LaTeX

Statistics 135

Autumn 2005

What is TEX?

TEX represents the state-of-the-art in computer typesetting. It is particularly valuable where the document, article, or book to be produced contains a lot of mathematics, and where the user is concerned about typographic quality. TEX software offers both writers and publishers the opportunity to produce technical text, in an attractive form, with the speed and efficiency of a computer system.

(from the back cover of The TEXbook by Donald E. Knuth, the initial developer of TEX)

Most of what is done today is not in plain TEX, but with addon macro packages. Parts of TEX tend to be cryptic, so these addons have been created to make things easier. The most popular of these addons is LATEX. Others you might come across are AMS-TEXand AMS-LATEX.

What is LaTeX?

LaTeX is a document preparation system based on the TeX formatter. It is a set of macros which simplifies much of plain TeX. LaTeX is the most popular approach to writing papers in Science, particularly in Mathematics and Physical Sciences. Much publishing today is done in LaTeXİ believe everything published by Springer-Verlag, is prepared in LaTeX. Many journals have also gone the LaTeX route as well, with packages prepared to assist with matching the journal format.

The most current version commonly available is LaTeX 2_ε, though LaTeX3 is currently in development. The previous version was LaTeX 2.09, which is quite a bit different. While files written in this version will run in LaTeX 2_ε (usually), you want to stick with LaTeX 2_ε for your writing.

LaTeX, and all TeX derivatives are page markup languages, with formatting commands mixed in with the text, similar to HTML. It is not a WYSIWYG approach, like Microsoft Word, though there are products that try to bring this to the LaTeX world.

A LaTeX file, is a plain text file which is then processed to give the desired output. The output is usually in a `.dvi` file, which then can be converted to other formats, such as postscript or pdf, if desired.

This approach to preparing documents allows for great flexibility and many things to be automated. These include

- Pagination
- Equation numbering
- Creation of tables of contents, list of figures, etc
- Bibliography creation and proper citation in the text
- Changing fonts and font sizes

How to Learn LaTeX

While checking manuals, online searches, etc can be helpful, the way most people learn LaTeX is to see other LaTeX files. In addition to these overheads, I will make available the LaTeX file for this document, plus others, so you can see how things work.

Even if you think you know how to do something, many things can be done many ways, so it can be interesting to see how other people will approach a problem.

For example `\textit{italic}` and `{\it italic}` will both italicize the word *italic*.

Example LaTeX file

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

% define the title
\author{H. ~Partl}
\title{Minimalism}

\begin{document}

% generate the title
\maketitle

% insert the table of contents
\tableofcontents

\section{Some interesting words}
Well, and here begins my lovely article.
```

```
\section{Good Bye World}
\ldots{} and here it ends.

\end{document}
```

Typical LaTeX session

1. Edit/Create your LaTeX input file. This file must be plain ASCII text. In addition, the file name must in in `.tex`

2. Run LaTeX on your input file. It may be necessary to run LaTeX more than once to get the table of contents, table numbers, citations, etc correct. If there is a bug in your input file, you will get an error message and processing will stop. If there are no errors you will get a `.dvi` file.

   In Unix, you can process your document with a command such as `latex foo.tex`.

3. Viewing the `.dvi` file. There are a number of ways to do this. In Unix (under X windows), the file can be display on your screen with `xdvi foo.dvi &`. In Windows, you can use the program `yap`. You can convert the file to postscript with `dvips foo.dvi -o foo.ps` or print it with `dvips foo.dvi` in Unix. It can be converted to pdf with `dvipdf foo.dvi`.

These three steps will be required regardless of your computer setup, however how they are implemented can very greatly. For example, many people in the Unix world will often run LaTeXfrom within an `emacs` session, where the approach is a bit more unified. However others tend to work from the command line. Also, the `emacs` setup can also be used in a Windows or Macintosh setup, though I think these are less common.

Document Layout

1. Document Classes.

   LaTeX needs to what type document is to be created. This is specified with the `documentclass` command.

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

   class specifies the type of document to be created. Common choices are `article`, `report`, `book`, and `slides`. These slides were created with `foils`, a different set of macros for preparing slides. Note foils is not in the standard LaTeX installations, but can downloaded via CTAN (The Comprehensive TeX Archive Network). I'll look into having it installed.

   The *options* parameter customize the behaviour of the document class. The options need to be separated by commas. Popular options are `10pt`, `12pt`, `onecolumn`, and `landscape`. The actual options available will vary by document class.

2. Packages

When preparing your document, you may find that there are things you would like to do which aren't available in the default LaTeX setup. If you want to include graphics, add color, or include source code from a file into your document, you will need to enhance your setup. These enhancements are known as packages and can be activated with the

$\boxed{\texttt{\textbackslash usepackage[}\mathit{options}\texttt{]}\{\mathit{class}\}}$

For example, to typeset the document using Times-Roman instead of the standard Computer Modern font, add the command

