

# Demography 213

## Week 10 L<sup>A</sup>T<sub>E</sub>X

Carl Mason  
carlm@demog.berkeley.edu

November 21, 2011

### Contents

<b>1</b>	<b>Introduction</b>	<b>1</b>
<b>2</b>	<b>L<sup>A</sup>T<sub>E</sub>X overview</b>	<b>2</b>
<b>3</b>	<b>Step by step procedure for producing output from LaTeX files</b>	<b>3</b>
<b>4</b>	<b>The absolute minimum “.tex” file</b>	<b>3</b>
<b>5</b>	<b>Somewhat more LaTeX</b>	<b>4</b>
<b>6</b>	<b>AucTeX: the emacs mode for LaTeX</b>	<b>5</b>
6.1	Invoking LaTeX mode . . . . .	5
6.2	Using LaTeX mode . . . . .	5
6.3	Printing and what to do with .dvi files . . . . .	7
<b>7</b>	<b>Assignment</b>	<b>8</b>

#### Abstract

In order to complete this assignment you will need to know a *little bit* about LaTeX. LaTeX (pronounced *lātek*) is a logically organized text processing package, but more to the point, it is the text processing package that you must use in order to complete 213 and go on to a successful and fulfilling life.

## 1 Introduction

The goal for this week is to expose you to LaTeX. Unfortunately, you are unlikely to experience, during this week, the productivity gains that are possible from using LaTeX. This sad truth holds both because the project at hand is rather

small and LaTeX really pays off for large projects—and also because LaTeX requires some skills which take time and experience to acquire.

By doing a small “dissertation” in LaTeX, you will get a taste of how LaTeX works, and have a useful template with which to begin your next project.

## 2 L<sup>A</sup>T<sub>E</sub>X Overview

The fundamental differences between LaTeX and the more typical word processors of today are:

- LaTeX requires you to specify how each chunk of text *functions* as opposed to how it *looks*. For example, to get the abstract of this paper to appear as it does, at the top of the first page with the optimal font size and indentation, I enclosed the appropriate text between “begin{abstract}” and “end{abstract}” tags. In LaTeX-speak, I put the appropriate text in the abstract *environment*.

```
\begin{abstract}
  text goes here
\end{abstract}
```

Because LaTeX documents are *logically* rather than *visually* structured, much of how the text actually looks in your “.tex” files is ignored. Line breaks, blank lines, extra spaces between words, in most cases make no difference as far as the final output is concerned.

- LaTeX must be compiled in order to see what its output will look like. Unlike typical word processors which promise to show you on screen what your document will look like when it’s printed, LaTeX requires an additional step to preview your document.

Your “.tex” file is analogous to a program you write in R. It is a set of instructions telling the computer what you would like it to do. Whereas with R, you are generally telling it what to do with a humongous pile of numbers, with LaTeX, you are telling it what you would like done with a bunch of words, graphics and perhaps some mathematical equations.

With R — in order to see any useful output, you must feed your program to the R process—and then correct whatever syntactical errors your code contains.

With LaTeX, the same is true, only instead of feeding your code to R, you use the command:

```
@:> latex filename.tex
```

If `filename.tex` is syntactically correct, the result is a *dvi* file that can be viewed or sent to the printer.

There are two important points here: first, there is this extra step that you must go through to see your output, and second, until your .tex file is syntactically correct, **all you'll see are error messages**.

### 3 Step by step procedure for producing output from LaTeX files

1. Create a file called `something.tex` containing LaTeX directives and text in various environments. You do this in emacs. It is the hard part.
2. Run the command `@:> latex something`. thereby producing a file called `something.dvi`. You may be told to run it again in order to resolve cross references. Note that it is also possible/easy to produce pdf output instead of dvi. It is a matter of which viewer you prefer.
3. If necessary, decipher the error messages, correct the file and return to step 2.
4. Run `@:> xdvi something.dvi` to see what your document will look like when printed.<sup>1</sup>
5. Print your `something.dvi` file with a command like:

```
| @:> dvips -P age something.dvi
```

The list above shows the Unix commands that you *could* enter to compile and display your document. But of course, emacs can spare you from ever actually typing those commands. Using the appropriate emacs mode, you simply hit `ctrl` + `c` `ctrl` + `c` as described below, to do all of these things.

