Some Tools, Tips, and Tricks

Michael Ash*
mash@econ.berkeley.edu

17 June 1999

1 LaTeX 2ε Tips

LaTeX is available free over Internet for many platforms including DOS, Windows, and Macintosh. See the CTAN web site, discussed at greater length in Section 1.4. There are also low-cost distributions, e.g., on CD–ROM, that may be significantly easier to install than the free distributions. Below I offer some of the LaTeX tricks that I use most frequently or that I had a hard time figuring out. LaTeX has a great worldwide community that will usually answer how–to questions. The newsgroup for this discussion is `comp.text.tex`. Phil Spector's LaTeX presentations are on his web site: `http://www.stat.berkeley.edu/~spector`. An excellent reference site for LaTeX with a hyperlink table of contents is `http://www.ens.fr/~cousot/software/LaTeX-html/`.

1.1 The `minipage` environment

Minipage lets you put boxes of text in your document with great flexibility about placement. Footnotes in a minipage environment are handled in a way that is particularly useful for putting footnotes in figures or tables. Footnotes are lettered instead of numbered, the footnote counter begins at a,b,... within each minipage, and the footnotes appear at the bottom of the minipage instead of at the bottom of the page.

Minipage takes an optional position argument, [t] for top or [b] for bottom, and a mandatory width argument. You can specify the width with a measurement, e.g., `{4in}` will give 4 inches, or

```
{0.75\textwidth}
```

will give a minipage that is $\frac{3}{4}$ the width of the text. The following minipage,

*For the time being, my web site is `http://socrates.berkeley.edu/~maash`. The web site has links to many of the resources listed in this document. I have attempted to avoid making this document too EML–specific.

This code will give a 2-inch wide minipage that begins at the 4-inch mark[a] on the page.

[a]Note the use of hspace to move the text to the 4-inch mark.

was produced with

```
\hspace{4in}\begin{minipage}[t]{2in}
  This code will give a 2-inch wide minipage that begins at the 4-inch
  mark\footnote{Note the use of hspace to move the text to the 4-inch
    mark.} on the page.
\end{minipage}
```

1.2 The `table` and `tabular` environments

The `tabular` environment allows the insertion of row-by-column grid of cells that I think of as a *table*. The `table` environment inserts a numbered and captioned box in the text at a place where you specify and LaTeX finds room. It's very common but not necessary to embed a tabular environment inside a table environment. If you want an unnumbered, uncaptioned grid of cells to appear in your text, you can jump right into the tabular environment.

One of my favorite constructions in LaTeX is to embed tabular inside minipage inside table. (You can save keystrokes by using emacs to edit your LaTeX files; there is a shortcut to insert environments with two keystrokes.) As noted above, this construction permits table footnotes as well as table notes that are offset from the rest of the text.

Table 1: Minipage/footnote demonstration: Home Run Kings

Slugger	Year	HRs
Mark McGuire[a]	1998	70
Roger Maris[b]	1961	61
George Herman Ruth[c]	1927	60

Notes: This table shows some home run kings and the number of home runs each hit in his banner year.
Source: `http://stats.espn.go.com/premium/mlb/profiles/chart/players/3866.html`

[a]On steroids, but natural steroids.
[b]No more asterisk, no more record.
[c]"Babe"

The output in Table 1 was produced with the input in Table 2.

Table 2: Source for Home Run Kings

```
\begin{table}[htbp]
  \begin{center}
\caption{Minipage/footnote demonstration: Home Run Kings}
\label{tab:hrk}
 \begin{minipage}[t]{3in}
  \begin{center}
  \begin{tabular}[t]{lcc}
  \hline\hline
  Slugger                                                    & Year &  HRs \\
  \hline
  Mark McGuire\footnote{On steroids, but natural steroids.}  & 1998 &  70 \\
  Roger Maris\footnote{No more asterisk, no more record.}    & 1961 &  61 \\
  George Herman Ruth\footnote{``Babe''}                      & 1927 &  60 \\
  \hline\hline
  \end{tabular}
  \end{center}
  Notes: This table shows some home run kings and the number of home runs
  each hit in his banner year.\\
  Source: \url{http://stats.espn.go.com/premium/mlb/profiles/chart/players/3866.html}
 \end{minipage}
  \end{center}
\end{table}
```

Aesthetic tip: Almost never use *vertical* lines in your tables. They do not aid reading and they do not look good.

1.2.1 multicolumn

Multicolumn is a useful tool for designing LaTeX 2$_\varepsilon$ tables. It allows text to take up more than one column. The syntax is

```
\multicolumn{cols}{pos}{text}
```

where: cols specifies the number of columns to span; pos specifies the formatting of the entry (c for centered, l for flushleft, r for flushright); and text specifies what text is to make up the entry. There is an example of this in Table 3 with source in Table 4.

Another use of multicolumn is to change the alignment or justification of a single cell, e.g., you would like the heading of a column to be centered but all of the numeric entries to be right justified. In the tabular statement, specify the format of the whole column as right justified; in the particular cell, use a one-column multicolumn statement to specify that particular cell as centered.

```
\multicolumn{1}{c}{My text}
```

Table 3: Multicolumn demonstration

		Age in	
Person	Year of birth	1969	1999
Dad	1940	29	59
Mom	1943	26	56
Me	1969	0	30

1.2.2 Column separators and decimal alignment

To align on decimal points, you can use the @ specifier in the tabular format. In scientific tables it is often desirable to align the columns on a decimal point. This can be done using the @ col specifier and breaking the number into the integral part in a right-justified column and the fractional part in a left-justified column. Note that the decimal point is replaced by the column separator, &, and that the @ suppresses the intercolumn space. The following input:

```
\begin{tabular}{r@{.}l}
  3&14159\\
  16&2\\
  123&456
\end{tabular}
```

Table 4: Source for multicolumn demonstration

```
\begin{table}[htbp]
  \caption{Multicolumn demonstration}
  \label{tab:multicol}
  \begin{center}
    \begin{tabular}[t]{lccc}
      \hline \hline
       &               & \multicolumn{2}{c}{Age in} \\
Person & Year of birth & 1969        & 1999         \\
\hline
Dad    & 1940          & 29          &  59          \\
Mom    & 1943          & 26          &  56          \\
Me     & 1969          &  0          &  30          \\
\hline
\end{tabular}
\end{center}
\end{table}
```

will display as:

Person	Year of birth	Age in 1969	Age in 1999
Dad	1940	29	59
Mom	1943	26	56
Me	1969	0	30

1.2.3 Other `tabular` tips

You can draw horizontal lines across the entire table with the `\hline` function or across several columns with `\cline{`i`-`j`}` function where i and j are column numbers.

You can stretch a narrow table to fill more of the page. LaTeX normally sets the width of the tabular environment to "natural" width, i.e., determined from the contents of the columns. For narrow tables it is sometimes more pleasing to make them wider. The tabular* environments allows for setting a width; however, it is necessary to have rubber space between columnns that can expand to the specified width. This can often be most easily accomplished by using an `extracolsep` command in an @ specifier as shown in the example below which sets the table width to 75 percent of the text width.

```
\begin{tabular*}{0.75\textwidth}{@{\extracolsep{\fill}}cccr}
label 1 & label 2 & label 3 & label 4 \\
\hline  % put a line under headers
item 1  & item 2  & item 3  & item 4  \\
\end{tabular*}
```

yields

label 1	label 2	label 3	label 4
item 1	item 2	item 3	item 4

1.3 Inserting graphics

LaTeX can include a variety of graphics. I have found Encapsulated PostScript easiest to include, but you can also include PostScript, `.gif` files, etc. The basic method is to use the package `graphicx` and then to use the `\includegraphics` command. This command can, but need not, appear inside the figure environment. The `\includegraphics` command offers various options for enlarging, shrinking, stretching, and rotating the graphic. Read Reckdahl (1997)'s excellent "Using Imported Graphics in LaTeX 2_ε" which is available in PostScript format from:

```
ftp://ftp.tex.ac.uk/tex-archive/info/epslatex.ps
ftp://ftp.dante.de/tex-archive/info/epslatex.ps
ftp://tug2.cs.umb.edu/tex-archive/info/epslatex.ps
```

or in PDF format from:

```
ftp://ftp.tex.ac.uk/tex-archive/info/epslatex.pdf
ftp://ftp.dante.de/tex-archive/info/epslatex.pdf
ftp://tug2.cs.umb.edu/tex-archive/info/epslatex.pdf
```

You can also read Section 3 of this document on the xfig application.

