# Regression Basics*

Chapter 9

Anna Ziff

Minicourse: R and Workflow

## Contents



*Please contact anna.ziff@duke.edu if there are errors.

Here are the libraries you will need for this chapter.

```
library(AER)
library(dplyr)
library(ggplot2)
library(plm)
library(splines)
library(stargazer)
```

## Linear Regression

First, you may want to inspect the data graphically using `plot()` or `ggplot()`.

R has a `formula` class that is used in linear models. The tilde (~) in the below indicates that "`y` is explained by `x`."

```
f <- y ~ x
f
```

```
## y ~ x
```

```
class(f)
```

```
## [1] "formula"
```

This format is then used in regression functions.

```
x <- seq(0, 10, by = 0.5)
y <- 2 + 3 * x + rnorm(21)
lm(formula = y ~ x)
```

```
##
## Call:
## lm(formula = y ~ x)
##
## Coefficients:
## (Intercept)            x
##       2.598        2.977
```

This function is more usually used with a data set (data frames or tibbles are acceptable). We will use the small data frame `Journals` from the `AER` package for illustration. First, we inspect the data.

```
data("Journals", package = "AER")
jtib <- tibble(Journals)
```

Let us say we are interested in relating the log of price per citations and the log of the number of subscriptions. We prepare the data for this analysis.

```
jtib_analysis <- jtib %>%
    select(title, subs, price, citations) %>%
    mutate(citeprice = price / citations)

summary(jtib_analysis)
```

```
##     title                subs            price          citations
##  Length:180         Min.   :   2.0   Min.   :  20.0   Min.   :  21.00
##  Class :character   1st Qu.:  52.0   1st Qu.: 134.5   1st Qu.:  97.75
##  Mode  :character   Median : 122.5   Median : 282.0   Median : 262.50
##                     Mean   : 196.9   Mean   : 417.7   Mean   : 647.06
```

```
##                     3rd Qu.: 268.2   3rd Qu.: 540.8   3rd Qu.: 656.00
##                     Max.   :1098.0   Max.   :2120.0   Max.   :8999.00
##    citeprice
##  Min.   : 0.005223
##  1st Qu.: 0.464495
##  Median : 1.320513
##  Mean   : 2.548455
##  3rd Qu.: 3.440171
##  Max.   :24.459459
```

We are interested in estimating the parameters of the following equation.

$$\log(\texttt{subs}_i) = \beta_1 + \beta_2 \log(\texttt{citeprice}_i) + \varepsilon_i$$

Translating this equation into R's formula class is as follows. This is because the intercept is automatically included.

```
log(subs) ~ log(citeprice)
```

To then estimate the model, we insert this formula into `lm()`. The second argument specifies which dataset to use.

```
lm(log(subs) ~ log(citeprice), data = jtib_analysis)
```

```
##
## Call:
## lm(formula = log(subs) ~ log(citeprice), data = jtib_analysis)
##
## Coefficients:
##    (Intercept)  log(citeprice)
##         4.7662         -0.5331
```

```
lm(log(subs) ~ 1 + log(citeprice), data = jtib_analysis) # Make the intercept explicit
```

```
##
## Call:
## lm(formula = log(subs) ~ 1 + log(citeprice), data = jtib_analysis)
##
## Coefficients:
##    (Intercept)  log(citeprice)
##         4.7662         -0.5331
```

While the output tells us the estimated coefficients, we can get more information by defining the object to an output. The 12 components are now easily accessible from the object `out`, which acts as a list. Calling `summary()` on the object gives a lot more information than the usual print-out.

```
out <- lm(log(subs) ~ log(citeprice), data = jtib_analysis)
class(out)
```

```
## [1] "lm"
```

```
names(out)
```

```
##  [1] "coefficients"  "residuals"     "effects"       "rank"
##  [5] "fitted.values" "assign"        "qr"            "df.residual"
##  [9] "xlevels"       "call"          "terms"         "model"
```

```
out$coefficients
```

```
##    (Intercept) log(citeprice)
##      4.7662121     -0.5330535
```

```
summary(out)
```

```
##
## Call:
## lm(formula = log(subs) ~ log(citeprice), data = jtib_analysis)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -2.72478 -0.53609  0.03721  0.46619  1.84808
##
## Coefficients:
##                Estimate Std. Error t value Pr(>|t|)
## (Intercept)     4.76621    0.05591   85.25   <2e-16 ***
## log(citeprice) -0.53305    0.03561  -14.97   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.7497 on 178 degrees of freedom
## Multiple R-squared:  0.5573, Adjusted R-squared:  0.5548
## F-statistic:   224 on 1 and 178 DF,  p-value: < 2.2e-16
```

The summarized object can be saved as well!

```
out_sum <- summary(out)
class(out_sum)
```

```
## [1] "summary.lm"
```

```
names(out_sum)
```

```
##  [1] "call"          "terms"         "residuals"     "coefficients"
##  [5] "aliased"       "sigma"         "df"            "r.squared"
##  [9] "adj.r.squared" "fstatistic"    "cov.unscaled"
```

