`knitr` intro and OLS

Kendon Bell

February 5, 2015

The objective of this section is twofold: 1) to briefly introduce you to `knitr` for producing documents, and 2) to produce our first OLS estimate using matrices.

At the highest level, `knitr`, is an `R` package that takes a text document and produces code for a presentable format. The power of `knitr` is its ability to generate a dynamic document which will update easily with new data or small changes to specifications. You will never have to manually change the numbers in a table again! This is another tool which will help you to be a more productive researcher.

In this section, we will be using `knitr` to take an `.Rnw` file and ultimately produce a `.pdf` file, via LaTeX. We will be using the OLS estimator to illustrate the use of loops and some matrix operations in `R`.

For a complete description of all `knitr` options, see:

`http://yihui.name/knitr/options/`

Getting started with `knitr` and LaTeX

To begin in RStudio, we need to make sure we have a distribution of LaTeX installed. You can download one of these for your operating system from:

`http://latex-project.org/ftp.html`

In RStudio, we need to set some options. Go to Tools → Global Options → Sweave. Then change "Weave Rnw files using:" to `knitr` and "Typeset LaTeX into pdf using:" to `pdfLatex`. Windows users should install `Sumatra` for viewing pdfs. This can be downloaded from:

`http://www.sumatrapdfreader.org/free-pdf-reader.html`

The internal viewer should work fine for Mac users. Now, open a new `.Rnw` file in RStudio and copy-paste the following:

```
\documentclass[12pt]{article}
\usepackage[margin=1in]{geometry}
\setlength{\parindent}{0in}
\usepackage{verbatim}
% For bold, italic letters in math
\usepackage{bm}
\usepackage{hyperref}
\author{Kendon Bell}
\title{Minimal \texttt{knitr} document}
\begin{document}
\maketitle

\end{document}
```

Now, we'll build up a knitr document using this template. This minimal document should compile now. You can do this in RStudio using `Ctrl + Shift + K` on Windows or `Command + Shift + K` on Mac.

OLS with matrices

As Max has mentioned, the use of canned routines is not permitted for most of this class; you'll have to write the econometric routines from first principles. First, create matrices of the data, since we will be working mainly with matrix operations. Let $\bm{y}$ be the dependent variable, price, and let $\bm{X}$ be a matrix of the other car characteristics, along with a column of ones prepended. The `cbind()` function binds the columns horizontally and, here, coerces the `matrix` class.

> *"Class" in* `R` *refers to the data type of an object. Some operations only work on some classes, and some classes make better use of memory than others in some situations. Examples of classes are* `logical`, `character`, `numeric`, `integer`, `matrix`, *and* `data.frame`.

Add the following `knitr` chunk using `Ctrl + Alt + I` on Windows or `Command + Option + I` on Mac:

```
<<echo=TRUE>>=
library(foreign)
mydata <- read.dta("auto.dta")
names(mydata) <- c("price", "mpg", "weight")
y <- matrix(mydata$price)
X <- cbind(1, mydata$mpg, mydata$weight)
head(X)
@
```

Now, try compiling this again using `Ctrl + Shift + K` on Windows or `Command + Shift + K` on Mac. Cool huh?! Remember to make sure you have the `auto.dta` file in the working directory. In `knitr`, the working directory is the `.Rnw` file location by default.

In `knitr`, we control the output of each chunk using the options. In the previous chunk, `echo=TRUE` makes `knitr` display the code in the chunk. I usually like to be able to play around with what I have in an `R`-workspace while I'm writing documents, so you can also run these lines through the console using `Ctrl + Enter` on Windows or `Command + Enter` on Mac.

Note that an alternative call in the `cbind` function would be:

```
X <- cbind(rep(1, NROW(mydata)), mydata$mpg, mydata$weight)
```

`rep()`, short for replicate, is an incredibly useful command. However, in this setting `cbind()` only needs to be passed a single 1 – it's smart enough to do the replication itself in order to ensure that the matrix is filled.

Just to make sure that our matrices will be conformable when we regress `y` on `X`, check that the number of observations are the same in both variables.

```
dim(X)[1] == NROW(y)
```

We can quickly estimate the ordinary least squared parameter vector using `R`'s matrix operations.

```
b <- solve(t(X) %*% X) %*% t(X) %*% y
b
```

Here, `solve()` finds the inverse, `t()` finds the transpose, and `%*%` is the matrix operation for multiplication.

