Tutorial 8: Multiple Regression

Johannes Karreth

RPOS 517, Day 8

This tutorial shows you:

- how to estimate a regression model with multiple predictors
- how to present regression results graphically
- how to calculate standardized regression coefficients

Note on copying & pasting code from the PDF version of this tutorial: Please note that you may run into trouble if you copy & paste code from the PDF version of this tutorial into your R script. When the PDF is created, some characters (for instance, quotation marks or indentations) are converted into non-text characters that R won't recognize. To use code from this tutorial, please type it yourself into your R script or you may copy & paste code from the *source file* for this tutorial which is posted on my website.

Note on R functions discussed in this tutorial: I don't discuss many functions in detail here and therefore I encourage you to look up the help files for these functions or search the web for them before you use them. This will help you understand the functions better. Each of these functions is well-documented either in its help file (which you can access in R by typing `?ifelse`, for instance) or on the web. The *Companion to Applied Regression* (see our syllabus) also provides many detailed explanations.

As always, please note that this tutorial only accompanies the other materials for Day 8 and that you are expected to have worked through the videos and reading for that day before tackling this tutorial.

Basic setup

This tutorial builds on the readings and videos you worked through for Day 8. These materials introduce a new form of the familiar regression equation to you. On Days 6 and 7, you worked with one outcome variable y and one explanatory variable x. The parameters you recovered via OLS were the intercept, α, and the slope coefficient β. For today, you encounter a second explanatory variable x_2, next to x_1.

This also increases the number of slope coefficients to two: β_1 is associated with x_1, and β_2 is associated with x_2:

$$y_i = \alpha + \beta_1 x_1 + \beta_2 x_2 + \varepsilon$$

We can simulate the data-generating process for this y, mirroring the approach you saw in the tutorial for Day 6:

```
set.seed(123)
n.obs <- 25
x1 <- rnorm(n = n.obs, mean = 5, sd = 2)
x2 <- runif(n = n.obs, min = -7, max = 7)
i <- c(1:n.obs)
e <- rnorm(n = n.obs, mean = 0, sd = 1)
a <- 2
b1<- 0.5
b2 <- -0.75
y <- a + b1 * x1 + b2 * x2 + e
sim.dat <- data.frame(y, x1, x2)
```

When estimating a linear model in statistical software, in our case R, the only change you have to make to add the additional explanatory variable(s) to the formula for the model you estimate:

```
mod <- lm(y ~ x1 + x2, data = sim.dat)
summary(mod)
```

```
##
## Call:
## lm(formula = y ~ x1 + x2, data = sim.dat)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -1.3996 -0.7212 -0.1522  0.4452  1.7291
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   1.7548     0.5700   3.078  0.00549 **
## x1            0.5517     0.1084   5.088 4.26e-05 ***
## x2           -0.7489     0.0518 -14.457 1.03e-12 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.9898 on 22 degrees of freedom
## Multiple R-squared:  0.9245, Adjusted R-squared:  0.9177
## F-statistic: 134.7 on 2 and 22 DF,  p-value: 4.53e-13
```

This `lm` object has some elements that you can use later on. You can view the structure of the model object with the `str()` function:

```
str(mod)
```

```
## List of 12
##  $ coefficients : Named num [1:3] 1.755 0.552 -0.749
##   ..- attr(*, "names")= chr [1:3] "(Intercept)" "x1" "x2"
##  $ residuals    : Named num [1:25] -0.72 0.298 -1.4 0.42 0.773 ...
##   ..- attr(*, "names")= chr [1:25] "1" "2" "3" "4" ...
##  $ effects      : Named num [1:25] -22.519 7.697 14.309 0.379 1.009 ...
##   ..- attr(*, "names")= chr [1:25] "(Intercept)" "x1" "x2" "" ...
##  $ rank         : int 3
##  $ fitted.values: Named num [1:25] 8.66 4.87 3.1 8.56 4.02 ...
##   ..- attr(*, "names")= chr [1:25] "1" "2" "3" "4" ...
##  $ assign       : int [1:3] 0 1 2
##  $ qr           :List of 5
##   ..$ qr   : num [1:25, 1:3] -5 0.2 0.2 0.2 0.2 0.2 0.2 0.2 0.2 0.2 ...
##   .. ..- attr(*, "dimnames")=List of 2
##   .. .. ..$ : chr [1:25] "1" "2" "3" "4" ...
##   .. .. ..$ : chr [1:3] "(Intercept)" "x1" "x2"
##   .. ..- attr(*, "assign")= int [1:3] 0 1 2
##   ..$ qraux: num [1:3] 1.2 1.02 1.35
##   ..$ pivot: int [1:3] 1 2 3
##   ..$ tol  : num 1e-07
##   ..$ rank : int 3
##   ..- attr(*, "class")= chr "qr"
```