Here are more functions that can be applied to `lm` objects.

```
# Print the object
print(out)

# Coefficients of the regression
coef(out)

# Residuals
residuals(out) # Alternatively, resid(out)

# Fitted values
fitted(out)

# ANOVA (helpful for F-tests)
anova(out)

# Predict (same as fitted() here but more flexible to calculate standard errors)
predict(out)
```

```r
# Confidence Interval
confint(out)
```

```
##                     2.5 %     97.5 %
## (Intercept)     4.6558822  4.8765420
## log(citeprice) -0.6033319 -0.4627751
```

```r
confint(out, level = 0.9)
```

```
##                      5 %       95 %
## (Intercept)     4.6737688  4.8586555
## log(citeprice) -0.5919384 -0.4741685
```

```r
# Residual Sum of Squares
deviance(out)
```

```
## [1] 100.0561
```

```r
# Variance-Covariance Matrix
vcov(out)
```

```
##                   (Intercept) log(citeprice)
## (Intercept)     3.125825e-03  -6.144352e-05
## log(citeprice) -6.144352e-05   1.268300e-03
```

```r
# Log-likelihood
logLik(out)
```

```
## 'log Lik.' -202.5586 (df=3)
```

```r
# Information criteria
AIC(out)
```

```
## [1] 411.1171
```

```r
# Hypothesis test
linearHypothesis(out, "log(citeprice) = -0.5")
```

```
## Linear hypothesis test
##
## Hypothesis:
## log(citeprice) = - 0.5
##
## Model 1: restricted model
## Model 2: log(subs) ~ log(citeprice)
##
##   Res.Df    RSS Df Sum of Sq      F Pr(>F)
## 1    179 100.54
## 2    178 100.06  1   0.48421 0.8614 0.3546
```

```r
linearHypothesis(out, hypothesis.matrix = c(0, 1), rhs = -0.5)
```

```
## Linear hypothesis test
##
## Hypothesis:
## log(citeprice) = - 0.5
##
## Model 1: restricted model
## Model 2: log(subs) ~ log(citeprice)
##
```

```
##   Res.Df    RSS Df Sum of Sq      F Pr(>F)
## 1    179 100.54
## 2    178 100.06  1   0.48421 0.8614 0.3546
```

Now that we are introduced to the basic functions available for linear regression, we can explore more complicated models.

```
data("CPS1988", package = "AER")
cps <- tibble(CPS1988)
summary(cps)
```

```
##       wage            education       experience   ethnicity     smsa
##  Min.   :   50.05   Min.   : 0.00   Min.   :-4.0   cauc:25923   no : 7223
##  1st Qu.:  308.64   1st Qu.:12.00   1st Qu.: 8.0   afam: 2232   yes:20932
##  Median :  522.32   Median :12.00   Median :16.0
##  Mean   :  603.73   Mean   :13.07   Mean   :18.2
##  3rd Qu.:  783.48   3rd Qu.:15.00   3rd Qu.:27.0
##  Max.   :18777.20   Max.   :18.00   Max.   :63.0
##        region     parttime
##  northeast:6441   no :25631
##  midwest  :6863   yes: 2524
##  south    :8760
##  west     :6091
##
##
```

Let us fit the Mincer equation. The function `I()` ensures that the squared term is interpreted mathematically rather than a formula specification. It is clear that `+` does not mean arithmetic addition in the formula class. Similarly, the asterisk, colon, forward slash, and carrot symbol have formula-specific meanings. If you need to use them in another capacity, they must be inside `I()`.

```
mincer <- lm(log(wage) ~ experience + I(experience^2) + education + ethnicity, data = cps)
summary(mincer)
```

```
##
## Call:
## lm(formula = log(wage) ~ experience + I(experience^2) + education +
##     ethnicity, data = cps)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -2.9428 -0.3162  0.0580  0.3756  4.3830
##
## Coefficients:
##                    Estimate Std. Error t value Pr(>|t|)
## (Intercept)       4.321e+00  1.917e-02  225.38   <2e-16 ***
## experience        7.747e-02  8.800e-04   88.03   <2e-16 ***
## I(experience^2) -1.316e-03  1.899e-05  -69.31   <2e-16 ***
## education         8.567e-02  1.272e-03   67.34   <2e-16 ***
## ethnicityafam    -2.434e-01  1.292e-02  -18.84   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.5839 on 28150 degrees of freedom
## Multiple R-squared:  0.3347, Adjusted R-squared:  0.3346
## F-statistic:  3541 on 4 and 28150 DF,  p-value: < 2.2e-16
```

The variable `ethnicity` is a factor with two levels, "cauc"and "afam." The formula call above automatically drops all but the first level.

```
table(cps$ethnicity)
```

```
##
##  cauc  afam
## 25923  2232
```

To change the level, use `relevel()`.

```
cps <- cps %>%
    mutate(ethnicity = relevel(ethnicity, ref = 2))
lm(log(wage) ~ experience + I(experience^2) + education + ethnicity, data = cps) %>%
    summary()
```

```
##
## Call:
## lm(formula = log(wage) ~ experience + I(experience^2) + education +
##     ethnicity, data = cps)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -2.9428 -0.3162  0.0580  0.3756  4.3830
##
## Coefficients:
##                   Estimate Std. Error t value Pr(>|t|)
## (Intercept)      4.078e+00  2.180e-02  187.09   <2e-16 ***
## experience       7.747e-02  8.800e-04   88.03   <2e-16 ***
## I(experience^2) -1.316e-03  1.899e-05  -69.31   <2e-16 ***
## education        8.567e-02  1.272e-03   67.34   <2e-16 ***
## ethnicitycauc    2.434e-01  1.292e-02   18.84   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.5839 on 28150 degrees of freedom
## Multiple R-squared:  0.3347, Adjusted R-squared:  0.3346
## F-statistic:  3541 on 4 and 28150 DF,  p-value: < 2.2e-16
```

Setting the coefficient to 0 is simple with the formula specification.

```
lm(log(wage) ~ experience + I(experience^2) + education + ethnicity - 1, data = cps) %>%
    summary()
```

```
##
## Call:
## lm(formula = log(wage) ~ experience + I(experience^2) + education +
##     ethnicity - 1, data = cps)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -2.9428 -0.3162  0.0580  0.3756  4.3830
##
## Coefficients:
##                   Estimate Std. Error t value Pr(>|t|)
## experience       7.747e-02  8.800e-04   88.03   <2e-16 ***
## I(experience^2) -1.316e-03  1.899e-05  -69.31   <2e-16 ***
```

```
## education        8.567e-02  1.272e-03   67.34   <2e-16 ***
## ethnicityafam    4.078e+00  2.180e-02  187.09   <2e-16 ***
## ethnicitycauc    4.321e+00  1.917e-02  225.38   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.5839 on 28150 degrees of freedom
## Multiple R-squared:  0.9912, Adjusted R-squared:  0.9912
## F-statistic: 6.316e+05 on 5 and 28150 DF,  p-value: < 2.2e-16
```