```
\usepackage{times}
```

Note how a package acts may depend on the document style being used. For example, the above would actually make this document be typeset in Helvetica, not Times (it is a quirk of the way `foils` is designed). However for `article`, `report`, etc, you would get times.

3. Page Styles

LaTeX supports three different page styles (header/footer) combinations. The *style* parameter of the command

```
\pagestyle{style}
```

defines which to use. The choices are `plain` (page number at the bottom of the page in the middle of the footer), `heading` (prints current chapter heading and the page number in the header, while the footer remains empty), and `empty` (both header and footer are empty).

To change it only for the current page, use the command

```
\thispagestyle{style}
```

To do more complicated things, see either The LaTeX Companion or The Not so Short Introduction to LaTeX2e (available on the course web site).

Document Structure - Sectioning

For most document styles, documents are divided using the different sectioning commands. The standard ones are

```
\part       \subsection      \paragraph
\chapter    \subsubsection   \subparagraph
\section
```

The `article` class doesn't contain the `\chapter` command. However this makes it easy to include an “article” as a chapter of a ”report” or “book”.

The form of these commands is similar to `\section{Example Section}`.

With this approach, it is easy to have your chapter, sections, subsections, etc, to be automatically numbered. For articles, number is of the form Section 2, Subsection 3.4, Subsubsection 3.1.5. For books and reports, the form is Chapter 3, Section 4.2, etc.

If you want to have an unnumbered subsection, for example, you need to add a `*` to the command, such as `\subsection*{Unnumbered subsection}`.

Adding a `*` to the end of a command that deals with numbering, will often remove that numbering from the object.

You can keep track off section numbers you use elsewhere in the text with `\label{`*keystring*`}` command. To refer to a stored label, use the `\ref{`*keystring*`}` and its page number with `\pageref{`*keystring*`}` command. These commands are particularly useful with equations.

Preparing and Input File

As mentioned before, your input file should be a plain ASCII text files with a filename ending with `.tex`. Only the standard character set, numbers, and punctuation characters

`. : ; , ? ! ‘ ’ ( ) [ ] - / * @`

should be used.

There are 10 special characters

`# $ % & ~ _ ^ \ { }`

that are only used in LaTeX commands. For example % initiates a comment, \$ is used to indicate mathematics, and \ indicates a command.

The characters `+ = | < >` are mainly used in mathematical formulas, though $+$ and $=$ can be used in ordinary text. Most of the special characters can be included in text by preceding it with a \. For example \\$ will give a \$.

When entering text, the ends of words and sentences are marked by spaces. It doesn't matter how many spaces you leave when you type; 100 spaces acts the same as 1.

To indicate a new paragraph, leave one or more blank lines before the next paragraph.

Fonts

Style	Approach 1	Approach 2
italics	`{\it italics}`	`\textit{italics}`
slanted	`{\sl slanted}`	`\textsl{slanted}`
emphasized	`{\em emphasized}`	`\textem{emphasized}`
boldface	`{\bf boldface}`	`\textbf{boldface}`
`typewriter`	`{\tt typewriter}`	`\texttt{typewriter}`
roman	`{\rm roman}`	`\textrm{roman}`
SMALL CAPS	`{\sc small caps}`	`\textsc{small caps}`

Table 1: Font styles

Size	Approach
tiny	`{\tiny tiny}`
scriptsize	`{\scriptsize scriptsize}`
footnotesize	`{\footnotesize footnotesize}`
small	`{\small small}`
normalsize	`{\normalsize normalsize}`
large	`{\large large}`
Large	`{\Large Large}`
huge	`{\huge huge}`
Huge	`{\Huge Huge}`

Table 2: Font sizes

Braces, Environments, and Delimeters

In LaTeX, braces { and } are used to indicate pieces of text that are to be treated in a certain way. For example on the previous page, construction like {`\Large match your braces`} were used to indicate a piece of text is to be displayed in the `Large` font size. Missing the closing }, could lead to the rest of the document being created in the `Large` font, or more likely an error message when being processed. You have to make sure your braces match.

This also occurs with other LaTeX constructs. For example, in-line math formulas are usually indicated with `$ ... $`. If the closing `$` is missing you will usually get an error message. The other common situation where you need to be careful is with LaTeX environments, which are indicated with `\begin{} ... \end{}`. Enviroments are used in many situations, such as equations, lists, tables, figures, quotes, and so on.

Also when dealing with these sorts of constructs, you need to deal with them in a LIFO (Last In, First Out) type approach. For example `{ $ \alpha } $` probably will return an error message, where `{ $ \alpha $ }` will work fine, displaying the greek letter α.

These sorts of problems are probably the second most common errors in LaTeX, following omitting a backslash for a command. For example, typing `$bar{x}$` instead of `$\bar{x}$`, which will lead to $barx$, instead of displaying $\bar{x}$.

Mathematical Formulas

All mathematical typesetting is done through one of the math modes, `math`, `displaymath`, and `equation`. These are used for *in-text* formulas, unnumbered display equations, and numbered display equations.

Math mode is usually indicated by `$ ... $`. For example the code