1.4 Packages/CTAN

LaTeX is a set of macros that use Donald Knuth's typesetting program TeX. One of the advantages of LaTeX is that it is (relatively) easy to extend. Many users worldwide have written packages that extend the functions of LaTeX. Some of these packages make one simple change to the functions of LaTeX. Others create a set of new environments or functions.[1] Some examples are `fullpage` (to widen all margins), `doublespace` (to control line spacing), `geometry` (to control document dimensions, e.g., margins), `url` (to insert URL's into documents), but packages range from offering more control over figure captions (`caption`, `ccaption`, `caption2`) through printing CD covers (`cdcover`) to notation for chess games (`chess`). The relevant package file typically ends in `.sty`. Sometimes the `.sty` file contains the documentation and sometimes it has accompanying text or dvi files that contain the documentation. Packages are easy to install in your own directories (be sure the directory is in the `TEXINPUTS` search path)

In most cases, the syntax to use the package is to include the following text in the preamble of the document (after the documentclass statement but before \begin{document}.

```
\usepackage[options]{package}
```

where options, if any, are specific to the package and discussed in the package documentation. If you are using several packages without options, you can include them in one line.

[1] When you introduce a package, you relinquish some of LaTeX's terrific portability; someone with whom you are sharing the source file must also have the package installed. Be aware of this as an important drawback of packages.

```
\usepackage{fullpage,doublespace,url,harvard}
```

Another contribution of LaTeX's generous and vast user base is new document classes (letter, article, report, and book are some of the document classes in the base LaTeX distribution). These files are typically named `.cls` or `.sty`. In this case, the invocation syntax is

```
\documentclass[option]{class}
```

The user-contributed documentclass for which I am most grateful is `ucthesis.cls` which handles all of the dissertation formatting (titlepage and other front matter, margins, etc.) required by Graduate Division at UC–Berkeley.

Many packages are maintained by the TeXUser Group. These are catalogued on-line with a search utility and brief descriptions and available for downloading at their web site, `http://www.tex.ac.uk/tex-archive/help/Catalogue/catalogue.html`, and its mirrors.

The Comprehensive TeXArchive Network (CTAN) of which this catalogue is a small part is an excellent source for LaTeX material, including free LaTeX distributions for various platforms. Several mirror sites for CTAN are `ftp://ftp.cdrom.com/pub/tex/ctan/` and `ftp://ftp.tex.ac.uk/tex-archive`

1.5 BibTeX

BibTeX is a wonderful bibliography manager for use with LaTeX. Erik Heitfield (PhD '98) wrote the following explanation of how to use BibTeX.

> This is an example of how to use bibtex, my favorite feature of LaTeX. To cite a reference, you use the "\cite" or "\citeasnoun" command. For example, you should read the complete description of TeX (Knuth 1990), and Lamport (1986) describes the LaTeX macro extensions. The previous sentence was generated with
>
> ```
> For example, you should read the complete description of \TeX\
> \cite{knuthtex}, and \citeasnoun{Lamport} describes the \LaTeX\ macro
> extensions.
> ```
>
> If you like referring to papers by their authors and dates (i.e. APA style), check out the "harvard.sty" package. (You must use the "harvard.sty" package to use "citeasnoun.") Finally, the "nocite" command is useful for making documents appear in your bibliography which are not cited in your document.
>
> Three files are required for a bibTeX document:
>
> 1. A .tex file containing a "bibliographystyle" command and a "bibliography" command.
> 2. A .bst file (referenced by the "bibliographystyle" command) which describes the format of bibliography entries. Several such files are included on the Suns, and hundreds more can be downloaded from ctan.

3. A .bib database file (referenced by the "bibliography" command) which contains information on the papers you want to cite.

To compile a bibTeX document, first run it through LaTeX 2$_\varepsilon$, then through bibTeX, and then through LaTeX 2$_\varepsilon$ a second (and maybe a third) time so all the references are resolved.

Here is a short excerpt from this document's associated bibliography file, tools.bib. Note that each entry: (1) names the type of publication; (2) includes a key word that is used to reference the entry in "cite" commands; and (3) contains the relevant bibliographic information, including your optional annotations.

```
@Manual{gnumake,
  title =        {GNU Make},
  OPTkey =       {},
  author =       {Richard M. Stallman and Roland McGrath},
  organization = {Free Software Foundation},
  OPTaddress =   {},
  OPTedition =   {},
  year =         {1988},
  OPTmonth =     {},
  note =         {Available from http://www.gnu.ai.mit.edu/software/make/make.html},
  OPTannote =    {}
}

@string{aw = "Addison--Wesley Publishing Company"}
@book{knuthtex,
                  author="Donald E. Knuth",
                  title="The {\TeX}book",
                  publisher=aw,
                  year=1990,
                  annote="This book is the definative reference on the
                  {TEX} document formatting language.  However, it
                  contains no information on \LaTeX\ extensions, and
                  is therefore of limited use to casual \LaTeX\
                  users."}
```

and here is a reprint of the lines that cause the bibliography to be printed at the end of this file. The following appear as the last lines of this file.

```
\bibliographystyle{/usr/local/texmf/bibtex/bst/kluwer}
\bibliography{tools}
\nocite{*}
```

I cannot emphasize enough how useful and time-saving I found BibTeX while writing my dissertation. There are utilities, many available for free that enable you to do more

sophisticated management of BibTeX databases. The emacs bibtex utility will allow you to alphabetize entries and do some simple management. There are also scripts available from the Comprehensive Perl Archive Network (CPAN, `ftp://uiarchive.cso.uiuc.edu/pub/lang/perl/CPAN/`) that will convert common database formats, e.g., EconLit and MedLine, into BibTeX entries.

1.6 `\label` and `\ref`

LaTeX keeps track of numbering sections, equations, list items, figures, and tables and can also manage reference to them at other points in the text. For example, you want to refer correctly to the table with your critical regression results whether it's Table 17 or Table 18. If you use emacs to edit your LaTeX files and to insert sections, figures, and tables (see Section 4.2), emacs will prompt you for labels. The basic labeling syntax is

```
\caption{This is my figure}
\label{fig:myfig}
```

or

```
\section{My section}
\label{sec:mysect}
```

It is not necessary to use "fig:" or "sec:" in the name, but I find it a useful mnemonic. To refer to the label, use the ref syntax:

```
As we shall see in Section \ref{sec:mysect}...
```

and LaTeX will insert the correct number each time it processes the file.

1.7 Fonts

LaTeX allows some control over fonts. You can choose the size of text. The following standard type size commands are supported by LaTeX. The commands as listed here are "declaration forms". The scope of the declaration form lasts until the next type style command or the end of the current group.

```
\tiny, \scriptsize, \footnotesize, \small, \normalsize, \large,
\Large, \LARGE, \huge, \Huge,
```

You can also use the environment form of these commands; e.g.

```
\begin{tiny}...\end{tiny}
```

You can also choose different typefaces, e.g., italic, boldface, sans serif, slanted. The following type style commands are supported by LaTeX. These commands are used so `\textit{italics text}` will give *italics text*. The corresponding command in parenthesis is the "declaration form", which takes no arguments. The scope of the declaration form lasts until the next type style command or the end of the current group.

The declaration forms are cumulative; i.e., you can say

```
\sffamily\bfseries
```

to get sans serif boldface. You can also use the environment form of the declaration forms; e.g.

```
\begin{ttfamily}...\end{ttfamily}.
```

```
\textrm (\rmfamily) Roman.
\textit (\itshape) \emph Emphasis (toggles between \textit and \textrm).
\textmd (\mdseries) Medium weight (default). The opposite of boldface.
\textbf (\bfseries) Boldface.
\textup (\upshape) Upright (default). The opposite of slanted.
\textsl (\slshape) Slanted.
\textsf (\sffamily) Sans serif.
\textsc (\scshape) Small caps.
\texttt (\ttfamily) Typewriter.
\textnormal (\normalfont) Main document font.
\mathrm Roman, for use in math mode.
\mathbf Boldface, for use in math mode.
\mathsf Sans serif, for use in math mode.
\mathtt Typewriter, for use in math mode.
\mathit Italics, for use in math mode, e.g. variable names with
several letters.
\mathnormal For use in math mode, e.g. inside another type style
declaration.
\mathcal Calligraphic letters, for use in math mode.
```

1.7.1 Fonts and Numbering in Section Headings and Captions

Because LaTeX does a good job maintaining the unity of structure of the document, it is slightly inflexible in allowing you to choose the format of these structures. The packages sectsty.sty and titlesec.sty available from CTAN (see Section 1.4) give some control over the look (fonts and numbering) of section headings. The packages caption.sty and ccaption.sty give some control over the look of figure or table captions.[2]

