An Intermediate Level LaTeX Primer: Moving Beyond the Basics

Jonathan W. Duggins

Department of Statistics

Virginia Polytechnic Institute and State University

Graduate School Seminar
February 23-24, 2010

VirginiaTech
Invent the Future

Outline

Defining and updating commands

- `\renewcommand{cmd}[args][opt]{def}` allows you to modify existing commands
- `\newcommand{cmd}[args][opt]{def}` allows you to create new commands
- `cmd` is the name of the command you are renewing/defining
- `args` is the number of arguments the command takes (default is none)
- `opt` allows the declaration of the first argument as optional (the value you put here is the default)
- `def` is your definition of the command, i.e. the commands executed by LaTeX when you issue `cmd`

Some examples

- `\qedsymbol` yields □ but `\renewcommand{\qedsymbol}{\blacksquare}` now makes `\qedsymbol` yield ■
- `\renewcommand{\sc}{\textsc}` allows me to replace the deprecated `\sc` command with the new, improved `\textsc`
- `\newcommand{\argmax}{\ensuremath{\text{argmax}}}` defines this command: $\text{argmax}_i f(x_i)$
- `\newcommand{\C}[2]{\text{COV}\left[#1,#2\right]}` allows me to input `$\C{X}{Y}$` and get $\text{COV}[X, Y]$

Reusing your preamble

- LaTeX allows you to load external files with `\input` or `\include`
- `\input` is recommended because they can be nested
- LaTeX uses standard *nix references
  - `.` refers to the current directory
  - `..` refers to the next higher level directory
  - `/` separates directory names
- `\input{MyPreamble.tex}` loads the commands in the file MyPreamble.tex, which must be in the same folder as your current *.tex* file
- `\input{../../MyLaTeXFiles/MyPreamble.tex}` goes up two levels, then looks in the folder MyLaTeXFiles for MyPreamble.tex
- The main body may be sectioned and loaded this way as well, but be aware these files can not be compiled on their own without loading an additional package and using different commands

Manipulating lists

- Change the numbers
- Change the numbering style
- Change the bullet symbol
- Change the spacing

Adjusting the counters

1. In LaTeX counters are how the current page, section or even location in a list are identified
2. Each counter has a name, so commands such as `\addtocounter{}{}` and `\setcounter{}{}` can be used to change their values
3. For lists, the counters are `enumi`, `enumii`, `enumiii` and `enumiv` depending on the level of nesting
4. By issuing `\addtocounter{enumi}{-2}` after this `\item` ...
3. ... we get this result, which wouldn't matter in an `itemize` environment
4. Though, `\setcounter{enumi}{2}` would have had the same result using an absolute command

Arranging lists horizontally

- Occasionally it would be nice for the lists to spread out horizontally as well as vertically
- Incorporating the `tabular` environment, along with your ability to set counters, allows for things such as

1. First we put in some items, but ...
2. space is at a premium.

3. So we move over to a new column ...
4. being sure to set the counter to 2.

1. Otherwise we would end up with...
2. something that looked like a new list!

Lists: Odds and ends

- Occasionally you want to make other changes as well
  1. `\renewcommand{\theenumi}{\Alph{enumi}}` changes the counter to using capital letters
  2. `\renewcommand{\labelenumi}{\theenumi}` then changes the label to using capital letters
  3. `\renewcommand{\labelenumii}{(\theenumii)}` changes the second level counter to be surrounded by parentheses, but only in the list itself, not in references
  4. `\itemsep=.1in` would increase the separation between successive items to 0.1 inches, as in this list
- □ In any list, I can change the symbol for that item by issuing something like `\item[$\Box$]`
- > `\renewcommand{\labelitemi}{\textgreater}` changes all the successive labels for the lists

The tabular environment

- The `tabular` environment allows for construction of table-like objects
- Very flexible, allowing for tables ranging from the very simple to the very complex
- Start with the command `\begin{tabular}[pos]{specs}` where
  - *pos* controls the vertical location of the table relative to the text's baseline
  - *specs* contains the majority of the specifications for table construction
- As indicated, commands may be issued inside the `tabular` environment
- Environments must be closed, in this case with `\end{tabular}`

Designing a basic table

- *specs* is where the number of columns and their default appearance is defined
- For each column you need to define the default look
  - `l, c, r` indicates a left-, center-, or right-aligned column
  - `p{width}` indicates a paragraph column is to be used
  - `|` inserts a vertical line
- In the body of the table we have
  - `&` to separate columns
  - `\\` to indicate the end of a row
  - `\hline` to insert a horizontal line the width of the table
  - `\cline{i-j}` to insert a horizontal line from column *i* to column *j*

A basic example

```
\begin{tabular}{lrc}
    -1.7 & -0.7 & No  \\
    -0.2 &  0.8 & Yes \\
     0.1 &  1.1 & No
\end{tabular}
```

-1.7	-0.7	No
-0.2	0.8	Yes
0.1	1.1	No

A more complex example: The result

		Sample Size		
		20	60	100
Weibull	H_0	No	Yes	Yes
	H_1	No	Yes	Yes
Gamma	H_0	No	No	No
	H_1	No	No	Yes