```
The equation $\alpha + \beta = \gamma$
is just for example.
```

gives the output “The equation $\alpha + \beta = \gamma$ is just for example”.

The other 2 environments deal with display equations. Examples of each are are

```
\begin{equation}
g(s, t; X_{t}) = \int \delta(r_{s,w})
f(w|s, X_t)dw;~s \in D
\end{equation}

\begin{displaymath}
g(s, t; X_{t}) = \int \delta(r_{s,w})
f(w|s, X_t)dw;~s \in D
\end{displaymath}
```

These give the following output

$$g(s, t; X_{t}) = \int \delta(r_{s,w}) f(w|s, X_t)dw;~s \in D \tag{1}$$

$$g(s, t; X_{t}) = \int \delta(r_{s,w}) f(w|s, X_t)dw;~s \in D$$

Note that the `displaymath` environment can also be indicated with `\begin{equation*} ... \end{equation*}` or with the even easier `$$ ... $$`

The `displaymath` and `equation` environments are designed for one-line formulas. For multiline formulas, one approach is the `eqnarray` environments. For example

```
\begin{eqnarray*}
y & = & \int_a^b (f(x) + g(x))dx \\
  & = & \int_a^b f(x)dx + C
\end{eqnarray*}
```

gives

$$\begin{aligned} y &= \int_a^b (f(x) + g(x))dx \\ &= \int_a^b f(x)dx + C \end{aligned}$$

Adding the package `amsmath` (`\usepackage{amsmath}`), part of $\mathcal{AMS}$−LaTeX, allows for other approaches.

For example, the `align` environment has a similar effect, but better spacing in my opinion.

```
  \begin{align*}
  y & = \int_a^b (f(x) + g(x))dx \\
    & = \int_a^b f(x)dx + C
  \end{align*}
```

gives

$$\begin{aligned} y &= \int_a^b (f(x) + g(x))dx \\ &= \int_a^b f(x)dx + C \end{aligned}$$

Other environments for mulitline formulas are `multiline`, `gather`, and `split`.

The package `amsmath` does much more than this. If you are dealing with mathematics, use almost always want to add this package, along with `amsfonts` and `amssymb`. For example, additional mathematical features and structures are added. One example is `\binom{a}{b}`, which gives

$$\binom{a}{b}$$

Adding unused packages to a LaTeX file usually has minimal effect. So many people will create lists of items that they will include in all files.

Math Symbols

There is a wide arrange of symbols available in LaTeX. I suggest checking any of the suggested reference on the web site. One useful is at `<http://www.agu.org/symbols.html>`.

Many of them are obvious, where a `\` proceeds the name, e.g. `\theta` or `\div`. However some are a bit cryptic, e.g. `\pm` for $\pm$.

A common convention in mathematical typesetting, is for functions like `log`, `sin`, `min` to in roman text, not italized. For these sorts of functions, adding a `\` to the command stops the function name from begin italized. For example `$\exp(x)$` gives $\exp(x)$, whereas, `$exp(x)$` gives $exp(x)$.

Common Structures

Subscripts and Superscripts

Subscripts and superscripts are made with the _ and ^ commands. These can be combined to make complicated structures

x^{2y}	`x^{2y}`	x^{y^2}	`x^{y^2}`
x_{2y}	`x_{2y}`	x_2^y	`x_2^y`

Fractions

Fractions can either be denoted with / with the `\frac` command. For example `$a/b$` gives a/b where as `\frac{a}{b}` gives

$$\frac{a}{b}.$$

The `\frac` command can be used inline, but it tends not to be.

Roots

Roots are indicated with the `\sqrt` command. A square root is indicated with `\sqrt{x \times y}` ($\sqrt{x \times y}$) and an n^{th} root with `\sqrt[n]{3}` ($\sqrt[n]{3}$).

Summation, Integrals, Products, ...

The commands for these functions have the property that they they adjust there size, depending on the formula. The basic structure is of the form

```
\begin{equation}
\frac{\pi^2}{8} = \sum_{k=1}^{\infty}
                      \frac{1}{(2k-1)^2}
\end{equation}
```

which gives

$$\frac{\pi^2}{8} = \sum_{k=1}^{\infty} \frac{1}{(2k-1)^2} \tag{2}$$

For integration use `int`, products use `prod`, unions ($\bigcup$) use `bigcup`, intersections ($\bigcap$), use `bigcap`, etc.

Defining commands

A useful feature in LaTeX is the ability to define commands. These are often used to create custom shortcuts for your commonly used commands. To do this use the `\newcommand` command. Its first argument is the command name and the second argument isthe command text. For example to create a command for $\bar{x}$, you could use