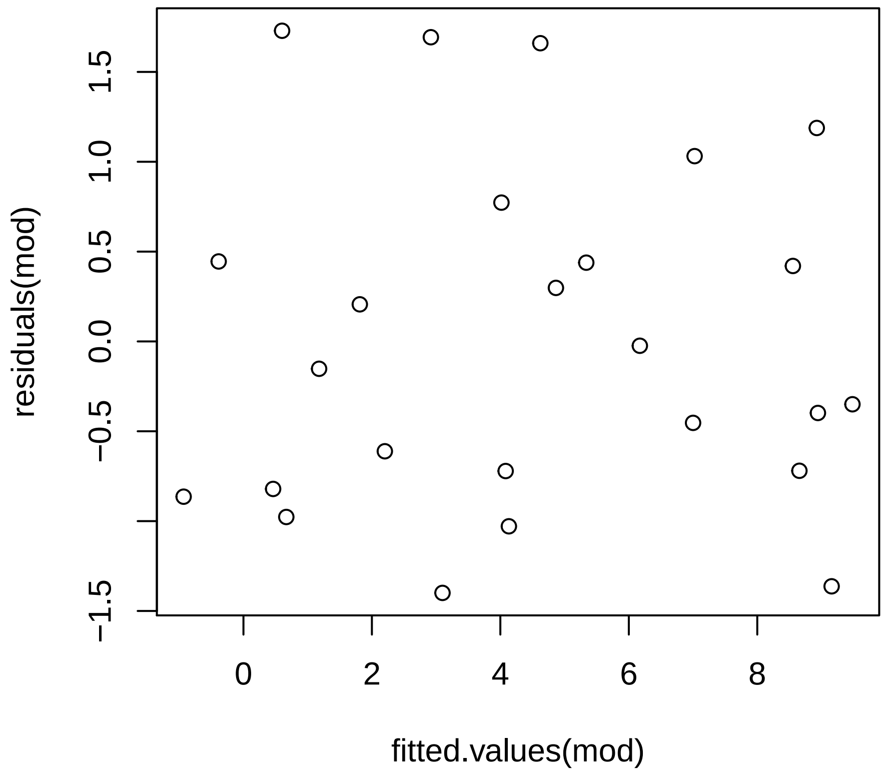
```
##  $ df.residual  : int 22
##  $ xlevels      : Named list()
##  $ call         : language lm(formula = y ~ x1 + x2, data = sim.dat)
##  $ terms        :Classes 'terms', 'formula'  language y ~ x1 + x2
##   .. ..- attr(*, "variables")= language list(y, x1, x2)
##   .. ..- attr(*, "factors")= int [1:3, 1:2] 0 1 0 0 0 1
##   .. .. ..- attr(*, "dimnames")=List of 2
##   .. .. .. ..$ : chr [1:3] "y" "x1" "x2"
##   .. .. .. ..$ : chr [1:2] "x1" "x2"
##   .. ..- attr(*, "term.labels")= chr [1:2] "x1" "x2"
##   .. ..- attr(*, "order")= int [1:2] 1 1
##   .. ..- attr(*, "intercept")= int 1
##   .. ..- attr(*, "response")= int 1
##   .. ..- attr(*, ".Environment")=<environment: R_GlobalEnv>
##   .. ..- attr(*, "predvars")= language list(y, x1, x2)
##   .. ..- attr(*, "dataClasses")= Named chr [1:3] "numeric" "numeric" "numeric"
##   .. .. ..- attr(*, "names")= chr [1:3] "y" "x1" "x2"
##  $ model        :'data.frame':   25 obs. of  3 variables:
##   ..$ y : num [1:25] 7.94 5.16 1.7 8.98 4.79 ...
##   ..$ x1: num [1:25] 3.88 4.54 8.12 5.14 5.26 ...
##   ..$ x2: num [1:25] -6.358 -0.809 4.185 -5.293 0.853 ...
##   ..- attr(*, "terms")=Classes 'terms', 'formula'  language y ~ x1 + x2
##   .. .. ..- attr(*, "variables")= language list(y, x1, x2)
##   .. .. ..- attr(*, "factors")= int [1:3, 1:2] 0 1 0 0 0 1
##   .. .. .. ..- attr(*, "dimnames")=List of 2
##   .. .. .. .. ..$ : chr [1:3] "y" "x1" "x2"
##   .. .. .. .. ..$ : chr [1:2] "x1" "x2"
##   .. .. ..- attr(*, "term.labels")= chr [1:2] "x1" "x2"
##   .. .. ..- attr(*, "order")= int [1:2] 1 1
##   .. .. ..- attr(*, "intercept")= int 1
##   .. .. ..- attr(*, "response")= int 1
##   .. .. ..- attr(*, ".Environment")=<environment: R_GlobalEnv>
##   .. .. ..- attr(*, "predvars")= language list(y, x1, x2)
##   .. .. ..- attr(*, "dataClasses")= Named chr [1:3] "numeric" "numeric" "numeric"
##   .. .. .. ..- attr(*, "names")= chr [1:3] "y" "x1" "x2"
##  - attr(*, "class")= chr "lm"
```