Adding interactions is done with `:` without specifying the main effects. With `*`, the interactions and main effects are included.

```
# No main effects
lm(log(wage) ~ experience + I(experience^2) + education : ethnicity, data = cps) %>%
    summary()
```

```
##
## Call:
## lm(formula = log(wage) ~ experience + I(experience^2) + education:ethnicity,
##     data = cps)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -2.9470 -0.3158  0.0580  0.3752  4.4036
##
## Coefficients:
##                            Estimate Std. Error t value Pr(>|t|)
## (Intercept)               4.304e+00  1.908e-02  225.51   <2e-16 ***
## experience                7.757e-02  8.800e-04   88.14   <2e-16 ***
## I(experience^2)          -1.320e-03  1.899e-05  -69.50   <2e-16 ***
## education:ethnicityafam   6.781e-02  1.642e-03   41.30   <2e-16 ***
## education:ethnicitycauc   8.699e-02  1.269e-03   68.55   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.5839 on 28150 degrees of freedom
## Multiple R-squared:  0.3347, Adjusted R-squared:  0.3346
## F-statistic:  3541 on 4 and 28150 DF,  p-value: < 2.2e-16
```

```
# Main effects
lm(log(wage) ~ experience + I(experience^2) + education * ethnicity, data = cps) %>%
    summary()
```

```
##
## Call:
## lm(formula = log(wage) ~ experience + I(experience^2) + education *
##     ethnicity, data = cps)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -2.9451 -0.3162  0.0578  0.3761  4.3929
##
## Coefficients:
##                            Estimate Std. Error t value Pr(>|t|)
## (Intercept)               4.189e+00  5.784e-02  72.426   <2e-16 ***
```

```
## experience                7.752e-02  8.803e-04  88.063   <2e-16 ***
## I(experience^2)          -1.318e-03  1.901e-05 -69.339   <2e-16 ***
## education                 7.666e-02  4.525e-03  16.941   <2e-16 ***
## ethnicitycauc             1.239e-01  5.903e-02   2.099   0.0358 *
## education:ethnicitycauc   9.648e-03  4.651e-03   2.074   0.0380 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.5839 on 28149 degrees of freedom
## Multiple R-squared:  0.3348, Adjusted R-squared:  0.3347
## F-statistic:  2834 on 5 and 28149 DF,  p-value: < 2.2e-16
```

Use the forward slash to fit separate regressions.

```
lm(log(wage) ~ ethnicity / (experience + I(experience^2) + education), data = cps) %>%
    summary()
```

```
##
## Call:
## lm(formula = log(wage) ~ ethnicity/(experience + I(experience^2) +
##     education), data = cps)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -2.9353 -0.3168  0.0571  0.3754  4.4418
##
## Coefficients:
##                                 Estimate Std. Error t value Pr(>|t|)
## (Intercept)                    4.159e+00  7.330e-02  56.743   <2e-16 ***
## ethnicitycauc                  1.509e-01  7.592e-02   1.987   0.0469 *
## ethnicityafam:experience       6.190e-02  2.940e-03  21.055   <2e-16 ***
## ethnicitycauc:experience       7.923e-02  9.231e-04  85.838   <2e-16 ***
## ethnicityafam:I(experience^2) -9.415e-04  6.069e-05 -15.513   <2e-16 ***
## ethnicitycauc:I(experience^2) -1.360e-03  2.001e-05 -67.966   <2e-16 ***
## ethnicityafam:education        8.654e-02  5.099e-03  16.974   <2e-16 ***
## ethnicitycauc:education        8.575e-02  1.313e-03  65.293   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.5835 on 28147 degrees of freedom
## Multiple R-squared:  0.3359, Adjusted R-squared:  0.3357
## F-statistic:  2034 on 7 and 28147 DF,  p-value: < 2.2e-16
```

Splines can easily be estimated for semiparametric models. Suppose instead of specifying a second-order polynomial for experience, we want to have a more flexible specification.

$$\log(\texttt{wage}_i) = \beta_1 + g(\texttt{experience}_i) + \beta_2 \texttt{education}_i + \beta_3 \texttt{ethnicity}_i + \varepsilon_i$$

The function `bs()` generates B splices with a specified number of degrees of freedom. See the documentation for alternative ways to specify the splines. The package `np` allows for kernel methods.

```
mincer_plm <- lm(log(wage) ~ bs(experience, df = 5) + education + ethnicity, data = cps)
```

Weights can be added to the model with the `weights` argument.