1.7.2 Text in math mode

Especially in economics, we may want to include text in equations. The `\mbox{text}` construction will keep text from being italicized and will preserve spaces. Also, some math functions involve text, e.g., log, exp, and sin. These should always be written with `$\log$`, `$\exp$`, and `$\sin$`

Compare

$$ln(Labor^{\alpha}Kapital^{\beta}) = \alpha lnLabor + \beta lnKapital \tag{1}$$

which was produced with

[2] In general, you can learn which package to use by searching the catalogue. In this case, the keyword "section" or "caption" would yield these packages. See Section1.4.

```
\begin{equation}
ln (Labor^\alpha Kapital^\beta) = \alpha ln Labor + \beta ln  Kapital
\end{equation}
```

and

$$\ln(\text{Labor}^{\alpha}\text{Kapital}^{\beta}) = \alpha \ln \text{Labor} + \beta \ln \text{Kapital} \tag{2}$$

which was produced with

```
\begin{equation}
  \ln (\mbox{Labor}^\alpha \mbox{Kapital}^\beta) =
  \alpha \ln  \mbox{Labor} + \beta \ln \mbox{Kapital}
\end{equation}
```

1.8 Mailing labels

LaTeX can be used the produce mailing labels that can be printed or xerographically reproduced onto self-adhesive labels. The document in Table 5 produces one sheet of 33 return address labels for me. The `labels` package is designed for use with standard mailing labels. You can learn more about mailing labels from the document, including how to read in a list of addresses, by reading the labels.dvi document:

```
% xdvi /usr/local/texmf/tex/labels/labels.dvi
```

Table 5: Return address labels—complete document

```
\documentclass{article}
\usepackage{labels}
\LabelCols=3%
\LabelRows=11%
\TopBorder=0in%
\BottomBorder=0in%
\LabelSetup
\numberoflabels=33
\begin{document}
\begin{labels}
Michael Ash
2238 Roosevelt, Apt B
Berkeley CA 94703-1722
\end{labels}
\end{document}
```

2 Some perl applications

There are many resources for learning perl, including Phil Spector's excellent presentation which is available in PostScript and PDF formats from `http://stat.berkeley.edu/~spector`. Rather than introduce perl, I want to show several ways I've applied perl to make my work easier. Please feel free to ask me about the code in these applications

The first application is used in conjunction with on-line data extraction from the Panel Study of Income Dynamics (`http://www.umich.edu/~psid`). The perl script (psidcode.pl) takes as input the variable list chosen by the user and generated by the PSID web site. The perl script then reads through *all* (16 codebooks representing 24 years of family and individual data) of the compressed PSID codebook stored as zip files on the EML and extracts only the chosen variable definitions. The script is currently available from `~mash/access/psidcode.pl` and is reprinted in Appendix A.

One of the most useful features of perl for me is that it can write LaTeX 2_ε code. I did this to mass produce my job market letters and I also use it to write Stata output to LaTeX 2_ε files. Remember that you need to use double backslashes (\\) when you print in `perl` to generate a single backslash in the LaTeX file `perl` generates. The following code, `stat2lat.pl` parses Stata output, extracts regressions results, and writes a LaTeX file that presents the regressions in a set of user-specified tables. For the time being, you can copy this code from `~mash/access/stat2lat.pl`, and the script is reprinted in Appendix B.

Another application I wrote parses *Job Openings for Economists* and writes addresses to a file which LaTeX can use for mailing labels and another perl script can use to address form letters. This application is reprinted in Appendix C and is available from `~mash/jobmarket/joe.pl`.

3 xfig

Unix offers an excellent drafting program called `xfig` which you can use to draw and label diagrams, e.g. indifference curves, phase diagrams, market equilibria, etc. The program has a mouse-driven graphical user interface with a helpful menu. It also has a nice snap-to-grid feature; it's easy to edit, move, and align objects; etc. Documentation on xfig is available from the xfig home page, `http://www.xfig.org`, and from `http://duke.usask.ca/~macphed/soft/fig/index.html`

Figures are saved in simple ASCII files with the suffix .fig.[3] You can export xfig figures to various formats, including Encapsulated PostScript and other formats suitable for import to LaTeX. The easiest way import an xfig figure (myfig.fig) to LaTeX is to use the menu to export it from xfig as an Encapsulated PostScript file (myfig.eps) and then to import it to LaTeX with `\includegraphics{myfig.eps}`. Remember that you must `\usepackage{graphicx}` in the preamble to use this command.

It is possible to embed LaTeX code in the text of an xfig figure; so you can use a full range of symbols, subscripts, superscripts, etc. Trying to insert sub/superscripted characters by hand is incredibly tedious and frustrating. To embed LaTeX code in an xfig figure, you should

[3] In fact, you can learn how to write to and to manipulate these files directly.

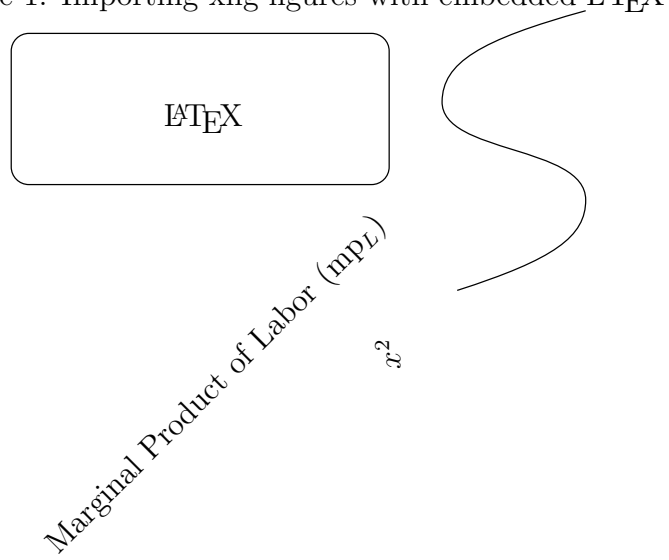
launch xfig with the following options:

```
% xfig -specialtext -latexfonts -startlatexFont default
```

Draw your diagram and save it as a `.fig` file. Then, export the diagram in two steps: (1) `Combined PS/LaTeX (PS part)`; and (2) `Combined PS/LaTeX (LaTeX part)`. There will be two exported files that LaTeX will overlay: myfig.pstext_t (the parts of the figure that contain the LaTeX code) and myfig.pstex (containing the rest of the figure). Be sure to keep these in the same directory as the LaTeX document. Then, in LaTeX, use the command \inputmyfig.pstext to insert the diagram where you want it. You may want to put the diagram inside a figure environment to give it a number and a caption. The resulting figure may look imperfect when you view the `.dvi` file with xdvi but should be correct in PostScript. Here is a very short example (not so helpful since you cannot see the associated `.fig` file).

```
\documentclass{article}
\usepackage{graphicx}
\begin{document}
\begin{figure}[htbp]
  \begin{center}
    \caption{Importing xfig figures with embedded \LaTeX}
    \label{fig:xfigtest}
    \input{ex.pstex_t}
  \end{center}
\end{figure}
\end{document}
```

Figure 1: Importing xfig figures with embedded LaTeX



4 emacs

Emacs is an editor and a lot more. Versions of emacs are available, free for many platforms, including DOS, Windows, and Macintosh. At the EML, you start emacs by typing emacs at the unix prompt:

```
% emacs
```

which gives a nice window interface if you are hooked up or a text interface if you are not. If you are at a workstation but do not want the window interface, you can use the `-nw` flag. You can get an especially user-friendly gui version of emacs by typing xemacs at the unix prompt:

```
% xemacs
```

4.1 Learning emacs

tutorial You should begin your career with emacs by running through the tutorial. The tutorial will explain how to open, save, and close documents, move through documents, and edit text. First, copy the essential lines of my `.emacs` file (See Table 6 that assign help and backspace. Then, start emacs and type "Control-x ? t"

help Emacs has an excellent on-line help facility (of which the tutorial is a small part). You can invoke the help facility with "Control-x ?" which offers a menu of options.

info The best part of the on-line help is complete, menu-driven documentation. You can invoke this with "Control-x ? i" and then navigate the menu with arrow keys.

quitting emacs You can quit emacs with "Control-x Control-c".

quitting command line You can stop the current command if you get caught on the command line with "Control-g".