### 4 The absolute minimum “.tex” file

The `demonstration.tex` file could serve as a template for a small paper, it contains all of the constructs that you need in order to produce a short but handsomely formatted paper of the sort that Demographers produce. Before turning to that file, however, it is useful to look at the absolute bare bones minimum LaTeX file. Figure 1 shows how simple it is.

---

<sup>1</sup>Generally, at this point you will probably need to do some more editing and then return to step 2

Figure 1: Minimum LaTeX file

---

```
\documentclass[11pt]{article}

\begin{document}
Boring sentence. Second boring
sentence. That makes a paragraph.

Yet another boring paragraph starts here.
\end{document}
```

---

Some observations about this pathetically simple example:

- All .tex files must contain a `\documentclass` directive which tells whether you are writing an article (most often the case) a report, a book, a ucthesis, a letter, or something more exotic.
- Nearly all LaTeX directives begin with a backslash `\`
- Many LaTeX directives take arguments which are enclosed in `{}`. Some also take arguments that are enclosed in `[]`. If an argument is **required** it will be enclosed in `{}`; if an argument is **optional** it will be enclosed in `[]`. Not all latex directives require arguments.
- A blank line indicates the beginning of a new paragraph.
- Linebreaks (not followed by blank lines) are irrelevant.

## 5 Somewhat more LaTeX

In order to complete the assignment, you will need to know a bit more about LaTeX than how to write the minimal .tex file. It is strongly recommended that you spend a little time with the *Not so Short Introduction to LaTeX* which can be found at

[//www.demog.berkeley.edu/Refs/lshort.pdf](http://www.demog.berkeley.edu/Refs/lshort.pdf).

The book *LaTeX A Document Preparation System*, by Leslie Lamport is also quite useful, Copies of this book can be found (if you are lucky) in the library and in the computer lab. There is also a wealth of information online.

[www.tug.org/begin.html](http://www.tug.org/begin.html), contains more than you will want to read. When you decide to become a hardcore LaTeX user, you will want to acquire a copy of *The LaTeX Companion (Tools and Techniques for Computer Typesetting) 2nd*

*edition*<sup>2</sup> A copy of this very thick book has also been seen from time to time in the basement lab.

To motivate all this effort, take a look at what you are in for regarding formatting a dissertation: <http://grad.berkeley.edu/policies/guides/diss-guide.html#Formatting%20your%20Manuscript>

## 6 AucTeX: the emacs mode for LaTeX

It should come as no surprise that in addition to ESS (the mode that we use for R), emacs also has AucTeX—a mode for editing LaTeX files. Like ESS mode, AucTeX (or LaTeX mode) does the following:

- Colors the parts of you file according to function in LaTeX.
- Matches braces and parentheses for you.
- Has key sequences for doing important things.
- Has lots of documentation that you will probably never read. If you would care to prove me wrong on this point, visit <http://www.gnu.org/s/auctex/manual/auctex.pdf>

### 6.1 Invoking LaTeX mode

You can invoke Auctex aka LaTeX mode either by visiting a file with a “.tex” suffix (in emacs<sup>3</sup>), or via the emacs command: `[alt]+[x] latex-mode`. When LaTeX mode is running, the status line at the bottom of the emacs window will indicate it with the word “LaTeX” near the middle.

### 6.2 Using LaTeX mode

Plain vanilla emacs is sufficient for editing LaTeX documents, you do not *need* to master any of the features of LaTeX mode in order to use LaTeX<sup>4</sup>. Consequently, what’s contained here is just the 4 most important (i.e. time-saving) tricks. You can pick up the rest as you write ever longer papers. If you are impatient or bored, you can also read all about AUCtex at <http://www.gnu.org/s/auctex/manual/auctex.pdf>

---

<sup>2</sup>by Frank Mittelbach, Michel Goossens, Johannes Braams, David Carlisle, Chris Rowley

<sup>3</sup>duh

<sup>4</sup>Of course, you don’t need to go to graduate school either—you could just invent Demography for yourself

Runs latex on the buffer that is, it submits the contents of the file to latex. After prompting you to make sure that you indeed wish to the run the “default” program: latex (hit `enter`), you will see either the happy message:

LaTeX: successfully formatted nn pages”

or the sad message:

LaTeX: Errors in /path/to/filename.tex

Use C-c ‘ to display.

If, as is likely, the latter sad message is what you see, then do what it says, and hit `ctrl`

`C-c C-c` →

+c followed by `‘`.