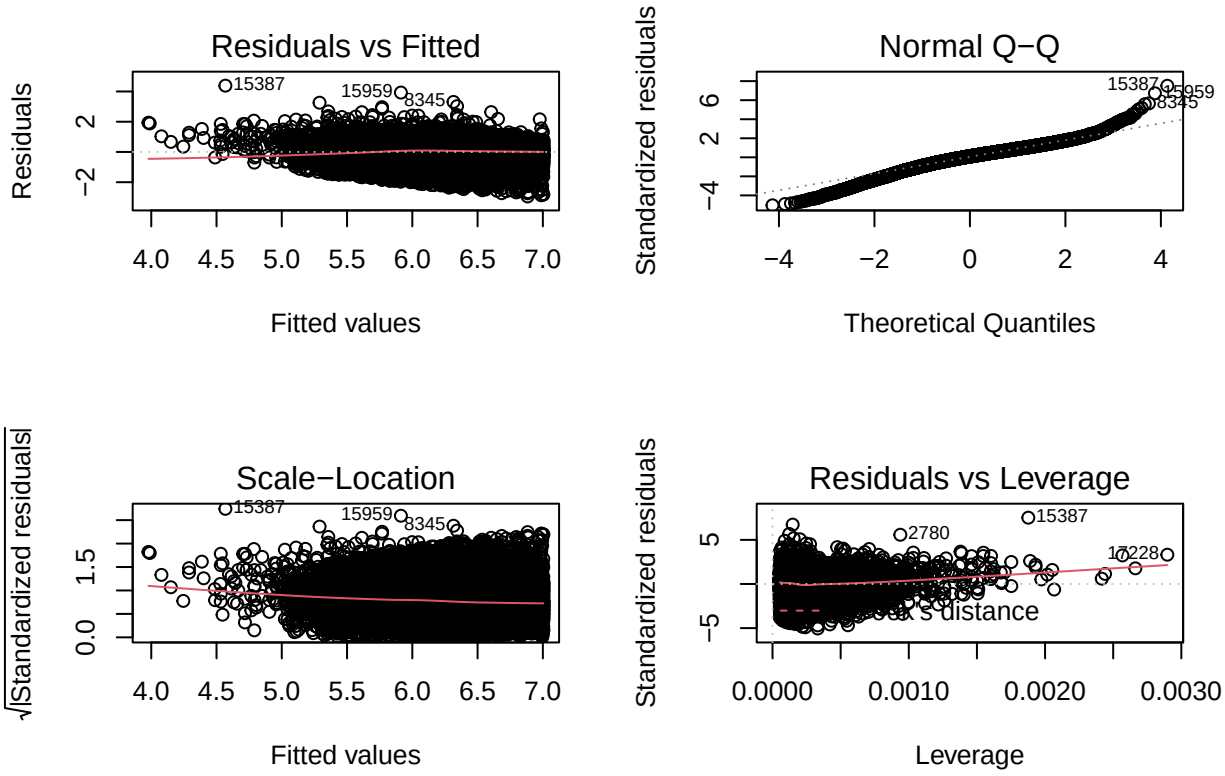
```
lm(log(subs) ~ log(citeprice), data = jtib_analysis, weights = 1/citeprice^2) %>%
    summary()
```

```
##
## Call:
## lm(formula = log(subs) ~ log(citeprice), data = jtib_analysis,
##     weights = 1/citeprice^2)
##
## Weighted Residuals:
##      Min       1Q   Median       3Q      Max
## -27.9773  -0.4632  -0.1428   0.2532  16.8990
##
## Coefficients:
##                Estimate Std. Error t value Pr(>|t|)
## (Intercept)     5.11053    0.08554   59.75   <2e-16 ***
## log(citeprice) -0.35881    0.01760  -20.39   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 3.46 on 178 degrees of freedom
## Multiple R-squared:  0.7002, Adjusted R-squared:  0.6985
## F-statistic: 415.8 on 1 and 178 DF,  p-value: < 2.2e-16
```

## Evaluate Assumptions

The `lm` object can be plotted with the built-in `plot()` function. This will automatically run some diagnostic plots. The function `par()` shows the plots at the same time.

```
mincer <- lm(log(wage) ~ experience + I(experience^2) + education + ethnicity, data = cps)
par(mfrow = c(2, 2))
plot(mincer)
```




```
par(mfrow = c(1, 1)) # Set the output back to 1 plot at a time
```

The function `anova()` can be helpful in comparing the residual sum of squares of nested models.

```
mincer_noeth <- lm(log(wage) ~ experience + I(experience^2) + education, data = cps)
anova(mincer_noeth, mincer)
```

```
## Analysis of Variance Table
##
## Model 1: log(wage) ~ experience + I(experience^2) + education
## Model 2: log(wage) ~ experience + I(experience^2) + education + ethnicity
##   Res.Df    RSS Df Sum of Sq      F    Pr(>F)
## 1  28151 9719.6
## 2  28150 9598.6  1    121.02 354.91 < 2.2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

The function `update()` is a more elegant way to estimate nested models.

```
mincer_noeth <- update(mincer, formula = . ~ . - ethnicity)
```

This can all be combined in the function `waldtest()`. The F-test is the same as above, but the residual sums of square are not printed. It is possible to specify other types of tests (e.g., quasi-F tests).

```
waldtest(mincer, . ~ . - ethnicity)
```

```
## Wald test
##
## Model 1: log(wage) ~ experience + I(experience^2) + education + ethnicity
## Model 2: log(wage) ~ experience + I(experience^2) + education
##   Res.Df Df      F    Pr(>F)
## 1  28150
```

```
## 2  28151 -1 354.91 < 2.2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

If there heteroskedasticity, the functions `vcovHC()` and `coeftest()` can be combined.

```
robustvcov <- vcovHC(mincer, type = "HC") # Variance-Covariance matrix
robustse <- diag(vcovHC(mincer, type = "HC"))
coeftest(mincer, vcov. = robustvcov)
```

```
##
## t test of coefficients:
##
##                     Estimate  Std. Error t value  Pr(>|t|)
## (Intercept)      4.0780e+00  2.3122e-02 176.372 < 2.2e-16 ***
## experience       7.7473e-02  1.0183e-03  76.085 < 2.2e-16 ***
## I(experience^2) -1.3161e-03  2.3472e-05 -56.071 < 2.2e-16 ***
## education        8.5673e-02  1.3750e-03  62.307 < 2.2e-16 ***
## ethnicitycauc    2.4336e-01  1.3112e-02  18.561 < 2.2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

## Fixed Effects

There are several different approaches to include fixed effects in regressions. To demonstrate, we will use the sample data `Produc` from the `plm` package. This data frame contains a panel of U.S. production for 48 states between 1970 and 1986. Recall that you can always type `?Produc` to read a description of sample data.

```
data("Produc", package = "plm")
```

Suppose we want to estimate the following regression with state fixed effects,

$$\log(\text{GSP}_{it}) = \beta_0 + \beta_1 \log(\text{Pub. Capital Stock})_{it} + \beta_2 \log(\text{Pri. Capital Stock})_{it} + \beta_3 \log(\text{Emp.}) + \beta_4 \text{Unemp.} + \text{State}_i + \varepsilon_{it}$$

Because `state` is a factor variable, we can simply add it to the formula and R will estimate fixed effects in relation to the first level.

```
lm(log(gsp) ~ log(pcap) + log(pc) + log(emp) + unemp + state, data = Produc) %>%
  summary()
```

If we try to do that with `year` to add year fixed effects, R would treat it as a number rather than a fixed effect. We can add `factor()` to cast it as a factor variable.