4.2 Advantages of emacs

The advantages of emacs are myriad. You can read e–mail and usenet groups, learn the islamic and hebrew calendar dates, and even get psychoanalyzed by emacs (try "Meta-x doctor"). Here are some of the features that I use most. It's also widely available and free to install if you don't find it.

dired and ftp Emacs has a nice built in file manager that will move, copy, and erase files. You can invoke it with "Meta-x dired" or "Control-x d". The file manager is not limited to the EML. You can find and manage files on remote computers too. To open a file on a remote computer, use the standard find file sequence, "Control-x Control-f" and when prompted for the file name, enter `/userid@remotehost:` You will be prompted for your password on the remote host and can then proceed to move, copy, open, erase, etc. files.

latex–mode Emacs has many modes that are designed to facilitate editing files of different types. For example, in latex–mode, emacs uses indentation, color, highlighting, and italics to make LaTeX files easier to read and edit. The latex-mode also allows you to process LaTeX files from within emacs, to view dvi files, to BibTeX files, to insert environments with several keystrokes or from the menu bar. For example, the emacs command "Control-c Control-e" will prompt you to start a document and choose a class (if you haven't already) and, thereafter, to insert an environment, etc. The BibTeX assistance in emacs is also excellent.

SAS, Stata, Splus, perl, java, c You can edit and *run* programs for these and other applications from within emacs, and emacs again uses indentation, color, highlighting, and fonts to facilitate editing. The basic development tools come with many emacs packages, including ours at the EML. For Splus and SAS, you need only include the following two lines in your `.emacs` file:

```
; Use ESS to edit and run SAS, Splus
(require 'ess-site)
(require 'essd-sas)
```

Stata will soon be included in this utility.[4] If you want to learn what's available for Stata immediately, please copy `~mash/stata.el` to your home directory, read the first paragraphs of `stata.el`, and insert the following lines in your `.emacs` file (you can copy my `.emacs` file):

```
; Use stata.el to edit and run Stata
(autoload 'stata "~/stata.el" "inferior stata mode" t )
(autoload 'stata-help "stata" "stata help mode" t)
(autoload 'stata-mode "~/stata.el" "stata mode" t)
(if (assoc "\\.do$" auto-mode-alist) nil
  (setq auto-mode-alist
        (append
         '(("\\.do$" . stata-mode)
           ("\\.ado$" . stata-mode))
         auto-mode-alist)))
```

Version control If you are serious about your programming, you need to keep a history of your work. For example, you may want to revert to the last-working-copy of a program you are building. Version control allows you to track changes in a program over time without keeping every version of the program in a separate file. And emacs has a fine interface with several standard version control programs. I use RCS. You can read more about it in emacs info or in the RCS man pages (`man rcs`)

[4]There are many modes available for emacs (and you can write your own). For example, a Matlab mode is available from Mathworks, `http://www-europe.mathworks.com/ftp/emacs_add_ons.shtml`.

Manipulate rectangles of text One of the coolest things about emacs is that it will manipulate (cut, copy, paste, or erase) rectangles of text. The basic technique is to set the mark at the upper-left corner of the rectangle that you want to move and then move the cursor to the lower right corner of the rectangle you want to move. "Control-x r k" will cut the rectangle and "Control-x r y" will paste it. Alternatively, "Control-x r r R" will copy the rectangle to register R and "Control-x r i R will past the rectangle from register R.

Spell-checking Emacs has a spell-checking facility. You can spell-check the entire document with "Meta-x ispell-buffer" or a word with "Meta-x ispell-word". The ispell facility is pretty smart about LaTeX codes and won't ask you if they are misspelled words.

diary Emacs can maintain a diary and warn you of upcoming appointments and anniversaries. I recommend reading the info section on the diary. My `.emacs` file (see Table 6 includes several commands that facilitate the use of the diary.

4.3 .emacs

You can customize emacs extraordinarily completely and easily by creating a `.emacs` configuration file, called "dot emacs," file in your home directory. The `.emacs` file contains instructions that are executed every time you start emacs. For example, you may reassign keys for your convenience, you may change the background color and the size of the font.

I am including my `.emacs` file in Table 6, fully commented, so that you can get an idea of what's possible. You may also copy my `.emacs` file to your home directory with

```
cp ~mash/.emacs  ~/.emacs
```

5 Odds and Ends

5.1 less

The best way I know to look at files is the utility `less`, a riff on `more` with which you may be more familiar. The usual way to invoke this command is `less <filename>`, but you can also pipe output to it, e.g., `finger @socrates | less`.

The two best things about less are the ability to scroll smoothly backwards and forwards and the ability to search for text. Some useful commands are printed in Table 7, but less does a lot more. See help or `man less` for details.

5.2 Choice of shell

In unix, you can choose the shell, or the program that permits you to interact with the operating system. You can use the shells interactively at the unix prompt or in shell-scripts. The default shell is csh. I recommend changing to either the `tcsh` or `bash` shell. Both of

Table 6: Michael Ash's `.emacs` file

```
; Comments are preceded by ;
; Be sure to include the next four lines in your .emacs
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; Define control-x ? as help
(define-key global-map "\C-x?" 'help-for-help)
;; Define control-h as backspace
(define-key global-map "\C-h" 'backward-delete-char)
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; Auto-fill (line-wrapping) in text-mode
(setq text-mode-hook 'turn-on-auto-fill)
; View of long lines to be wrapped, not truncated
(setq truncate-lines nil)
; Show line number in status-line
(setq line-number-mode 'on)
; Show column number in status-line
(setq column-number-mode 'on)
; Version control prompts for initial comment
(setq vc-initial-comment t)
; Show diary entries on start-up
(setq view-diary-entries-initially t)
; Show diary entries for today and tomorrow
(setq number-of-diary-entries 2)
; Load diary
(diary)
; Prepare diary to print in \TeX
(setq cal-tex-diary t)
; Highlight selected area
(transient-mark-mode t)
; Allow the downcase and upcase region commands
(put 'downcase-region 'disabled nil)
(put 'upcase-region 'disabled nil)
; Set background color to cyan
;(set-background-color "cyan")
```

Table 7: Getting started with less

h	invokes help.
q	quits from less.
b	scrolls backwards (space bar scrolls forwards as in `more`).
/	gives a prompt to search for text or a regular expression.
–i	toggles between case insensitive and case sensitive searches. You can use this in the command line, `less -i <filename>` or while viewing a file—don't forget the –.
–S	toggles between wrapping lines of text and letting them stretch. Again, you can use this in the command line or while viewing a file. This option is very useful for looking at raw data.
Shift–<	jumps to the beginning of the file.
Shift–>	jumps to the end of the file.

these shells allow you to edit the command line if you have made a mistake and to cycle through commands you've given previously with the arrow keys. You can change the current shell any time you want by entering tcsh or bash at the prompt. You can change your default shell by entering `chsh` (`ch`ange `sh`ell) at the command prompt. When prompted, enter your password; then enter /bin/tcsh or /bin/bash.

5.2.1 .cshrc

You can give the computer a set of commands to customize your workspace by putting them in your `.cshrc` file. My current, fully-commented `.cshrc` is reproduced in Table 8 and can be copied from `~mash/.cshrc`.

5.3 unix input and output

> redirects the output of a command to a file that you name, e.g.,

```
% finger @socrates > whosonsocrates
```

will send the output of `finger @socrates` to the file `whosonsocrates` instead of to the screen. The pipe symbol, |, redirects the output of a command to another command, e.g.,

```
% finger @socrates | less
```

will send the output of the command to the less file-viewing utility (see Section 5.1).

5.4 Regular Expressions

Regular expressions provide a mechanism to select specific strings from a set of character strings. The tools for searching and replacing regular expressions are extremely powerful.

Table 8: Michael Ash's `.cshrc` file

```
# Pound sign (#) indicates comment line
# source the standard .cshrc file
source  /usr/local/skel/std.cshrc
# set the path
set path = ( $path ~/census /usr/etc )
# set the prompt so it shows the current host and directory
set prompt = "%m:%~%% "
# make ls show a symbol for directories/ and executables*
alias ls ls -F
# make lsmine give a full listing of only my files--useful in /tmp/
alias lsmine 'ls -alF | grep $user'
# make rm prompt before removing.  \rm overrides
alias rm rm -i
# make mv prompt before moving onto an existing file.  \mv overrides
alias mv mv -i
# make cp prompt before overcopying.  \cp overrides
alias cp cp -i
# make dvips put postscript file in /tmp (instead of straight to printer)
alias dvips 'dvips -o /tmp/\!#:$:r\.ps \!$'
# make grelp search help files for string
alias grelp 'help- -l | grep -i \!*'
# use the old help facility
alias help  'help-'
# make connection to socrates
alias socrates 'ssh socrates -l maash'
alias desktop '/usr/local/x11/lib/xdm/Xsession.ow35&'
# Choose editor emacs
setenv EDITOR emacs
# Choose file viewer less
setenv PAGER less
# Expand TEXINPUTS path
setenv TEXINPUTS ".:/usr/local/texmf/tex/INPUTS:~mash/tex"
```

The ability to manipulate regular expressions is incredibly helpful. See Table 9 for an example.

