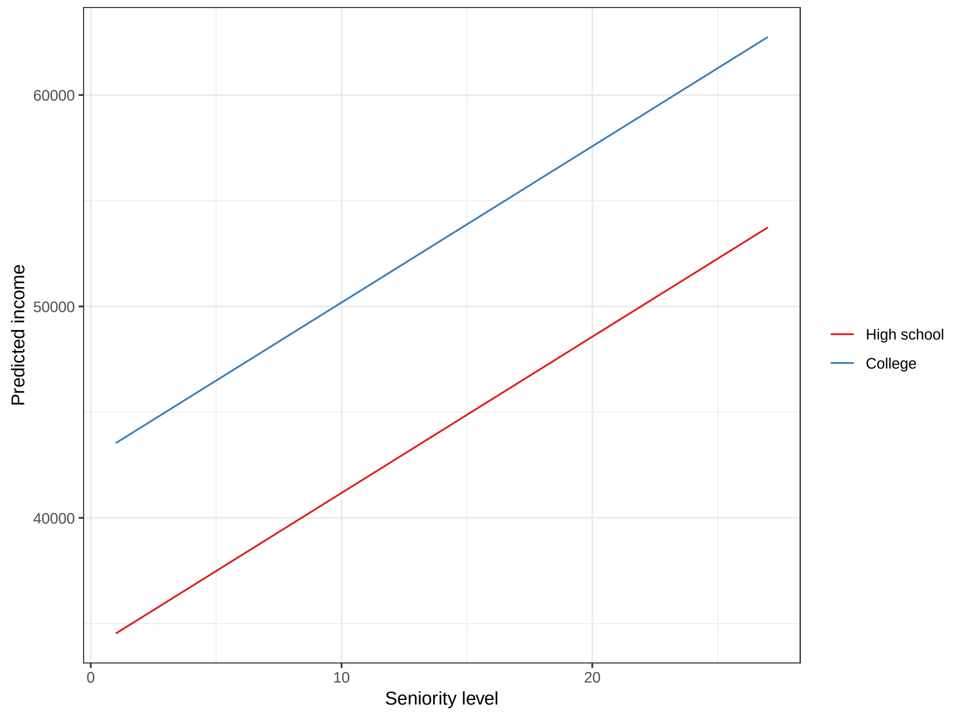


Figure 2: Predicted income as a function of seniority level for employees with a high school (12 yrs) and college (16 yrs) education.

One Last Option

Sometimes you do not want to show the effect of a predictor. Although we still have to include it in fitted model to get appropriate predicted values, we can opt not to display it. To do that, we will set that predictor to its mean value (or any other value; but having only one value will produce only one line . . . resulting in not displaying that effect). Because there is only one line, we also do not have to turn that predictor into a factor (since we are not plotting it).

```
crossing(
  seniority = seq(from = 1, to = 27, by = 1),
  education = mean(city$education)
  ) %>%
  mutate(
    yhat = predict(lm.1, newdata = .)
    ) %>%
  ggplot(aes(x = seniority, y = yhat)) +
    geom_line() +
    theme_bw() +
    xlab("Seniority level") +
    ylab("Predicted income")
```

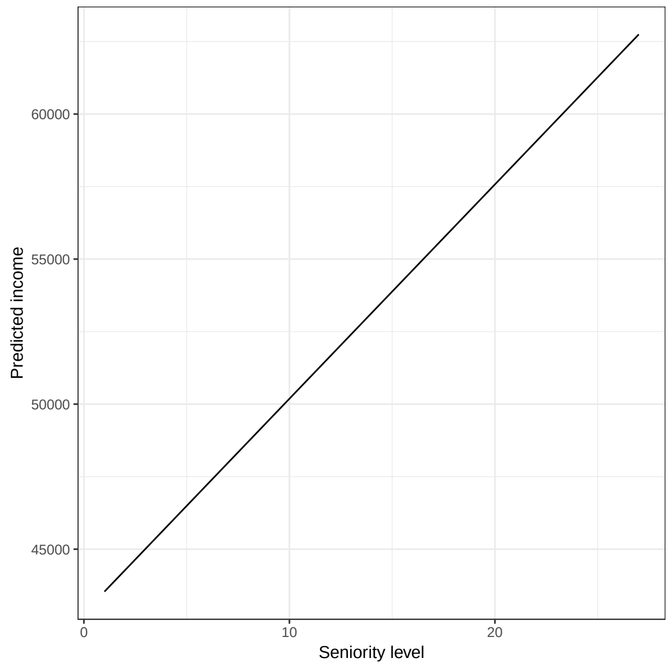


Figure 3: Predicted income, controlling for education level, as a function of seniority level. Education was set to its mean value of 16 yrs.

References

Lewis-Beck, C., & Lewis-Beck, M. (2016). *Applied regression: An introduction* (2nd ed.). Thousand Oaks, CA: Sage.