} Better Fit

- This requires the `array` package (column-wise commands) `multirow` package
- Other important packages:
  - `rotating` for sideways tables
  - `xcolor` with the `tables` option for coloring cells or rows

A more complex example: The code

```
\begin{tabular}{cc|c|c|c|>{\scriptsize}l}
\cline{3-5}
& & \multicolumn{3}{|c|}{Sample Size} \\
\cline{3-5}
& & 20 & 60 & 100 \\
\cline{1-5}
\multicolumn{1}{|c|}{\multirow{2}{*}{Weibull}} &
    \multicolumn{1}{|c|}{$H_0$} & No & Yes & Yes &
    \multirow{2}{*}{$\Bigg\}$Better Fit}     \\
\cline{2-5}
\multicolumn{1}{|c|}{}                          &
    \multicolumn{1}{|c|}{$H_1$} & No & Yes & Yes &  \\
\cline{1-5}
\multicolumn{1}{|c|}{\multirow{2}{*}{Gamma}}   &
    \multicolumn{1}{|c|}{$H_0$} & No & No  & No  &  \\
\cline{2-5}
\multicolumn{1}{|c|}{}                          &
    \multicolumn{1}{|c|}{$H_1$} & No & No  & Yes &  \\
\cline{1-5}
\end{tabular}
```

Table versus Tabular

- The `tabular` environment is devoted to typesetting the actual table
- The `table` environment deals with placement in the document, labeling, captioning, etc.
- For most articles each `tabular` is enclosed in a `table` environment
- Using the `table` environment makes your table into a float - something LaTeX will never break across pages
- Opening syntax is `\begin{table}[pos]` and closing is `\end{table}`
- The *pos* allows some control over where to put the float
  - `h` – here (approximately)
  - `t` – top of the page
  - `b` – bottom of the page
  - `p` – on a page designated for floats only

More on floats

- The default is [tbp], but you often want more control
- Many internal float parameters guide LaTeX's placement of a float – often invisibly and frustratingly!
- Using [h], for example, doesn't always place a float "here" – using [h!] can fix this, but ...
- Since it doesn't always work the `float` package provides an option [H] that does place it exactly where you say
- LaTeX has some internal restrictions on how many floats it can simultaneously process – if you hit the limit (it's a little technical) you can use the `morefloats` package to increase the limit some
- So, now let's revisit our basic table from before, in Table 1!

Our first float

```
\begin{table}[h!]
\begin{tabular}{lrc}
    -1.7 & -0.7 & No  \\
    -0.2 &  0.8 & Yes \\
     0.1 &  1.1 & No
\end{tabular}
\caption{\label{basictable} This is our first example of a table!}
\end{table}
```

-1.7	-0.7	No
-0.2	0.8	Yes
0.1	1.1	No

Table 1: This is our first example of a table!

The figures environment

- Figures are usually used as floats, and so behave much like tables - starting with `\begin{figure}[pos]`, ending with `\end{figure}` and allowing captioning and labeling
- We now use `\includegraphics[attributes]{filename}`, rather than using a `tabular` to define the table
  - *attributes* include things like, width, height, angle, etc.
  - *figure* is the filename for the graphic and may be a relative or absolute path but must contain the extension
- Since they depend on external files, so we have to make the "right" choice about what format to use
- In any case, including graphics requires the `graphicx` package is loaded and it will process the graphics as needed depending on how you compile

Types of figures

- If you compile with LaTeX you can *only* use Encapsulated PostScript (EPS) or PostScript (PS) figures
  - EPS are vector graphics, and so can be *very* high resolution
  - The encapsulation means the file comes with its own bounding box to tell LaTeX the native size of the image, this makes EPS files very easy to deal with in LaTeX
  - Often your program won't output files in EPS format, so you would need a converter
  - PS files require you to define the bounding box yourself in the *attributes* part of the environment declaration

Types of figures

- If you compile with pdfLaTeX you *can not* use EPS or PS (*ahem*), but you can use JPEG (or JPG), PNG or PDF figures
  - JPEG is great for photographs, but it uses lossy compression and so is not good for drawings or diagrams that tend to have a need for reliable rendering of adjacent pixels with high contrast
  - The Portable Networks Graphic (PNG) format uses a lossless compression and so is better for most bitmap images. *Never* use GIF!
  - PDF figures are good for non-bitmap images such as figures that also have text and symbols
  - With JPEG and PNG it is recommended to give an explicit height or width since they don't have internal data on their native size

A figure

```
\includegraphics[height=0.65\textheight]{Hannah.jpg}
```

Some options

```
\includegraphics[scale=.15, angle=25]{Hannah.jpg}
```

Cropping

```
\includegraphics[scale=.4, clip, trim=250mm 80mm 50mm 60mm]{Hannah.jpg}
```