Table 9: Example of replace-regexp in emacs

For example, if you are trying to convert SAS scripts into Stata do files, you may want to change many instances of

```
if xxx = XXX then yyy = YYY ;
```

into

```
replace yyy = YYY if xxx == XXX
```

where `xxx` and `yyy` are variable names and `XXX` and `YYY` are numbers that vary in each instance. With the replace-regexp function in emacs, you can achieve this with few keystrokes. Invoke the function with "Meta-x replace-regexp". When prompted with "Replace regexp:", enter

```
if \([a-z]+\) = \([0-9]+\) then \([a-z]+\) = \([0-9]+\)
```

Finally, when prompted with "with:", enter

```
replace \3 = \4 if \1 == \2
```

I use them in four contexts: in perl scripts; in the less viewing program (see Section 5.1 for more about less); in emacs for search/replace commands; and from the command line.

Sources of terse, on-line information: man regexp, man perl, man grep.

5.5 Finding files

We typically use the `ls` command to list the files in a directory. The `du` command is good for this purpose when used with the `-a` flag but `du` also looks for files down the entire directory tree.

```
% du -a
```

You can combine this technique with searching for regular expressions Section 5.4 and use a construction such as:

```
% du -a | grep '\.sas'
```

to find all sas files (all the files with the `.sas` extension) in your directory structure.

5.6 ghostview, acroread, and xv

The latest version of ghostview is invoked with `gv` and allows you to view and print both PostScript and PDF files. Acroread is better for viewing and printing PDF files, e.g., the hyperlinks are active for acroread but not ghostview. The program xv is good for viewing and processing `.jpeg` or `.gif` graphics files.

5.7 psnup

Here is a way to pack more pages on a page when you are printing at the EML. Advantages: fewer printed pages (save paper and money); view more material in front of you at once. Disadvantages: smaller text, hard on eyes.

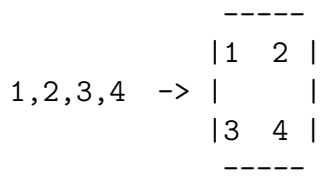
1. Begin with a PostScript file (which I will call oldpsfile.ps below). You may download material from the web in PostScript; you may produce PostScript from LaTeX/dvi files with dvips or from other programs; or you may choose a PostScript printer and "print" PostScript files from various word processors, e.g., Word, WordPerfect. See note 5 on converting PDF/Acrobat files to PostScript below.

2. Use `psnup` (Read: "ps" = PostScript "n up" = n pages on each page). You can read the manual page on psnup with "man psnup," but here's a primer that contains all you probably need. (You can substitute any name you want for newpsfile4 or newpsfile 2L. I use them below as mnemonics for 4 on a page, 2 on a landscape page.)

   (a) To put 4 portrait pages on one 8.5"×11" portrait page (as in Figure 2), use the command,

   ```
   % psnup -n4 -p newpsfile4.ps oldpsfile.ps
   ```

   Figure 2: Placing PostScript pages 4–up

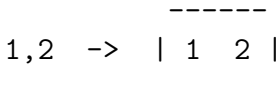
   

   (b) To put 2 portrait pages onto one 11"×8.5" landscape page (as in Figure 3), use the command,

   ```
   % psnup -n2 -r -p newpsfile2L.ps oldpsfile.ps
   ```

   NOTES: The `-n<number>` option tells psnup how many on a page. You can make this number large, but the text will get smaller. The `-r` option in part 2b tells psnup to use landscape format, which makes more sense if you are putting 2 portrait pages on one page.

Figure 3: Placing PostScript pages landscape, 2–up



3. (Optional) You can view your masterpiece with ghostview, invoked with the command gv (also available on PC):

   ```
   gv newpsfile4.ps
   gv newpsfile2L.ps
   ```

4. You can print from gv (see step 3) or with directly with the lpr command:

   ```
   lpr newpsfile4.ps
   lpr newpsfile2L.ps
   ```

5. (PDF/Acrobat) Lots of times you may download PDF files from the web, e.g., articles from the JSTOR archive, NBER Working Papers. Typically you view these with acroread. To print these several pages to a page, you must first convert them to PostScript. The command below translates mypdffile.pdf to oldpsfile.ps, and then you can use psnup (see steps 1–4 above) on oldpsfile.ps

   ```
   % acroread -toPostScript -pairs mypdffile.pdf oldpsfile.ps
   ```

5.8 enscript

A good way to print text files on unix systems is with the `enscript` utility which has lots of cool options. For example, `-2r` prints the named file in landscape with two columns and the `-G` option prints a gaudy header with you name on it at the top of each page.

```
% enscript -2r -G filename
```

By default `enscript` sends the output straight to the printer. If you want to send the output to a file instead (for examination by `ghostview` before printing), try

```
% enscript -2r -G -p /tmp/newfilename.ps filename
```

6 lynx and The Web

Lynx is a text-based web browser: no pictures, but it loads much faster than Netscape and other gui browsers. I recommend trying it out if you are tired of waiting for Netscape to load. (You can download pictures and view them with xv if you really want to see them.)

Here are some archives that our `berkeley.edu` domain makes available to us. These sites would be extremely expensive otherwise and are often useful. My web site, `http://socrates.berkeley.edu/~maash` for the moment, contains links to many other text and data archives.

`http://www.jstor.org`	Complete, full-image collections of many journals, economics and otherwise.
`http://www.lexis-nexis.com/universe`	Premier news and legal archive
`http://www.eb.com:180/`	Encyclopaedia Britannica
`http://webspirs.silverplatter.com/cgi-bin/customers/ucb/ucb2b.cgi`	EconLit

Another useful web trick is that you can ftp (file transfer protocol, used here as a verb) datasets within shell-scripts, make files (see Section 7), or SAS/Stata scripts. For example, to do this in SAS, include the line:

```
x "ncftp -f emlab.berkeley.edu:/pub/data/89raw.txt.Z;" ;
```

to ftp the named file to the directory from which you launched SAS. By the way, I recommend `ncftp`, which is a souped-up version of ftp, both interactively and in scripts.

7 make

The program make is a versatile project management tool. Make allows the user to specify a series of "dependencies" among files so that you can be sure that your data and output are up to date.[5]

Suppose that in the course of your research, you ftp several years of Current Population Survey data from `emlab.berkeley.edu` to /scratch/public. You then read the data into several SAS datasets. You then convert the SAS datasets into Stata datasets (see Section 9), merge it with some data that are already in Stata format, and perform some statistical procedures in Stata. You don't want to waste time (not to mention file access and bandwidth) by repeating the particularly time-consuming parts of the process, e.g., the file transfers and inputs of raw data.

Create a file called Makefile in your project directory. Makefile should contain dependency lines and command lines. The dependency lines state that the file[6] to the left of the colon *depends* on the files to the right of the colon. If the right-hand files are more recent than the left hand files, then the set of commands in the block of command lines following the dependency line will be executed. Note: each of the command lines must begin with a TAB character.

```
# If the Stata output has changed, extract and label the regression results
```

[5] make was originally designed for software development, but I think that it meets the needs of empirical economists extremely well.

[6] In more complicated uses, this is not limited to being a file.

```
# with the homegrown perl scripts, run LaTeX on the extracted
# results, and convert the .dvi file to a PostScript file.
final.ps : final.output labelvars.pl
        stat2lat.pl < final.output | labelvars.pl > final.tex
        latex2e final.tex
        dvips -p/tmp/final.ps final.dvi

# If either the
# merged dataset or the do-file for the statistical procedures
# has changed, run the statistical procedure in Stata.
final.output : final.do final.dta
        stata -b do final.do

# If any of the input data for the merged data file has changed,
# rerun the merging do-file.
final.dta : cps79.dta cps89.dta otherdat.dta
        stata -b do mergeall.do

# If the do-file that creates the other data has changed,
# recreate the other data.
otherdat.dta : otherdat.do
        stata -b do otherdat.do

# If the SAS program that reads the raw 1979 CPS data has changed, rerun
# this program and convert the dataset to Stata format.
cps79.dta : cps79.sas
        sas cps79.sas
        sas2stata -f -r cps79.ssd01

# If the SAS program that reads the raw 1989 CPS data has changed, rerun
# this program and convert the dataset to Stata format.
cps89.dta : cps89.sas
        sas cps89.sas
        sas2stata -f -r cps89.ssd01

# If the 1979 raw data has disappeared, e.g., because scratch has been
# cleaned, ftp it from emlab again.
cps79.sas : 79raw.txt
        ncftp -f emlab.berkeley.edu:/pub/data/79raw.txt
        touch 79raw.txt

# If the 1989 raw data has disappeared, e.g., because scratch has been
# cleaned, ftp it from emlab again.
cps89.sas : 89raw.txt
```

```
ncftp -f emlab.berkeley.edu:/pub/data/89raw.txt
touch 89raw.txt
```

Now if you type `make final.ps`, make will read Makefile and perform *only* and *all* the tasks required to produce an up-to-date version of final.ps, going all the way back to raw data if necessary. Make is capable of much more than I've described. There is an excellent book on make called *Managing Projects with Make* (Oram and Talbott 1991) published by by O'Reilly and Associates[7] which costs about $20, and a very good document called "GNU Make" (Stallman and McGrath 1988) available with the most current distribution of `make` from the Free Software Foundation at `http://www.gnu.ai.mit.edu/software/make/make.html`.