We can extract some of these quantities with commands that you're already familiar with:

```
coef(mod)
```

```
## (Intercept)          x1          x2
##   1.7548443   0.5516712  -0.7488609
```

```
plot(x = fitted.values(mod), y = residuals(mod))
```



Before you advance, a reminder: the assumptions we discussed on Day 6 are the same assumptions underling the linear regression model with more than one predictor. These assumptions are:

- Linearity (A1)
- Constant error variance: $V(\varepsilon) = \sigma^2$ (A2)
- Normality of the errors: $\varepsilon \ \mathcal{N}(0, 1)$ (A3)
- Independence of observations: ε_i and ε_j are independent (A4)
- None of the predictors x_1, x_2, etc. is a function of ε (A5)

Example data: Occupational prestige

We'll now use some example data that you encountered in your *AR* reading. These data come from the "car" package, so I first load the package, then create the object `prestige.dat` containing the dataset.

```r
library(car)
prestige.dat <- data.frame(Prestige)
head(prestige.dat)
```

```
##                     education income women prestige census type
## gov.administrators      13.11  12351 11.16     68.8   1113 prof
## general.managers        12.26  25879  4.02     69.1   1130 prof
## accountants             12.77   9271 15.70     63.4   1171 prof
## purchasing.officers     11.42   8865  9.11     56.8   1175 prof
## chemists                14.62   8403 11.68     73.5   2111 prof
## physicists              15.64  11030  5.13     77.6   2113 prof
```

```r
summary(prestige.dat)
```

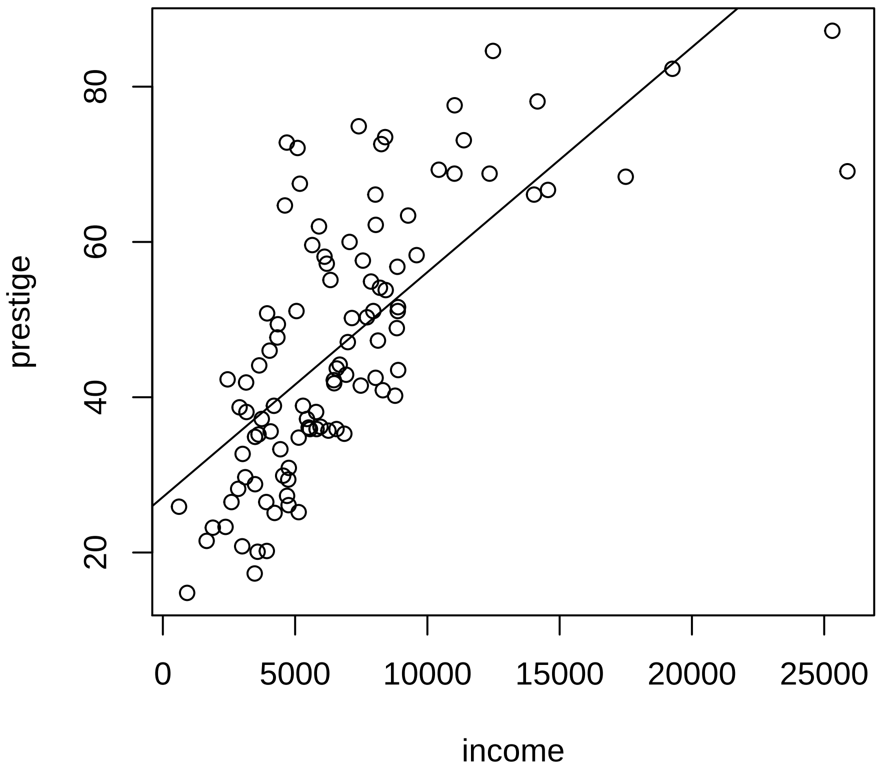
```
##    education          income          women           prestige    
##  Min.   : 6.380   Min.   :  611   Min.   : 0.000   Min.   :14.80  
##  1st Qu.: 8.445   1st Qu.: 4106   1st Qu.: 3.592   1st Qu.:35.23  
##  Median :10.540   Median : 5930   Median :13.600   Median :43.60  
##  Mean   :10.738   Mean   : 6798   Mean   :28.979   Mean   :46.83  
##  3rd Qu.:12.648   3rd Qu.: 8187   3rd Qu.:52.203   3rd Qu.:59.27  
##  Max.   :15.970   Max.   :25879   Max.   :97.510   Max.   :87.20  
##      census       type   
##  Min.   :1113   bc  :44  
##  1st Qu.:3120   prof:31  
##  Median :5135   wc  :23  
##  Mean   :5402   NA's: 4  
##  3rd Qu.:8312            
##  Max.   :9517            
```