```
lm(log(gsp) ~  log(pcap) + log(pc) + log(emp) + unemp  + state + factor(year), data = Produc) %>%
  summary()
```

Recall that you can remove the intercept if you would like to remove the intercept and include all the factors (depending on your degrees of freedom).

```
lm(log(gsp) ~ log(pcap) + log(pc) + log(emp) + unemp + factor(year) - 1, data = Produc) %>%
  summary()
```

```
##
## Call:
## lm(formula = log(gsp) ~ log(pcap) + log(pc) + log(emp) + unemp +
##     factor(year) - 1, data = Produc)
##
## Residuals:
```

```
##      Min       1Q   Median       3Q      Max
## -0.22355 -0.05832 -0.00135  0.04898  0.35817
##
## Coefficients:
##                   Estimate Std. Error t value Pr(>|t|)
## log(pcap)         0.164780   0.017491   9.421  < 2e-16 ***
## log(pc)           0.303596   0.010443  29.073  < 2e-16 ***
## log(emp)          0.588811   0.013776  42.743  < 2e-16 ***
## unemp            -0.006057   0.001770  -3.422 0.000653 ***
## factor(year)1970  1.630510   0.058141  28.044  < 2e-16 ***
## factor(year)1971  1.636502   0.058509  27.970  < 2e-16 ***
## factor(year)1972  1.647834   0.058431  28.201  < 2e-16 ***
## factor(year)1973  1.658358   0.058308  28.441  < 2e-16 ***
## factor(year)1974  1.625408   0.058438  27.814  < 2e-16 ***
## factor(year)1975  1.618653   0.059113  27.382  < 2e-16 ***
## factor(year)1976  1.621605   0.058929  27.518  < 2e-16 ***
## factor(year)1977  1.629921   0.058699  27.768  < 2e-16 ***
## factor(year)1978  1.638746   0.058328  28.096  < 2e-16 ***
## factor(year)1979  1.633775   0.058253  28.046  < 2e-16 ***
## factor(year)1980  1.619712   0.058557  27.660  < 2e-16 ***
## factor(year)1981  1.632509   0.058757  27.784  < 2e-16 ***
## factor(year)1982  1.624689   0.059449  27.329  < 2e-16 ***
## factor(year)1983  1.639832   0.059470  27.574  < 2e-16 ***
## factor(year)1984  1.661174   0.058789  28.256  < 2e-16 ***
## factor(year)1985  1.668949   0.058685  28.439  < 2e-16 ***
## factor(year)1986  1.676169   0.058717  28.547  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.08733 on 795 degrees of freedom
## Multiple R-squared:  0.9999, Adjusted R-squared:  0.9999
## F-statistic: 5.68e+05 on 21 and 795 DF,  p-value: < 2.2e-16
```

Additionally, you could demean your independent and dependent variables to get the same coefficient estimates and standard errors. If you have a really large dataset, this could ease the computational burden of explicitly estimating many fixed effects.

```
produc_demean <- Produc %>%
  mutate(across(c("gsp", "pcap", "pc", "emp"), log)) %>%
  group_by(year) %>%
  mutate(across(c("gsp", "pcap", "pc", "emp", "unemp"),
                ~ .x - mean(.x)))

lm(gsp ~ pcap + pc + emp + unemp - 1, data = produc_demean) %>%
  summary()
```

```
##
## Call:
## lm(formula = gsp ~ pcap + pc + emp + unemp - 1, data = produc_demean)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.22355 -0.05832 -0.00135  0.04898  0.35817
##
## Coefficients:
```

```
##        Estimate Std. Error t value Pr(>|t|)
## pcap   0.164780   0.017307   9.521  < 2e-16 ***
## pc     0.303596   0.010333  29.382  < 2e-16 ***
## emp    0.588811   0.013631  43.197  < 2e-16 ***
## unemp -0.006057   0.001752  -3.458 0.000572 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.08641 on 812 degrees of freedom
## Multiple R-squared:  0.9927, Adjusted R-squared:  0.9927
## F-statistic: 2.774e+04 on 4 and 812 DF,  p-value: < 2.2e-16
```

Another option is to use `plm`, which is a widely used and reliable package. The `plm` function transforms the data and implements `lm`. The standard errors are robust, which is why they differ from what we got using `lm` directly.

```
plm(log(gsp) ~ log(pcap) + log(pc) + log(emp) + unemp - 1,
    data = Produc,
    index = "year",
    model = "within") %>%
  summary()
```

```
## Oneway (individual) effect Within Model
##
## Call:
## plm(formula = log(gsp) ~ log(pcap) + log(pc) + log(emp) + unemp -
##     1, data = Produc, model = "within", index = "year")
##
## Balanced Panel: n = 17, T = 48, N = 816
##
## Residuals:
##       Min.    1st Qu.     Median    3rd Qu.       Max.
## -0.2235495 -0.0583250 -0.0013538  0.0489805  0.3581747
##
## Coefficients:
##             Estimate Std. Error t-value  Pr(>|t|)
## log(pcap)  0.1647800  0.0174912  9.4207 < 2.2e-16 ***
## log(pc)    0.3035960  0.0104427 29.0727 < 2.2e-16 ***
## log(emp)   0.5888107  0.0137757 42.7428 < 2.2e-16 ***
## unemp     -0.0060575  0.0017702 -3.4220 0.0006534 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Total Sum of Squares:    834.46
## Residual Sum of Squares: 6.0629
## R-Squared:      0.99273
## Adj. R-Squared: 0.99255
## F-statistic: 27156 on 4 and 795 DF, p-value: < 2.22e-16
```

It is simple to add another set of fixed effects. See the documentation for additional complications or choices you could make.