8 A mailfilter

If you subscribe to active mailing lists, e.g., Statalist, you may want to sort your e-mail to different inboxes.

1. Create a mailfilter file, e.g., `.mailfilter`, modeled on Table 10. The filtering based on choosing a text string that will appear only in mail intended for a particular inbox, e.g., "statalist". Make this `.mailfilter` file executable with `chmod +x .mailfilter`

2. Edit your `.pinerc` file to identify all of your inboxes. Table 11 contains the portion of `.pinerc` that must be modified.

3. Create a `.forward` file that pipes all of your mail to your mailfilter. Your file `.forward` file should contain: `|/accounts/grad/userid/.mailfilter`

9 Database Conversion

1. The unix utility `sas2stata` on the EML will convert unix SAS datasets (*.ssd01) to all-platform Stata (*.dta) datasets, keeping variable names, lengths, and labels intact. Thus,

   ```
   % sas2stata mydata.ssd01
   ```

   will create mydata.dta; For brief or detailed documentation, you can try respectively:

   ```
   % sas2stata
   % man sas2stata
   ```

[7] O'Reilly and Associates have many excellent books on some of the tools discussed in this document. In particular, keep an eye out for their *... in a Nutshell* series

Table 10: .mailfilter filters mail into 4 incoming folders

```
#!/bin/sh
PATH=/bin:/usr/bin:/usr/ucb
export PATH
user=mash
if [ "`whoami`" != "$user" ]; then
        exit 1
fi

mailbox=/var/spool/mail/$user
home=/srv/accounts/grad/$user
penmail=$home/mail/penmail
stata=$home/mail/stata
datalist=$home/mail/datalist
tmp=$home/.tmp

cat - > $tmp
if grep -s -i "lbo-talk" $tmp
then
        sed -e '2,$ s/^From />From /' $tmp >> $penmail
        echo >> $penmail
elif grep -s -i "statalist" $tmp
then
        sed -e '2,$ s/^From />From /' $tmp >> $stata
        echo >> $stata
elif grep -s -i "sas-l" $tmp
then
        sed -e '2,$ s/^From />From /' $tmp >> $datalist
        echo >> $datalist
elif grep -s -i "saspac" $tmp
then
        sed -e '2,$ s/^From />From /' $tmp >> $datalist
        echo >> $datalist
elif grep -s -i "labor-data" $tmp
then
        sed -e '2,$ s/^From />From /' $tmp >> $datalist
        echo >> $datalist
else
        sed -e '2,$ s/^From />From /' $tmp >> $mailbox
        echo >> $mailbox
fi

rm -f $tmp
exit 0
```

Table 11: Name incoming folders in `.pinerc`

```
# incoming-folders are those other than INBOX that receive new messages.
# Folder syntax: optnl-label {optnl-imap-hostname}folder-path
# Use only if you filter incoming email into multiple files or receive
# email on several different machines.
# Example:
# incoming-folders=Consulting {carson.u.washington.edu}filter/to-help,
#                  Widget-Project{carson.u.washington.edu}filter/to-widget,
#                  Old-Student-Acct {imap.berkeley.edu}inbox
# Michael Ash's incoming folders:
incoming-folders=penmail  /srv/accounts/grad/mash/mail/penmail,
                 stata    /srv/accounts/grad/mash/mail/stata,
                 datalist /srv/accounts/grad/mash/mail/datalist
```

Note that sas2stata runs both SAS and Stata as well as some of the classic unix utilities like awk or sed in the process of writing the dataset. So you are limited to UNIX systems that have BOTH applications, e.g., the EML but not `socrates.berkeley.edu` (SAS but not Stata). I think that it sas2stata is available free from the RAND Corporation if you want to install it on your own UNIX system.

2. On the EML (or for PCs if you buy it), you can also use

   ```
   % dbmscopy
   ```

   an interactive utility that will do lots of cross-program dataset conversions, e.g., PC SAS ↔ unix SAS ↔ Stata ↔ Excel ↔ Lotus 1–2–3, etc. After you run dbmscopy interactively several times, you can learn the syntax to use `dbmsnox`, a conversion program command-line that runs from the command line. For example,

   ```
   % dbmsnox /tmp/mydata.dbf /tmp/mydata.stata4
   ```

   will convert dBase file mydata.dbf into Stata file mydata.dta.

3. For PCs, you can buy Stat/Transfer, a utility sold (but not written) by Stata Corporation (`http://www.stata.com`). The academic price is low (c. $50), and it does lots of cross-program dataset conversions. I think this is well worth it if you buy the Stata package for the PC.