The dataset has 102 rows (each row is one occupation) and 6 variables. The variables are:

Variable	Description
`education`	Average education of occupational incumbents, years, in 1971.
`income`	Average income of incumbents, dollars, in 1971.
`women`	Percentage of incumbents who are women.
`prestige`	Pineo-Porter prestige score for occupation, from a social survey conducted in the mid-1960s.
`census`	Canadian Census occupational code.
`type`	Type of occupation. A factor with levels: b(lue)c(ollar), prof(essional), and w(hite)c(ollar).

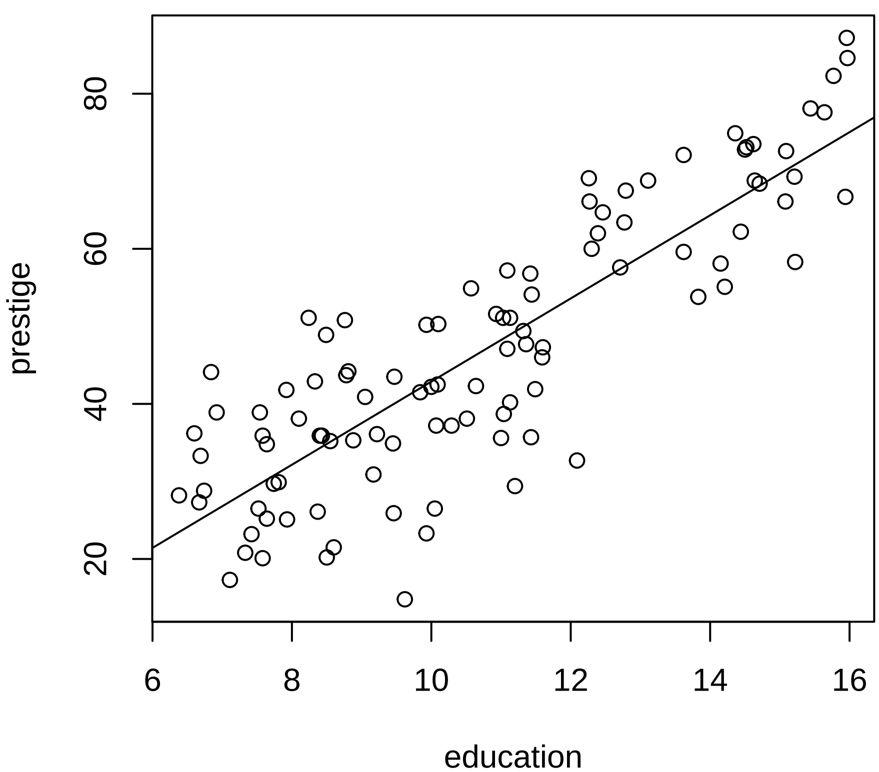
For this example, I'd like to investigate the correlates of the `prestige` score. I could first focus on the `income` of each occupation and create a scatterplot of `income` and `prestige`.

```r
with(prestige.dat, plot(x = income, y = prestige, main = ""))
income.mod <- lm(prestige ~ income, data = prestige.dat)
abline(income.mod)
```