```
plm(log(gsp) ~ log(pcap) + log(pc) + log(emp) + unemp,
    data = Produc,
    index = c("state", "year"),
    model = "within") %>%
```

```
  summary()
```

# Display Results

The below offers some **very** basic examples. In practice, you can combine all of your knowledge of graphing and creating tables to output the exact figures and tables you wish. The statistics, residuals, fitted values, etc. are readily available and be combined as needed.

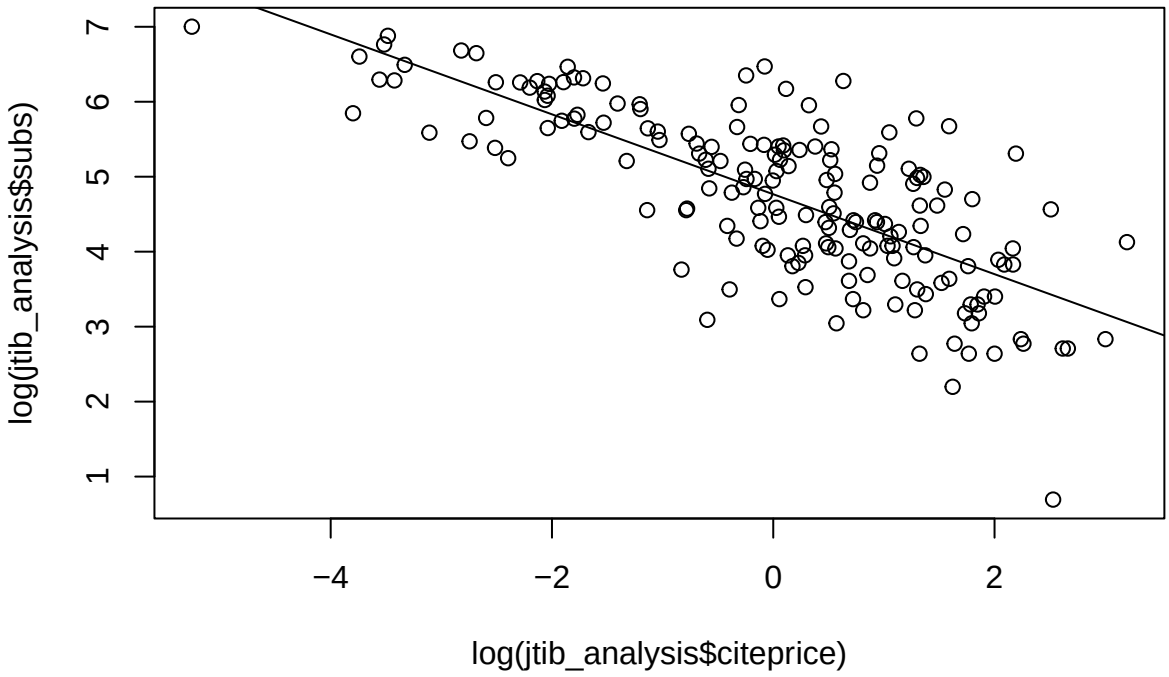
## Graphs

Results of regressions can be displayed with the built-in graphing functions using the `abline()` function