Figures: Some odds and ends

- It is possible to mix EPS/PS and JPEG/PNG/PDF figure usage in a single document with the help of packages such as `epstopdf`
- Journals typically advocate a specific file format, so check first!
- Figures can be placed side-by-side using the `subfloat` package
- It is possible to put the captions in other locations (left, right, top) and to do word-wrapping as needed
- Each file must be a single graphic (no animation)

Bibliography management

- LaTeX provides a built-in environment `thebibliography` that allows references to be placed in the same file as the document itself
- This isn't very practical for academe, where we tend to reuse citations from within our field (but some journals require it!)
- It also lacks flexibility in that if a journal wants a different format I may need to add/remove/edit information
- BibTeX creates a *.bib* file that is plain-text and contains your references
- Since the *.bib* file itself doesn't control what goes into your bibliography, you can enter more information than your current journal requires
- Multiple files can be used and not all entries need be printed
  - You can publish three papers, creating a *.bib* file for each, then load all three when writing your lit review
  - You can write the lit review, then use the same *.bib* file for your resulting publications

Getting started with BibTeX

- Effective use of BibTeX then requires three things:
  1. Creating a *.bib* document containing your references
  2. Getting your LaTeX file to use your BibTeX file according to your preferred style
  3. Knowing the commands needed to actually cite the material
- Each BibTeX entry is essentially a listing of all the relevant attributes for your reference and in general looks like

```
@type{cite-key,
    attribute={},
    another_attribute={},
    third_attribute={},
    .
    .
    .
}
```

Templates

- The `@type` tells LaTeX to use the `type` template, thus controlling what BibTeX keywords are available (some required, some optional)
- Common choices are `@book`, `@article`, `@proceedings` etc. – it can take a while to learn what keywords are associated with each template
- The *cite-key* is your name for this reference, to be used when you cite it and it is *internal to LaTeX*
- The attributes are things like, author, title, year, journal, etc.

```
@article{Barrowman,
    author = {Nicholas J. Barrowman and Ransom A. Myers},
    title = {Still More Spawner-Recruitment Curves: {T}he
        Hockey Stick and its Generalizations},
    journal = {Canadian Journal of Fisheries and Aquatic Sciences},
    year = {2000},
    volume = {57},
    number = {},
    pages ={665-676}
}
```

Google Scholar

Making the bibliography available

- Using your bibliography means: LaTeX needs to know how to format it and where to find it
- Formatting is done via `\bibliographystyle{my-style-choice}`
- Locating is done via `\bibliography{bib1,bib2,...,bibn}`
  ↑ notice no spaces!

```
\bibliographystyle{plain}  %use the bib style "plain.bst"
\bibliography{IntroLaTeX,InterLaTeX} %load two .bib files
```

- Most journals have style files prepared, but common ones can be found here at UC Boulder's LaTeX FAQ

Citing

- `\cite{cite-key}` places a citation according to your current style and adds that reference to the bibliography
- `\nocite{cite-key}` *does not* place a citation, but adds that reference to the bibliography
- `\nocite{*}` places no citations and adds *all* entries from the *.bib* into the bibliography
- The standard .bst files use numbers as the reference, rather than a *author-year* style
- The `natbib` package allows for easy generation of *author-year* citations with many variants
  - `\citet{}` gives *Author*, (*year*)
  - `\citep{}` gives (*Author, year*)
  - Other commands allow you to cite just the author or year, the full set of author names (no et al.), etc

Compiling

- BibTeX is not part of LaTeX and so writing a document that uses BibTeX requires some extra steps
- For concreteness, suppose I have a LaTeX document "thesis.tex" and a BibTeX file "refs.bib"
  1. Run LaTeX or pdfLaTeX on thesis.tex
  2. Run BibTeX on thesis.tex
  3. Run LaTeX or pdfLaTeX on thesis.tex
  4. Run LaTeX or pdfLaTeX on thesis.tex
- The extra processing ensures that LaTeX and BibTeX communicate the cross-referencing correctly
- This process only needs to be done if the references are changed
- As with most of LaTeX, there are tons of bibliography styles that you can look into
- Adding the `hyperref` package allows you to create citations that actually link to the reference's location in the bibliography

The beamer package

- There have been presentation packages in the past, but beamer is the new front-runner
- It is a regular LaTeX class
- Allows for overlays and dynamic effects such as successively revealing parts of slides or using animation
- Comes with many flexible themes so your aesthetics don't suffer
- Final product is a PDF for easy viewing and distribution
- Allows for high-level user control, but it is not required
- Allows for the easy creation of handouts and presentation notes to accompany the presentation
- The user manual is 224 pages long!

References

- Google!
- The Wikibook on LaTeX
- Helmut Kopka's "Guide to LaTeX" (4th edition)
- Leslie Lamport's "LaTeX: A Document Preparation System" (2nd edition)
- The TeX users group (www.tug.org)
- The Comprehensive TeX Archive Network (www.ctan.org)
- Beamer user's guide (from CTAN) http://www.ctan.org/tex-archive/macros/latex/contrib/beamer/doc/beameruserguide.pdf