A psidcode.pl

```
#!/usr/local/bin/perl

# psidcode.pl
# michael ash
# march 1997
# Bug reports to mash@econ.berkeley.edu

# Main use: Parses the data-center file created during the creation
# of a PSID data set at \url{www.umich.edu/~psid} for
# year, level (individual or family), and variable name (V#####).
# Writes codebook for those variables.

# NonEML users should make sure that the documentation directory
# is properly specified.

# Reads any input file with rows containing
# Year Level Variable
# in that order, e.g.,
# 1984 Family V10263

# Reads "data-center" in current directory
# Reads zipped PSID documentation in /archive/psid_all/documentation
# Writes "codebook" in current directory

# Glitches:
# includes page breaks and page headers from the PSID codebooks.
# Selecting the last variable in a year may cause too much output.



open(VLIST,"<$ARGV[0]") ;
open(CODEBOOK,">codebook") ;

# Read included variables and individual or family.
# If family, read year too.

while ($line = <VLIST>) {
    chop($line) ;
    $line =~ s/^\s+// ;
    ($year,$level,$vname,@junk) = split(/\s+/,$line) ;
    $level =~ s/^\s+// ;
    $level =~ s/\s+$// ;
    $vname =~ s/^\s+// ;
    $vname =~ s/\s+$// ;
    $vname =~ s/^V// ;
    if ($level eq "Individual") {
        $flist{ind} = "unzip -c /archive/psid_all/documentation/68-92doc.zip |" ;
        $vlist{ind} .= "$vname:" ;
    }
    elsif ($level eq "Family") {
        if ($year >= 1968 & $year <= 1978) {
            $flist{fam6878} = "unzip -c /archive/psid_all/documentation/68-78doc.zip |" ;
            $vlist{fam6878} .= "$vname:"
            }
        if ($year == 1979) {
            $flist{fam79} = "unzip -c /archive/psid_all/documentation/79doctxt.zip |" ;
            $vlist{fam79} .= "$vname:"
            }
        if ($year == 1980) {
            $flist{fam80} = "unzip -c /archive/psid_all/documentation/80doctxt.zip |" ;
            $vlist{fam80} .= "$vname:"
            }
        if ($year == 1981) {
            $flist{fam81} = "unzip -c /archive/psid_all/documentation/81doctxt.zip |" ;
            $vlist{fam81} .= "$vname:"
```

```
            }
        if ($year == 1982) {
            $flist{fam82} = "unzip -c /archive/psid_all/documentation/82doctxt.zip |" ;
            $vlist{fam82} .= "$vname:"
            }
        if ($year == 1983) {
            $flist{fam83} = "unzip -c /archive/psid_all/documentation/83doctxt.zip |" ;
            $vlist{fam83} .= "$vname:"
            }
        if ($year == 1984) {
            $flist{fam84} = "unzip -c /archive/psid_all/documentation/84doctxt.zip |" ;
            $vlist{fam84} .= "$vname:"
            }
        if ($year == 1985) {
            $flist{fam85} = "unzip -c /archive/psid_all/documentation/85doctxt.zip |" ;
            $vlist{fam85} .= "$vname:"
            }
        if ($year == 1986) {
            $flist{fam86} = "unzip -c /archive/psid_all/documentation/86doctxt.zip |" ;
            $vlist{fam86} .= "$vname:"
            }
        if ($year == 1987) {
            $flist{fam87} = "unzip -c /archive/psid_all/documentation/87doctxt.zip |" ;
            $vlist{fam87} .= "$vname:"
            }
        if ($year == 1988) {
            $flist{fam88} = "unzip -c /archive/psid_all/documentation/88doctxt.zip |" ;
            $vlist{fam88} .= "$vname:"
            }
        if ($year == 1989) {
            $flist{fam89} = "unzip -c /archive/psid_all/documentation/89doctxt.zip |" ;
            $vlist{fam89} .= "$vname:"
            }
        if ($year == 1990) {
            $flist{fam90} = "unzip -c /archive/psid_all/documentation/90doctxt.zip |" ;
            $vlist{fam90} .= "$vname:"
            }
        if ($year == 1991) {
            $flist{fam91} = "unzip -c /archive/psid_all/documentation/91doctxt.zip |" ;
            $vlist{fam91} .= "$vname:"
            }
        if ($year == 1992) {
            $flist{fam92} = "unzip -c /archive/psid_all/documentation/92doctxt.zip |" ;
            $vlist{fam92} .= "$vname:"
            }
    }
}
close(VLIST) ;


# File loop: individual cross-year file and each family year file
foreach $j (sort(keys(%flist))) {
    $curfile = $flist{$j} ;
    @vars = split(/:/,$vlist{$j}) ;
    @varno = sort {$a <=> $b} @vars ;
    print "\n\nRead $j: @vars\n" ;
    print "Sorted $j: @varno \n\n" ;

# Variable loop within file
    $i = 0 ;
    $curvar = @varno[$i] ;
    open(CURFILE,"$curfile") ;


    if ($j eq fam6878) {
        while ($line = <CURFILE>){
          CLABEL: {
```

```
            while ($curvar<1100){
            $curvar = @varno[$i] ;
                if ($line =~ /^\s*$curvar/) {
                    print "Found $curvar.\n";
                    $i++ ;
                    print CODEBOOK $line ;
                    $k=0;
                    until ((($line=<CURFILE>) =~ /^\s{0,4}[0-9]/) || ($k==1000)) {
                        print CODEBOOK $line ;
                        $k++ ;
                    }
                    goto CLABEL ;
                }
                $line = <CURFILE> ;
            }
        }

        BLABEL: {
            if  ($i <= $#varno) {
                $curvar = @varno[$i] ;
                if ($line =~ /^\s*\($curvar\)/) {
                    print "Found $curvar.\n";
                    $i++ ;
                    print CODEBOOK $prevline ;
                    $k=0 ;
                    until (( ($prevline=$line) && (($line=<CURFILE>) =~ /^\s{0,4}[0-9]/) || ($k==1000) )){
                        print CODEBOOK $prevline ;
                        $k++ ;
                    }
                    goto BLABEL ;
                }
                else {
                    $prevline = $line ;
                    $line=<CURFILE> ;
                    goto BLABEL ;
                }
            }
        }
      }
    }


    if ($j ne fam6878) {
        while ($line = <CURFILE>)   {
          ALABEL: {
            if ($i <= $#varno) {
                $test = $line ;
                $test=~s/ V/ / ;
# Check if codebook should include variable by comparing it to next in
# the list of variables
                if (($test =~/^\s+$curvar/) && (($test=~/TLOC=/) ||
                                                ($test=~/Name=/))){
                    print "Found $curvar.\n";
                    $i++ ;
                    $curvar = @varno[$i] ;
# Read and write all codebook lines until reach the next variable
                    print CODEBOOK $line ;
                    $k=0 ;
                    until ((($line = <CURFILE>)=~/TLOC=/) || ($line=~/Name=/)) || ($k==1000)){
                        print CODEBOOK $line ;
                        $k++ ;
                    }
                    goto ALABEL ;
                }

            }
        }
```

```
        }
    }
    close (CURFILE) ;
}
close(CODEBOOK);
```