```
summary(out)
```

```
##
## Call:
## lm(formula = log(subs) ~ log(citeprice), data = jtib_analysis)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -2.72478 -0.53609  0.03721  0.46619  1.84808
##
## Coefficients:
##                Estimate Std. Error t value Pr(>|t|)
## (Intercept)     4.76621    0.05591   85.25   <2e-16 ***
## log(citeprice) -0.53305    0.03561  -14.97   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.7497 on 178 degrees of freedom
## Multiple R-squared:  0.5573, Adjusted R-squared:  0.5548
## F-statistic:   224 on 1 and 178 DF,  p-value: < 2.2e-16
```

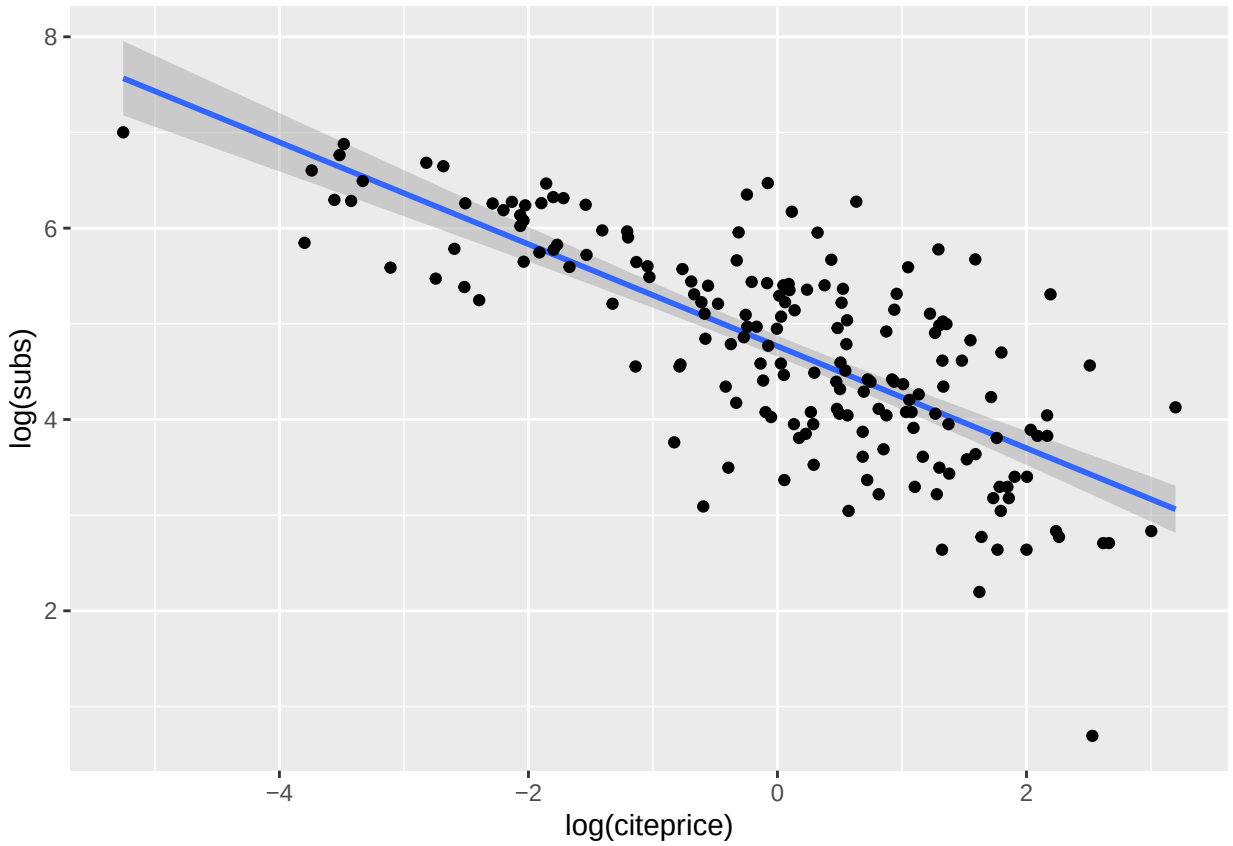
```
# Uncomment the lines below to save the plot
# jpeg("Journals.jpg") # Save plot
plot(log(jtib_analysis$citeprice), log(jtib_analysis$subs))
abline(out)
```



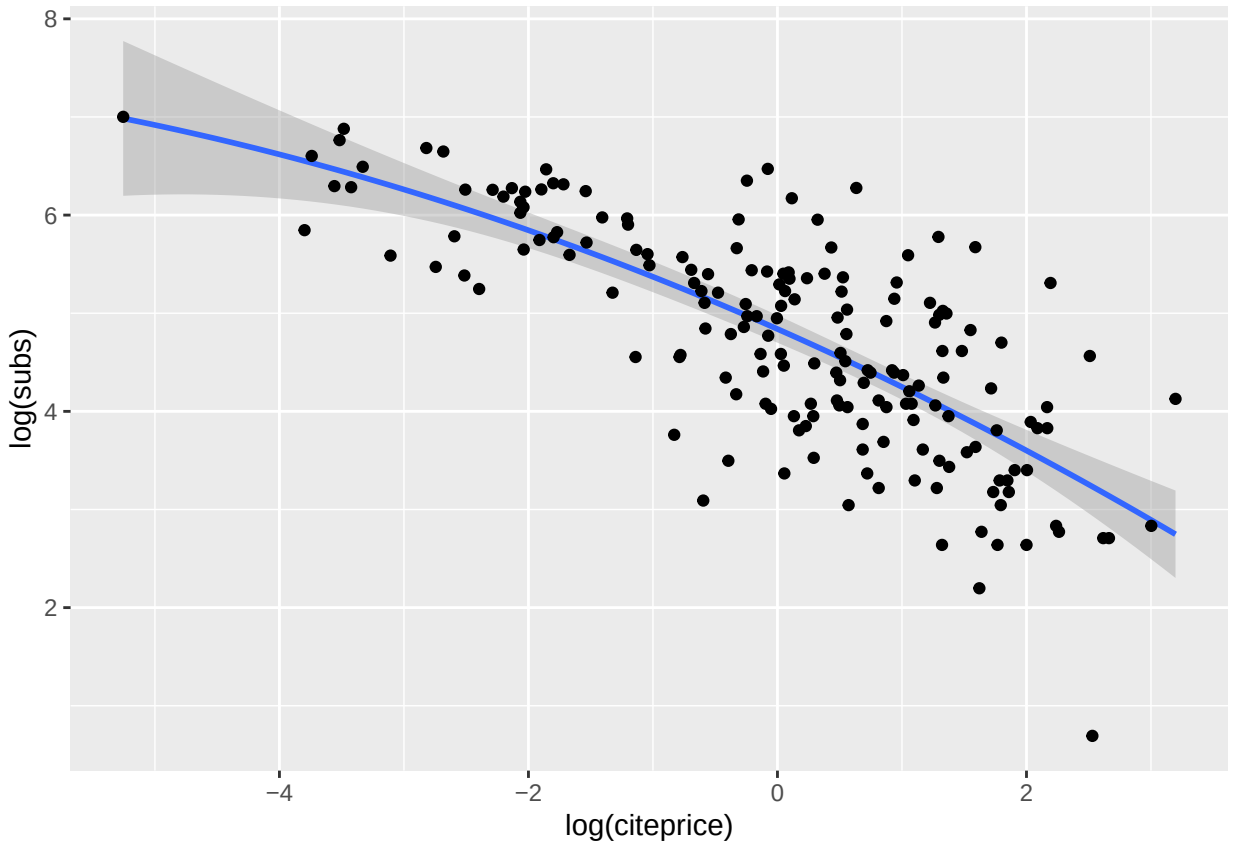

```
# dev.off() # Close the device of the saved plot
```

Recall that `geom_smooth()` is a useful layer for `ggplot2` plots. Note that the formula can be altered.

```
ggplot(jtib_analysis, aes(x = log(citeprice), y = log(subs))) +
  geom_smooth(method = "lm", formula = y ~ x) +
  geom_point()
```




```
ggplot(jtib_analysis, aes(x = log(citeprice), y = log(subs))) +
  geom_smooth(method = "lm", formula = y ~ x + I(x^2)) +
  geom_point()
```




## Tables

The package **stargazer** has many options to display regression results in tables.

```
stargazer(out)
```

```
##
## % Table created by stargazer v.5.2.2 by Marek Hlavac, Harvard University. E-mail: hlavac at fas.harva
## % Date and time: Tue, Aug 02, 2022 - 15:11:24
## \begin{table}[!htbp] \centering
##   \caption{}
##   \label{}
## \begin{tabular}{@{\extracolsep{5pt}}lc}
## \\[-1.8ex]\hline
## \hline \\[-1.8ex]
##  & \multicolumn{1}{c}{\textit{Dependent variable:}} \\
## \cline{2-2}
## \\[-1.8ex] & log(subs) \\
## \hline \\[-1.8ex]
##  log(citeprice) & $-$0.533$^{***}$ \\
##   & (0.036) \\
##   & \\
##  Constant & 4.766$^{***}$ \\
##   & (0.056) \\
##   & \\
## \hline \\[-1.8ex]
## Observations & 180 \\
```

```
## R$^{2}$ & 0.557 \\
## Adjusted R$^{2}$ & 0.555 \\
## Residual Std. Error & 0.750 (df = 178) \\
## F Statistic & 224.037$^{***}$ (df = 1; 178) \\
## \hline
## \hline \\[-1.8ex]
## \textit{Note:}  & \multicolumn{1}{r}{$^{*}$p$<$0.1; $^{**}$p$<$0.05; $^{***}$p$<$0.01} \\
## \end{tabular}
## \end{table}
```

More than one model can be included.