Coding

Hopefully you now feel sufficiently prepared (or bored) to start coding up some functions. Remember, there are many ways to calculate these values, this is just one of them. We will be coding up a function that will be able to be reused for general left hand side and right hand side variables.

We'll be using the `auto.dta` data. To keep things simple, we'll define a univariate model, where `y` is price, `X` contains a column of 1's and `mpg`. We'll also define our own `OLS()` function with the following chunk:

```
<<echo=TRUE>>=
y <- matrix(mydata$price)
X <- cbind(1, mydata$mpg)
n <- NROW(mydata)
OLS <- function(y,X) {
  b <- solve(t(X) %*% X) %*% t(X) %*% y
  b
}
@
```

We'll test our function by calling it with `X`, our data matrix that contains an intercept, `mpg`, and `weight`. We'll also confirm our results with `R`'s canned OLS function: `lm()`.

```
<<echo=TRUE>>=
resultsOLS <- OLS(y, X)
lm(y ~ X)
OLS(y, X)
@
```

Standard errors with Loops

We don't yet know how to compute standard errors properly; that'll come later. But, we can cheat a little bit and simulate some standard errors using bootstrapping. We will use this to illustrate loops in `R`. The following chunk will perform 10000 bootstrap replications of the OLS estimator:

```
<<echo=TRUE>>=
set.seed(222015)
resultsBoot <- sapply(1:10000, function(x){
  indices <- sample(1:length(y), size = length(y), replace = TRUE)
  OLS(y[indices], X[indices,])
})
@
```

What does `sapply` do? `sapply` is one of `R`'s many loop functions which repeats a function many times, and returns the results (usually) in a matrix or vector. The first argument of `sapply` is the vector of values that are passed to the `function` in the second argument of the function. Functions passed to functions; weird. Usually, the values of the vector `1:10000` would have some significance; here they are just counters. Using `rep(1, 10000)` would have worked the same. Let's look at the `sapply` output:

```
head(resultsBoot)
```

Oops. This looks like garbage. It seems that the `sapply` function takes our column vector `OLS()` results and stacks them horizontally in a big wide matrix. Try this:

```
head(t(resultsBoot))
```

That's better. Now we see the bootstrap replications nicely. Now, let's turn this into a standard error function:

```
<<echo=TRUE>>=
seBoot <- function(y, X, nReps){
  # Get the bootstrap results
  resultsBoot <- sapply(1:nReps, function(x){
    indices <- sample(1:length(y), size = length(y), replace = TRUE)
    OLS(y[indices], X[indices,])
  })

  # Get the standard deviations
  myRowMeans <- rowMeans(resultsBoot)
  sapply(1:NCOL(X), function(x){
    sqrt(sum((resultsBoot[x,] - myRowMeans[x])^2)/(nReps - 1))
  })
}

# These are the canned standard errors.
resultsLm <- summary(lm(y ~ X))
resultsLm$coefficients[,2]

# These are our bootstrap standard errors.
seBootResults <- seBoot(y, X, 1000)
seBootResults
@
```

Tables in `knitr`

Here, we'll introduce the simple, but immensely powerful `\Sexpr{}` function. This is what `knitr` uses for inline expressions. This allows us to insert dynamic numbers in our documents. This is great for tables and preventing errors!

Firstly, you should note that the `xtable` and `stargazer` packages exist; these packages generate LaTeX code for tables on the fly and can sometimes suffice when using `knitr`. I seldom use `xtable` and haven't tried `stargazer` yet. I like using LaTeX, which I know well and it gives me flexibility when generating tables. Also, check out the `dcolumn` package in LaTeX, which allows you to align columns by decimal points.

The following `knitr` code generates a simple table of our results:

```
\begin{center}
  \textsc{Table 1: OLS Results and Bootstrap Standard Errors}\\
  \vspace{1em}
  \begin{tabular}{lr}
    \hline
    Variable & Coefficient \\
    \hline
    Intercept & \begin{tabular}{r}
                  \Sexpr{prettyNum(resultsOLS[1], digits = 3)} \\
                 (\Sexpr{prettyNum(seBootResults[1], digits = 3)})
                \end{tabular} \\
    MPG & \begin{tabular}{r}
             \Sexpr{prettyNum(resultsOLS[2], digits = 3)} \\
            (\Sexpr{prettyNum(seBootResults[2], digits = 3)})\\
          \end{tabular} \\
    \hline
  \end{tabular}
\end{center}
```