Note that the character `‘` is **not** the apostrophe but rather the *backtick* it usually shares a key with the `~`

at the upper left side of the keyboard. After you hit C-c ‘, emacs will split the screen and show you the error message in one window and *position the cursor at the location of the error in the other window.*

In most cases (with some experience) you can quickly correct the problem, save the file and hit `C-c C-c` again. Without experience, however, this can be quite confusing

---

It is particularly confusing if the error in your file is that of omitting a closing } for some environments. If this case, after seeing the sad message and hitting `ctrl` + `‘`, instead of showing the error message and location of the error you might instead see a blank buffer and the latex error message:

```
Search failed: ‘^1.’
```

So if you encounter this oddness, look for a missing curly bracket. If you compile fairly frequently as you go along it will be easier to find syntax errors as they will have crept into the code with your most recent changes.

`C-c C-c` → Runs `xdvi` on a compiled latex file. That's right, it's the same command as you use to compile. Emacs can tell what you want to do by what you have just done. If the `.dvi` file is newer than the file holding the latex commands then emacs figures that you probably want to see what your document looks like, so it launches `xdvi`. A common sequence of events is to hit `C-c C-c`, notice see the happy message then hit `C-c C-c` again to see the document (4 keystrokes, 1.35 seconds).

---

`C-c C-e` → Sets up an “environment”. This speeds the insertion of environment definition commands. It prompts you as to which environment you would like—and, since it knows what the choices are, you need only type the first few letter + `tab`. If there are required or optional arguments for the particular environment, you will be prompted accordingly.

---

`C-c C-f < ? >` → Where `< ? >` is `C-b`, `C-e`, `C-i` or `C-t`, inserts commands to change typefaces to **bold** *emphasis italics* **keyboard** respectively.

---

### 6.3 Printing and what to do with `.dvi` files

After successfully compiling a `.tex` file, you wind up with a `.dvi` file. If you are operating in emacs/AucTex then you might not be aware of the `.dvi` file because your last `C-c C-c` command spawned an `xdvi` process which displays your work on the screen in a new window.

What's really happening there is that your first `C-c C-c` created a `.dvi` file and your second `C-c C-c` executed the command:

```
@:> xdvi filename.dvi
```

It is possible to print from the `xdvi` application via the menu. So if you have an old `.dvi` file laying about you can always run `xdvi` on it and print that way. But it is also possible and often useful to convert your `.dvi` file into some other format for emailing, storing or printing.

As noted above, it is also possible to skip the `.dvi` stage entirely by using the command `pdflatex` in place of `latex`. In Emacs/Auctex the key combination for toggling between pdf dvi mode is `C-c C-t C-p`.

**.pdf** An excellent choice for emailing and storing. To convert an existing `.dvi` into a `.pdf` type:

```
@:> dvipdf filename.dvi
```

the result will be a `filename.pdf`.

Obviously, if you are using pdf mode, then your output is already in pdf and there is not much point in trying to convert your nonexistent `.dvi` file into pdf.

**postscript** When you print a `dvi` file, you do it by converting it to postscript and then sending it to the printer. The same command, can be used for converting your `.dvi` into a postscript file:

```
@:> dvips filename.dvi -o filename.ps
```

The `-o` option specifies the output file. It is not optional, if you leave it off, your file will wind up at your default printer.

To print a `dvi` file to a specific printer type:

```
@:> dvips filename.dvi -P printer-name
```

The above command will convert `filename.dvi` to postscript format and send it to the printer named `printer-name`.

## 7 Assignment

Using the graphs, R code, and profound insight developed over the last 2 weeks (or longer perhaps), create a dissertation suitable for publication in a not very selective journal. Obviously, you must use LaTeX to accomplish this task. You will probably want to use the `demonstration.r` file as a dissertation template. Just as always, you'll need to create a new directory called `213/WeekN`, and snork in the demonstration file from: `~carlm/213/Latex/demonstration.r`. Unlike previous weeks, this demonstration file is a latex file rather than an R file.

Your finished document should include:

1. All the obvious bits like a title and the authors name.
2. An abstract.
3. At least two sections.
4. At least one graph included as a figure.
5. At least one reference to your figure.

6. At least one thing that makes me laugh.

Also unlike every other week, please print this assignment and hand it in on paper.