```
stargazer(mincer, mincer_noeth)
```

```
##
## % Table created by stargazer v.5.2.2 by Marek Hlavac, Harvard University. E-mail: hlavac at fas.harva
## % Date and time: Tue, Aug 02, 2022 - 15:11:24
## \begin{table}[!htbp] \centering
##   \caption{}
##   \label{}
## \begin{tabular}{@{\extracolsep{5pt}}lcc}
## \\[-1.8ex]\hline
## \hline \\[-1.8ex]
##  & \multicolumn{2}{c}{\textit{Dependent variable:}} \\
## \cline{2-3}
## \\[-1.8ex] & \multicolumn{2}{c}{log(wage)} \\
## \\[-1.8ex] & (1) & (2)\\
## \hline \\[-1.8ex]
##  experience & 0.077$^{***}$ & 0.078$^{***}$ \\
##   & (0.001) & (0.001) \\
##   & & \\
##  I(experience$\hat{\mkern6mu}$2) & $-$0.001$^{***}$ & $-$0.001$^{***}$ \\
##   & (0.00002) & (0.00002) \\
##   & & \\
##  education & 0.086$^{***}$ & 0.087$^{***}$ \\
##   & (0.001) & (0.001) \\
##   & & \\
##  ethnicitycauc & 0.243$^{***}$ &  \\
##   & (0.013) &  \\
##   & & \\
##  Constant & 4.078$^{***}$ & 4.278$^{***}$ \\
##   & (0.022) & (0.019) \\
##   & & \\
## \hline \\[-1.8ex]
## Observations & 28,155 & 28,155 \\
## R$^{2}$ & 0.335 & 0.326 \\
## Adjusted R$^{2}$ & 0.335 & 0.326 \\
## Residual Std. Error & 0.584 (df = 28150) & 0.588 (df = 28151) \\
## F Statistic & 3,541.036$^{***}$ (df = 4; 28150) & 4,545.929$^{***}$ (df = 3; 28151) \\
## \hline
## \hline \\[-1.8ex]
## \textit{Note:}  & \multicolumn{2}{r}{$^{*}$p$<$0.1; $^{**}$p$<$0.05; $^{***}$p$<$0.01} \\
## \end{tabular}
## \end{table}
```

The different components of the output can be controlled with the various options.

```
stargazer(mincer, mincer_noeth,
          covariate.labels = c("Experience", "Experience$^2$",
                               "Education", "African American",
                               "Constant"),
          ci.level = 0.90)
```

```
##
## % Table created by stargazer v.5.2.2 by Marek Hlavac, Harvard University. E-mail: hlavac at fas.harva
## % Date and time: Tue, Aug 02, 2022 - 15:11:24
## \begin{table}[!htbp] \centering
##   \caption{}
##   \label{}
## \begin{tabular}{@{\extracolsep{5pt}}lcc}
## \\[-1.8ex]\hline
## \hline \\[-1.8ex]
##  & \multicolumn{2}{c}{\textit{Dependent variable:}} \\
## \cline{2-3}
## \\[-1.8ex] & \multicolumn{2}{c}{log(wage)} \\
## \\[-1.8ex] & (1) & (2)\\
## \hline \\[-1.8ex]
##  Experience & 0.077$^{***}$ & 0.078$^{***}$ \\
##   & (0.001) & (0.001) \\
##   & & \\
##  Experience$^2$ & $-$0.001$^{***}$ & $-$0.001$^{***}$ \\
##   & (0.00002) & (0.00002) \\
##   & & \\
##  Education & 0.086$^{***}$ & 0.087$^{***}$ \\
##   & (0.001) & (0.001) \\
##   & & \\
##  African American & 0.243$^{***}$ &  \\
##   & (0.013) &  \\
##   & & \\
##  Constant & 4.078$^{***}$ & 4.278$^{***}$ \\
##   & (0.022) & (0.019) \\
##   & & \\
## \hline \\[-1.8ex]
## Observations & 28,155 & 28,155 \\
## R$^{2}$ & 0.335 & 0.326 \\
## Adjusted R$^{2}$ & 0.335 & 0.326 \\
## Residual Std. Error & 0.584 (df = 28150) & 0.588 (df = 28151) \\
## F Statistic & 3,541.036$^{***}$ (df = 4; 28150) & 4,545.929$^{***}$ (df = 3; 28151) \\
## \hline
## \hline \\[-1.8ex]
## \textit{Note:}  & \multicolumn{2}{r}{$^{*}$p$<$0.1; $^{**}$p$<$0.05; $^{***}$p$<$0.01} \\
## \end{tabular}
## \end{table}
```

## Caution!

It is always important to vet packages and make sure that they are reliable before using them in your code. Here are some questions you might try to answer qualitatively.

- Is the package updated frequently?

- Is the package widely cited as reliable on StackExchange or other help forums? Or, does it seem to have a lot of errors and forums present workarounds?
- Does the creator seem to respond to Issues on the GitHub quickly?

Vetting packages is especially important for those used in your main analysis. Some economists who publish their own estimators will also publish an R package. If the package is reliable, then this is wonderful. You can rely on their expertise and rigorous testing procedures to ensure that the results are calculated correctly *and* save yourself many hours of coding. However, if the package is not reliable, then your analysis will be wrong. Even if you deem a package to be reliable, keeping an eye on the package's GitHub page can help keep you apprised of updates and corrections.

## Further Reading

The information in the above chapter comes from Kleiber and Zeileis (2008). See chapters 5 and 6 of that textbook for more examples of micro and macro methods. This list is a great resource for other topics, including IV regression.

### References

Kleiber, Christian, and Achim Zeileis. 2008. *Applied Econometrics with R*. Use R! New York, NY: Springer.