I could also check the relationship between education and prestige:

```
with(prestige.dat, plot(x = education, y = prestige, main = ""))
education.mod <- lm(prestige ~ education, data = prestige.dat)
abline(education.mod)
```



Regression with two predictors

Considering that both variables are clearly related to occupational prestige, you might decide that both should be part of a multiple regression model. You can fit this model by adding both variables to the equation:

```
mod <- lm(prestige ~ income + education, data = prestige.dat)
summary(mod)
```

```
##
## Call:
## lm(formula = prestige ~ income + education, data = prestige.dat)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -19.4040  -5.3308   0.0154   4.9803  17.6889
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) -6.8477787  3.2189771  -2.127   0.0359 *
## income       0.0013612  0.0002242   6.071 2.36e-08 ***
## education    4.1374444  0.3489120  11.858  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 7.81 on 99 degrees of freedom
## Multiple R-squared:  0.798,  Adjusted R-squared:  0.7939
## F-statistic: 195.6 on 2 and 99 DF,  p-value: < 2.2e-16
```

You can already see some important quantities from the summary of the `lm` object. The intercept and the two slope coefficients are interpreted as you learned in today's video and the *AR* reading. To quote Fox (p. 89):

> The slope coefficients for the explanatory variables in multiple regression are *partial* coefficients... That is, each slope in multiple regression represents the "effect" on the response variable of a one-unit increment in the corresponding explanatory variable *holding constant* the value of the other explanatory variable.

In our example, this means that an occupation that requires one additional year of average `education` receives a 4.13 higher rating on the occupational `prestige` score, **regardless of the average income of that occupation**. The qualifier at the end may sound trivial, but it will become important later on. In this setup, you are specifying a model of the DGP that assumes that the effect of one explanatory variable on the response variable is completely independent of the value of other explanatory variables.

You can now use the handy set of functions in the "texreg" package to produce a publication-ready regression table. If you haven't done so, install the package. First, you may wan to print the table to your screen:

```
library(texreg)
screenreg(mod)
```

```
##
## =======================
##              Model 1
## -----------------------
```

```
## (Intercept)   -6.85 *
##               (3.22)
## income         0.00 ***
##               (0.00)
## education      4.14 ***
##               (0.35)
## -----------------------
## R^2            0.80
## Adj. R^2       0.79
## Num. obs.    102
## =======================
## *** p < 0.001, ** p < 0.01, * p < 0.05
```

This table can be improved. You can find out all available options by checking the help page using the `?screenreg` command. I make some modifications below that I recommend for your work with regression tables as well:

- print only two digits
- use only one marker (usually an asterisk) to show whether a coefficient's standard error lets you reject H_0: $\beta = 0$ at the 0.05 level
- use proper variable names that also show up in your main text
- reorder coefficients in a sensible order

```r
screenreg(list(mod),
          stars = 0.05,
          digits = 2,
          custom.model.names = c(""),
          custom.coef.names = c("Intercept", "Income", "Education"),
          reorder.coef = c(2, 3, 1))
```

```
##
## ===================
##
## -------------------
## Income       0.00 *
##             (0.00)
## Education    4.14 *
##             (0.35)
## Intercept   -6.85 *
##             (3.22)
## -------------------
## R^2          0.80
## Adj. R^2     0.79
## Num. obs.  102
## ===================
## * p < 0.05
```

1. Given what you know about p-values and z-scores, can you think of a shortcut (a calculation in your head) that would allow you to quickly figure out whether a coefficient estimate is "statistically significant", based on a point estimate of that coefficient and its standard error?

I can now use the same options for the `htmlreg()` and `texreg()` functions to produce tables for Word or for the PDF file produced by RMarkdown.

