How to Include 'R' Code in a 'LaTeX' Document

J.P. Olmsted

Updated September 21, 2010

Introduction. There are several ways one can easily and attractively include 'R' code (or any other computer source code) within a LaTeX document. The first way (using the 'verbatim' environment and related commands) is covered just to provide contrast to my usual and preferred approach (using the 'listings' package). I also mention one approach that I don't use despite it being quite sophisticated.

Requirements. I assume that you have a working LaTeX installation, a file containing the 'R' code you want to include in the document, and either (1) the 'listings' package installed in the appropriate location as far as the 'LaTeX' executables are concerned or (2) a package manager that you can use to easily install it such that the first assumption in this pair is "effectively" met.

Using 'verbatim'. The 'verbatim' environment is provided immediately for use when preparing a document of class 'article'. It is begun with "`\begin{verbatim}`" and ended with "`\end{verbatim}`". Anything between those two 'LaTeX' commands is printed by 'LaTeX' verbatim and not processed in the usual way. Thus, you can include 'R' code in-between and it will not be processed as if it were LaTeX markup (which would result in errors). Related to this is the `\verb!!` command. Here, anything between the two exclamation points is treated as text to be processed verbatim. The 'verbatim' environment is to display math as the `\verb!!` declaration is to inline math. There are several deficiencies with this approach that the 'listings' package remedies:

- If you 'R' code is changed you have to remember to change it in your write-up
- 'verbatim' approaches can only *print* your code, they can't *understand* it

- You have to manually copy and paste your code
- There are no line-numbers, which are useful for human consumption of and digestion of computer code

Using 'listings'. The 'listings' package is loaded at the beginning of a LaTeX file in the preamble—that is, after `\documentclass{}` and before `\begin{document}`—with `\usepackage{listings}`. The next thing I do is change the settings used by 'listings' by using the `\lset{}` declaration. There are many settings one can change. Both `http://en.wikibooks.org/wiki/LaTeX/Packages/Listings` and `http://www.ctan.org/tex-archive/macros/latex/contrib/listings/` provide ample documentation. However, here is an example of my `\lset{}` declaration for 'R' code.

```
\lstset{
language=R,
basicstyle=\scriptsize\ttfamily,
commentstyle=\ttfamily\color{gray},
numbers=left,
numberstyle=\ttfamily\color{gray}\footnotesize,
stepnumber=1,
numbersep=5pt,
backgroundcolor=\color{white},
showspaces=false,
showstringspaces=false,
showtabs=false,
frame=single,
tabsize=2,
captionpos=b,
breaklines=true,
breakatwhitespace=false,
title=\lstname,
escapeinside={},
keywordstyle={},
morekeywords={}
}
```

Then, after having including this (also in the preamble), I simply include my external 'R' files. They can be included using an `\include{}`-like interface to the external files. By using the command `\lstinputlisting{./code.R}`

to include the code from the file `code.R` which is located in the same directory as the 'LaTeX' source file. Any other location for the to-be-inputted 'R' source code can be specified using the normal *parent directory* and *current directory* shortcuts (`../` and `./`, respectively). Now, the 'listings' package does provide an environment into which you can copy and paste code, but I am ideologically opposed to this. Because I set up my 'R' code such that it can be `source()`-ed, it is coherent to display it using the `\lstinputlisting{}` approach. Beyond that, I need only reference file names (which are printed as the caption of the code in my settings) and line numbers to pick out certain calls.

Using Sweave. Sweave (said s-weave) is a way of integrating 'R' code with 'LaTeX' documentation of the 'R' code. In most cases, problem sets requiring work in 'R' fit into this framework. The reason I don't find this particularly attractive is that I view the *doing* and *documenting* steps as discrete and prefer they remain modular. I should neither have to make a document to run some 'R' code nor should I have to calculate means just to see a new header. This approach may be ideal for certain kinds of tasks, but I am skeptical that problem sets in Political Science using data, statistics, and 'R' are one of them.