B stat2lat.pl

```
#!/usr/local/bin/perl
# Parses Stata output writing regression results as LaTeX tables.
# Michael Ash
# 20 June 1997

# stat2lat.pl [-a][-p][-t][-l] statalogfile > LaTeXfile

# options
# -a   Write to align on decimal point, default is centered
# -l   Write LaTeX longtables
# -p   Write p-values instead of standard errors
# -t   Search log file for table numbered TABLE 1 - TABLE N
#      If you want to sort your regressions into N different tex
#      tables, then in the logfile, in the line before each regression
#      that you want to keep, label the line TABLE i where i is the
#      table number to which you want to assign this particular
#      regression. (They don't have to be in order, but there does
#      need to be a TABLE 1.)  The script will use the rest of the
#      line after the first instance TABLE i as the caption for the
#      table i, e.g.
#      TABLE 1 Wage Regressions
#      . reg lwage x1 x2
#      <output>
#      TABLE  2 Hour Regressions
#      . xi: reg hour x1 x3 x4 i.x6
#      <output>
#      TABLE 1    This sentence gets ignored.
#      . reg lwkwage x1 x3 x5
#      <output>

require 'getopts.pl' ;
&Getopts(":alpt") ;

print("\\documentclass{article}[12pt] \n
\\usepackage{amstex,fullpage,geometry,longtable}
\%\\setlongtables
\%\\geometry{body={8in,10in}}
\\begin{document}") ;



# Regression marker
# Put regressions in designated tables
if ($opt_t) {
    $table = 1 ;
}
# Else put all regressions in one table
else {
    $regind = '\..*reg' ;
}


$tableflag = 1 ;
while ($tableflag == 1) {
    if ($opt_t) {
        $regind = "TABLE $table";
    }
    open(LOGFILE,"<$ARGV[0]") ;
```

```
# Initialize table
    undef @vars ;
    undef %beta ;
    undef %se ;
    undef $R2 ;
    undef $N ;
    undef $Dpndt ;
    $tableflag = 0 ;
    $R2 = "\\\\[1mm] \\hline \\\\[1mm] \n\$R^2\$ " ;
    $N = "\nN" ;
    $reg = 0 ;

    while ($line = <LOGFILE>) {
        if ($line =~ /${regind}[^0-9]/) {
        chop($line) ;
            if ($table >= 1) {
                # Include note with dependent variable
                $tableflag = 1 ;
                $note = $line ;
                $note =~ s/^.*$regind// ;
                $note =~ s/^\s+// ;
                $note =~ s/\s+$// ;
            }
            $reg++ ;
            $r2flag = 0 ;
            $nflag = 0 ;
            $listind = "" ;
            # Read in R2 and N from the regression output
            until (($line = <LOGFILE>)=~ /-{78}/) {
                chop($line) ;
                if (($line =~ /R-sq/) && ($line !~ /Adj/)){
                    $r2flag = 1 ;
                    ($junk,$r2) = split(/R-sq/,$line) ;
                    ($junk,$r2) = split(/=/,$r2) ;
                    $r2 =~ s/ //g ;
                    $r2 = sprintf("%2.2f",$r2) ;
                    if ($opt_a) {
                        $R2 .= " & \\multicolumn{2}{c}{$r2 }" ;
                    }
                    else {
                        $R2 .= " & $r2 " ;
                    }
                }
                if ($line =~ /Pseudo R2/ && $r2flag==0){
                    $r2flag = 1 ;
                    ($junk,$r2) = split(/R2/,$line) ;
                    ($junk,$r2) = split(/=/,$r2) ;
                    $r2 =~ s/ //g ;
                    $r2 = sprintf("%2.2f",$r2) ;
                    if ($opt_a) {
                        $R2 .= " & \\multicolumn{2}{c}{$r2 }" ;
                    }
                    else {
                        $R2 .= " & $r2 " ;
                    }
                }

                if ($line =~ /Number of obs/) {
                    $nflag = 1 ;
                    ($junk,$n) = split(/obs =/,$line) ;
                    $n =~ s/ //g ;
                    if ($opt_a) {
                        $N .= " & \\multicolumn{2}{c}{$n }" ;
                    }
                    else {
                        $N .= " & $n " ;
                    }
```

```
        }
    }
    if ($r2flag == 0) {
        if ($opt_a) {
            $R2 .= " & \\multicolumn{2}{c}{---} " ;
        }
        else {
            $R2 .= " & --- " ;
        }
    }
    if ($nflag == 0) {
        if ($opt_a) {
            $N .= " & \\multicolumn{2}{c}{---} " ;
        }
        else {
            $N .= " & --- " ;
        }
    }
if (($line = <LOGFILE>) =~ /Robust/) {
    $line = <LOGFILE> ;
}
chop($line) ;
$line =~ s/^\s+// ;
$line =~ s/\|// ;
($dpndt,$junk) = split(/\s+/,$line) ;
$dpndt =~ s/ //g ;
if ($opt_a) {
    $Dpndt .= " & \\multicolumn{2}{c}{$dpndt$note} " ;
}
else {
    $Dpndt .= " & $dpndt$note " ;
}
$line = <LOGFILE> ;
# Read in variables, beta's, se's until the end of the regression
until (($line = <LOGFILE>)=~ /-{60,}/) {
    chop($line) ;
    $line =~ s/^\s+// ;
    $line =~ s/\|//g ;
    ($ind,$beta,$se,$tstat,$pval,@junk) = split(/\s+/,$line) ;
    if ($opt_p) {
        $se = $pval ;
        $error_type = "P-values" ;
        }
    else {
        $error_type = "Standard errors" ;
    }

    # Significant figures
    $num = sprintf("%12.3e",$beta) ;
    $numa = sprintf("%12.3e",$se) ;
    if (-1 < $num && $num < 1) {
        ($dec) = ($num =~ /e-(\d+)/);
        $dec = $dec + 2;
        $beta = sprintf("%7.${dec}f","$num");
        if ($opt_p) {
            $se = sprintf("%1.3f","$numa");
        }
        else {
            $se = sprintf("%7.${dec}f","$numa");
        }
        if ($opt_a) {
            $beta =~ s/\./\&./g ;
            $se   =~ s/\./\&./g ;
        }
        $se = "($se)" ;
    }
    elsif ($num <= -100 || $num >= 100) {
```

```
        $beta = sprintf("%7.0f","$num");
        if ($opt_p) {
            $se = sprintf("%1.3","$numa");
        }
        else {
            $se = sprintf("%7.0f","$numa");
        }
        if ($opt_a) {
            $beta = "$beta &" ;
            $se = "($se & )" ;
        }
    }
    else {
        ($dec) = ($num =~ /e\+(\d+)/) ;
        $dec = 2 - $dec ;
        $beta = sprintf("%7.${dec}f", "$num");
        if ($opt_p) {
            $se = sprintf("%1.3","$numa");
        }
        else {
            $se = sprintf("%7.${dec}f", "$numa");
        }
        if ($opt_a) {
            $beta =~ s/\./\&./g ;
            $se   =~ s/\./\&./g ;
        }
        $se = "($se)" ;
    }
    # Create the list of independent variables
    $ind =~ s/_// ;
    $ind =~ s/\*// ;
    $listind .= $ind . ":" ;
    if ($reg==1) {
        push(@vars,$ind) ;
    }
    # if new $ind, i.e., $beta{$ind} empty, then create and fill blanks
    if ($beta{$ind} eq "") {
        if ($reg>1) {
            push(@vars,$ind) ;
        }
        for($i=1;$i<$reg;$i++) {
            if ($opt_a) {
                $beta{$ind} .= " & \\multicolumn{2}{c}{--- } " ;
                $se{$ind} .= " & \\multicolumn{2}{c}{ }    " ;
            }
            else {
                $beta{$ind} .= " & --- " ;
                $se{$ind} .= " &     " ;
            }
        }
    }
    $beta{$ind} .= "& $beta " ;
    $se{$ind} .= "& $se " ;
}
# put in filler for all existing $inds not included in this regression
foreach $j (keys(%beta)) {
    if ($listind !~ m/$j:/) {
            if ($opt_a) {
                $beta{$j} .= " & \\multicolumn{2}{c}{--- } " ;
                $se{$j} .=  " & \\multicolumn{2}{c}{ }    " ;
            }
            else {
                $beta{$j} .= " & --- " ;
                $se{$j} .= " &     " ;
            }
        }
}
```

```
    }
}


# Output table
if (($tableflag == 1) || ($regind !~ /TABLE/) ) {

    # Longtable
    if ($opt_l) {
        print ("\\begin{longtable}[c]{l");
    }
    # Regular table
    else {
        print ("
\\begin{table}[htbp]
\\caption{$caption}
\\begin{center}
\\begin{minipage}[t]{\\textwidth}
\\begin{center}
\\begin{tabular}{l");
    }


        for($i=0;$i<$reg;$i++){
            if ($opt_a) {
                print ("r\@\{\}l") ;
            }
            else {
                print ("c");
            }
        }
    print ("}\n") ;
    if ($opt_l) {
        print ("\\caption{$caption}") ;
    }
    print ("\\\\ \\hline\\hline \\\\[2mm]") ;
        # Longtable header material
        if ($opt_l) {
            print ("      ",$Dpndt, " \\\\ \\\\ \n\\hline \\\\ \\endfirsthead \n") ;
            print (" \\hline \\\\", $Dpndt, " \\\\ \\\\ \n\\hline \\\\ \\endhead \n") ;
            print ("\\\\[2mm] \\hline continued \\endfoot\n") ;
            print ("\\hline\\hline \\endlastfoot\n") ;
        }
        # Regular table header material
        else {
            print ("      ",$Dpndt, " \\\\ \\\\ \n\\hline \\\\ \n") ;
        }
        # Body of table
        foreach $j (@vars) {
            $se{$j} =~ s/\( +/(/g ;
            print ("$j ", $beta{$j}, " \\\\ \n        ", $se{$j}, "\\\\\%[2mm] \n") ;
        }
        print ($R2, " \\\\[2mm] \n") ;
        print ($N, " \\\\[2mm] \n") ;
        # Longtable end
        if ($opt_l) {
            print ("
\\end{longtable}\n
$error_type in parentheses.") ;
        }
        # Regular table end
            else {
        print ("
\\\\ \\hline\\hline
\\end{tabular}
\\end{center}
\\end{minipage}
```

```
\\label{tab:reg$table}
\\end{center}
\\end{table}") ;
            }
            print ("\\clearpage\n\n") ;
    }



    close(LOGFILE) ;
    $table++ ;

}

print("\\end{document}") ;
```

C joe.pl

```
#!/usr/local/bin/perl
# joe.pl parses Job Openings for Economists
$num_apps = 0;


while($line=<>) {

    if (($line =~ /^[A-Z][A-Z][A-Z]/ | $line =~ /U\.S\./) & $line !~ /^CONTACT/) {
        $school = "\%$line" ;
        &get_contact ;
        if ($line =~ /^\.\./) {
            &get_contact ;
        }
        if ($line =~ /^\.\./) {
            &get_contact ;
        }
    }
}


sub get_contact {
    print $school ;
    print "\\textsc{\n";
    until (($line=<>) =~ /CONTACT/) {
    }
    $num_apps++ ;
    ($junk,$address) = split(/CONTACT:/,$line) ;
    until ((($line=<>) eq "\n") || ($line =~ /^\.\./) ){
        $address .= $line ;
    }
    &process_address ;
    print "}\n\n" ;
}


sub process_address {
            $address =~ s/\n/ /g ;
            $address =~ s/^\s+// ;
            $address =~ s/\s+$// ;
            $address =~ s/#/\\#/g ;
            $address =~ s/&/\\&/g ;
            $address =~ s/\.$// ;
            $address =~ s/\(.*\)$// ;
            $address =~ s/comments.+$//i ;
            $address =~ s/,\s*([A-Z][A-Z])[\s\n]/ $1 /g ;
            $address =~ s/(.+) (Chair),./$1 $2\n/i ;
            $address =~ s/, +/\n/g ;
```

```
            $num_lines = ($address =~ tr/\n/\n/) ;
            print ($address,"\n") ;
}
print "\n$num_apps\n\n" ;
```

References

Beebe, N. H. F.: December 1993, Bibliography prettyprinting and syntax checking, *TUGboat* **14**(4), 395–419.

Diller, A.: 1993, *LaTeXLine by Line*, John Wiley and Sons.

Goossens, M., Mittlebach, F. and Samarin, A.: 1994, *The LaTeXCompanion*, Addison–Wesley Publishing Company.

Knuth, D. E.: 1990, *The TeXbook*, Addison–Wesley Publishing Company.

Kopka, H. and Daly, P. W.: 1995, *A Guide to LaTeX 2ε*, Addison–Wesley Publishing Company.

Lamport, L.: 1986, *LaTeX: A Document Preparation System*, Addison–Wesley Publishing Company.

Oram, A. and Talbott, S.: 1991, *Managing Projects with make, 2nd Edition*, O'Reilly.

Reckdahl, K.: 1997, *Using Imported Graphics in LaTeX 2ε*. Available from ftp://ftp.tex.ac.uk/tex-archive/info/epslatex.ps.

Stallman, R. M. and McGrath, R.: 1988, *GNU Make*, Free Software Foundation. Available from http://www.gnu.ai.mit.edu/software/make/make.html.