Table 2: Analysis of occupational prestige

Income	0.00^{*}
	(0.00)
Education	4.14^{*}
	(0.35)
Intercept	-6.85^{*}
	(3.22)
R^2	0.80
Adj. R^2	0.79
Num. obs.	102

$^{*}p < 0.05$

To create a table for a Word document that you can insert into a manuscript, use `htmlreg()` and save the table to your working directory:

```
htmlreg(list(mod),
          file = "~/Documents/Uni/Teaching/POS 517/Tutorials/Day 8 - Multiple regression/tab_m1.doc",
          stars = 0.05,
          digits = 2,
          custom.model.names = c(""),
          custom.coef.names = c("Intercept", "Income", "Education"),
          reorder.coef = c(2, 3, 1),
          caption = "Analysis of occupational prestige",
          caption.above = TRUE)
```

If you would like to add a table to the PDF file you're producing with RMarkdown, just replace `htmlreg()` with `texreg()` and be sure to set the options of your R code chunk to `results = "asis"`. Note that the `texreg()` function will not produce any output in a HTML or Word document that you are knitting from Rstudio. *Therefore, use `texreg()` only if you are knitting a PDF document, or if you are working in LaTeX.*

```
texreg(list(mod),
          stars = 0.05,
          digits = 2,
          custom.model.names = c(""),
          custom.coef.names = c("Intercept", "Income", "Education"),
          reorder.coef = c(2, 3, 1),
          caption = "Analysis of occupational prestige",
          caption.above = TRUE)
```

However, one problem you may notice with this output is that it returns a coefficient estimate for income that is 0.00 (but marked as "statistically significant"). The actual coefficient estimate we recover from the model object is 0.0013612:

```
coef(mod)
```

```
##  (Intercept)       income    education
## -6.847778720  0.001361166  4.137444384
```

Since a "significant" coefficient of 0.00 is an oxymoron, you should either increase the number of digits shown in the regression table, or consider rescaling the `income` variable.

2. Investigate the `income` variable and decide whether you should do anything with it before including it in your regression model.

Graphical output

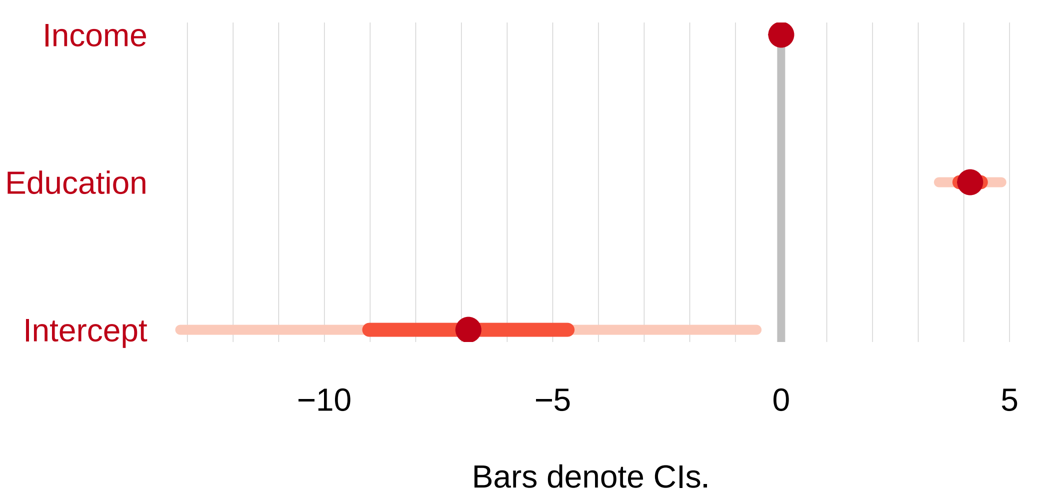
While you can summarize a regression model with the table we just produced, it is often beneficial to use graphical methods to present regression results. We'll devote more time to this later on in the seminar, but here are four papers you can consult that provide some detailed explanations for why regression results (and other statistical quantities) are often best presented graphically. The papers also make useful recommendations on how to present these quantities.

- Epstein, Lee, Andrew D. Martin, and Matthew M. Schneider (2006). "On the Effective Communication of the Results of Empirical Studies, Part I". *Vanderbilt Law Review* 59: 1811-1871.
- Epstein, Lee, Andrew D. Martin, and Christina L. Boyd (2007). "On the Effective Communication of the Results of Empirical Studies, Part II". *Vanderbilt Law Review* 60: 801-846.
- Kastellec, Jonathan P. and Leoni, Eduardo L (2007). "Using Graphs Instead of Tables in Political Science". *Perspectives on Politics* 5 (4): 755-771.
- Jacoby, William G. "The Dot Plot: A Graphical Display for Labeled Quantitative Values". *The Political Methodologist* 14 (1): 6-14.