```
\newcommand{\xbar}{\ensuremath{\bar{x}}}
```

It is possible to create more complicated commands, such as ones that take arguments. An example, albeit silly, is

```
\newcommand{\silly}[2]
   {\ensuremath{f(#1 + 2, #2 - 2)}}
```

for which `\silly{2x}{z}` gives $f(2x + 2, z - 2)$.

You need to be a bit careful in naming your new commands. You don't want them to start with the names of built in commands.

Adding Graphics

While there are many ways of adding graphics to a LaTeX document, probably the most popular way currently is with the `graphicx` package, which I believe is now part of most standard LaTeX installations. To use it, you just need to give the command `\usepackage{graphicx}`.

This is particularly useful with postscript graphics created by either **S** or **MATLAB**. I believe that the package will handle other graphics format such as tiff, but I've never used anything but encapsulated postscript.

The base command is

$$\boxed{\texttt{\textbackslash includegraphics[}\mathit{key{=}value,\ \ldots}\texttt{]}\{\mathit{file}\}}$$

The important *keys* are `width`, `height`, `angle`, and `scale`.

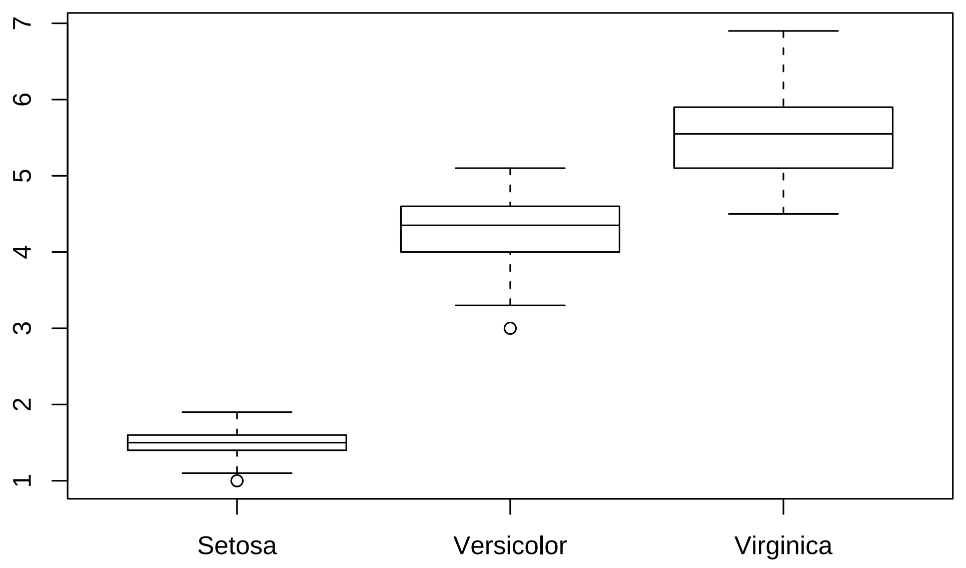

Figure 1: Boxplot of Petal Width generated by R

which was formatted by the following code

```
\begin{figure}[ht]
\begin{center}
\includegraphics[scale=0.8]{rbox.ps}
\caption{\label{fig:rbox}
Boxplot of Petal Width generated by R}
\end{center}
\end{figure}
```

Usually you want to create the graphic the correct size and set the scale at 1 `[scale=1]`, but there can be times that rescaling the size of the graphic can be useful.

List Structures

There are a number of different list environments available in LaTeX. These include

- Unnumbered lists: `itemize`

  Items in this type of list are headed by bullets, dashes, etc

- Numbered lists: `enumerate`

  Items in this type of list are headed by numbers, letters, roman numerals, etc

- Lists with user specified labels: `description`

  In this type of list, the user can specified the header of each item, including no header, though this usually looks weird.

This list was created by the following LaTeX code:

```
\begin{itemize}

\item Unnumbered lists: \code{itemize}

\item Numbered lists: \code{enumerate}

\item Lists with user specified labels: \code{description}

\end{itemize}
```

To create a list, the appropriate `\begin` and `\end` commands must bracket the list. A new list item is indicated with the `\item` command. For a `description` list, the item header are created by `\item[headername]`.

For example

```
\begin{description}

\item[gnat] A small animal, $\ldots$

\item[gnu] A large animal, $\ldots$

\item[armadillo] A medium sized animal, $\ldots$
\end{description}
```

will create

gnat A small animal, $\ldots$

gnu A large animal, $\ldots$

armadillo A medium sized animal, $\ldots$