The `plotreg()` function

R offers some easy functions to create dot plots for regression coefficients. If you are already using the "texreg" package to create tables, its `plotreg()` function is a very easy-to-use extension. It works just like the `screenreg()` function and the other ones you just encountered, but an acceptable plot requires even fewer options to be set:

```
plotreg(list(mod),
          # file = "/Users/johanneskarreth/Documents/Uni/Teaching/POS 517/Tutorials/Day 8 -
            # Multiple regression/plot_m1.pdf",
          custom.coef.names = c("Intercept", "Income", "Education"),
          custom.model.names = c(""),
          reorder.coef = c(2, 3, 1),
          lwd.vbars = 0)
```



Again, have a look at the help page via `?plotreg` to learn more about how this plot can be customized. If you want to set the size of the figure to avoid empty white space, you can print the figure into your working directory as usual using the `pdf()` (or `png()` etc.) functions. See Quick-R: Creating and Saving Graphs for more information.

```
pdf(file = "~/Documents/Uni/Teaching/POS 517/Tutorials/Day 8 - Multiple regression/plot_m1.pdf",
    width = 6, height = 3.5)
plotreg(list(mod),
          custom.coef.names = c("Intercept", "Income", "Education"),
          custom.model.names = c(""),
          reorder.coef = c(2, 3, 1),
          lwd.vbars = 0)
dev.off()
```

```
## pdf
##   2
```

Standardized regression coefficients

You already noticed one problem with the regression output that also carries over into the gregression coefficient dot plot: the estimate of the income coefficient is quite small compared to the education coefficient and the intercept. One potential benefit of multiple regression is that it allows to compare the relationships between explanatory variables and the outcome variable. In our current example, this requires some additional thinking because income and education are on different scales (dollars and years, respectively).

One solution that you encounter in section 5.2.4 of *AR* is to use standardized regression coefficients. The idea behind standardized coefficients is that they estimate relationships between variables that are all on a comparable scale. You already know one way to standardize variables (calcualating the z-score), and here we use a similar strategy. This strategy will yield the same results as the sequence you read about in section 5.2.4 of *AR*, but it standardizes *variables*, not coefficients:

- first, we rescale each variable so that its mean is 0 (we subtract the mean of the variable from each observation's value),
- second, we divide the rescaled variable by the standard deviation of the variable.

In R, this is easy to do by hand:

```
prestige.dat$prestige.std <- (prestige.dat$prestige -
                                mean(prestige.dat$prestige, na.rm = TRUE)) /
                              sd(prestige.dat$prestige, na.rm = TRUE)

prestige.dat$income.std <- (prestige.dat$income -
                              mean(prestige.dat$income, na.rm = TRUE)) /
                            sd(prestige.dat$income, na.rm = TRUE)

prestige.dat$education.std <- (prestige.dat$education -
                                 mean(prestige.dat$education, na.rm = TRUE)) /
                              sd(prestige.dat$education, na.rm = TRUE)

summary(prestige.dat)
```

```
##    education          income          women           prestige    
##  Min.   : 6.380   Min.   :  611   Min.   : 0.000   Min.   :14.80  
##  1st Qu.: 8.445   1st Qu.: 4106   1st Qu.: 3.592   1st Qu.:35.23  
##  Median :10.540   Median : 5930   Median :13.600   Median :43.60  
##  Mean   :10.738   Mean   : 6798   Mean   :28.979   Mean   :46.83  
##  3rd Qu.:12.648   3rd Qu.: 8187   3rd Qu.:52.203   3rd Qu.:59.27  
##  Max.   :15.970   Max.   :25879   Max.   :97.510   Max.   :87.20  
##      census       type     prestige.std       income.std     
##  Min.   :1113   bc  :44   Min.   :-1.8619   Min.   :-1.4571  
##  1st Qu.:3120   prof:31   1st Qu.:-0.6747   1st Qu.:-0.6340  
##  Median :5135   wc  :23   Median :-0.1879   Median :-0.2043  
##  Mean   :5402   NA's: 4   Mean   : 0.0000   Mean   : 0.0000  
##  3rd Qu.:8312             3rd Qu.: 0.7232   3rd Qu.: 0.3272  
##  Max.   :9517             Max.   : 2.3463   Max.   : 4.4940  
##  education.std     
##  Min.   :-1.59726  
##  1st Qu.:-0.84042  
##  Median :-0.07258  
##  Mean   : 0.00000  
##  3rd Qu.: 0.69983  
##  Max.   : 1.91756
```

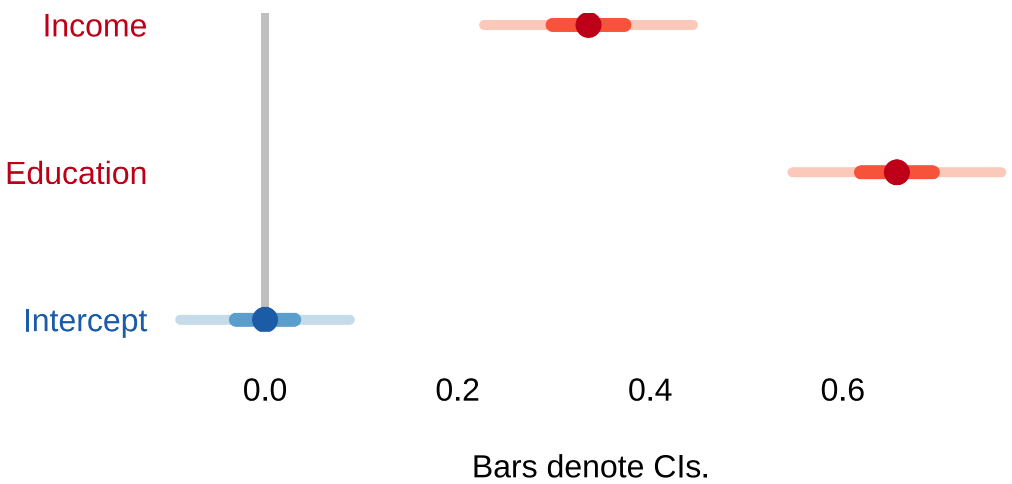
3. Fox warns you in *AR* (on pp. 95-96) that standardizing variables (and coefficients) is not useful if a variable is not normally distributed. Is any of the variables we just standardized not normally distributed? Is this reflected in the summary of the standardized variable?

We can now re-estimate our model using the rescaled variables, and have a look at the results:

```
mod.std <- lm(prestige.std ~ income.std + education.std, data = prestige.dat)
screenreg(list(mod.std),
          stars = 0.05,
          digits = 2,
          custom.model.names = c(""),
          custom.coef.names = c("Intercept", "Income", "Education"),
          reorder.coef = c(2, 3, 1))
```

```
## 
## ===================
##                    
## -------------------
## Income       0.34 *
##             (0.06) 
## Education    0.66 *
##             (0.06) 
## Intercept   -0.00  
##             (0.04) 
## -------------------
## R^2          0.80  
## Adj. R^2     0.79  
## Num. obs.  102     
## ===================
## * p < 0.05
```

```r
plotreg(list(mod.std),
          custom.coef.names = c("Intercept", "Income", "Education"),
          custom.model.names = c(""),
          reorder.coef = c(2, 3, 1),
          lwd.vbars = 0)
```



These results are now somewhat more helpful if we want to compare relationships between different explanatory variables and the response variable.

4. Is there a reason that the intercept is now estimated to be 0?
5. Did any other quantities of the model change compared to the initial model (`mod`) you fit? If yes, why? If not, why?

Note: R has a built-in function to standardize variables, `scale()`, so you do not need to perform the operation above each time you want to us standardized variables. The following code exploits this function:

```r
prestige.dat$prestige.std <- scale(prestige.dat$prestige)
prestige.dat$income.std <- scale(prestige.dat$income)
prestige.dat$education.std <- scale(prestige.dat$education)
```

`scale()` also allows you to only "center" a variable (set its mean to 0) or to only divide it by its standard deviation. Have a look at `?scale` for more information.

Lastly, you could use the `scale()` function directly in the model formula:

```r
mod.std <- lm(scale(prestige) ~ scale(income) + scale(education), data = prestige.dat)
```

Next steps

You have now learned the mechanics of fitting a linear regression model with multiple predictors in R. Adding additional predictors is trivial. Next, you should examine whether the assumptions of the linear regression model are met in your application. Multiple regression also brings some additional challenges that we will explore in the next few days, including collinearity, outliers, and heteroskedasticity. Lastly, you should always be aware of two issues. While you can fit linear regression models on almost any kind of data, the linear regression model often does not appropriately represent the underlying data generating process, and may therefore return biased and/or inefficient estimates. And, regression coefficients cannot be interpreted as causal effects unless your research design is set up as to clearly test for a causal relationship.