

# Getting Funky: Headers, Tables, Abstracts etc.

## Introduction

Today, we will be finishing off our need-to-know course with some extra bits and pieces that didn't really fit anywhere else in the tutorials. We won't be doing bibliographies, because they can be very fiddly, very irritating and teaching people to use the required program called BibTeX can be very frustrating (for all parties).

## Funky Headers

Some people like to have special headers on their L<sup>A</sup>T<sub>E</sub>X files. I have one at the top of this page. These can be helpful on hand-outs or homework problems. Fortunately, they are not difficult to do.

Hopefully you will recall that when we open WinEdt to write an input file, and we are using the math packages, it looks something like this:

```
\documentclass{article}
\usepackage{amsmath, amssymb}

\begin{document}

\end{document}
```

To get a nice header, you need to add a couple of bits of code.

- The first is to tell L<sup>A</sup>T<sub>E</sub>X to use another package. That package is called `fancyhdr` and you just type it anywhere in the box after `\usepackage{ }`. Make sure it is separated from the other package names by a comma.
- then, you need to write a command telling L<sup>A</sup>T<sub>E</sub>X the type of page style to operate. This command is `\pagestyle{fancy}` and goes straight under the `\usepackage{ }` command.
- now, you need to tell L<sup>A</sup>T<sub>E</sub>X what to actually write as your left and right header. You do this with the commands `\lhead{ }` where the words to appear on the top left go in the braces and `\rhead{ }` where the words to appear on the top right go in the braces.
- as an example, let's say I wanted my name on the right, and the course title "Math Modelling" on the left. My input file would need to look like this:

```

\documentclass{article}
\usepackage{fancyhdr, amsmath, amssymb}
\pagestyle{fancy}

\lhead{Math Modelling} \rhead{Arthur Spirling}
\begin{document}

\end{document}

```

## Making Tables

The `tabular` environment is, in some senses, just like L<sup>A</sup>T<sub>E</sub>X 's other environments, though it takes different commands and produces very different outputs. We use this environment to produce the following sort of table:

Name	Country of Origin	Favorite Greek letter
David Carter	USA	$\pi$
Arthur Spirling	UK	$\theta$
Arnd Pagge	Germany	$\rho$
Subhashish Ray	India	$\alpha$
Mike Epstein	USA	$\gamma$

This table was produced using the following commands:

```

\begin{center}
\begin{tabular}{|l|c|c|}\hline
Name& Country of Origin & Favorite Greek letter\\\hline %
David Carter & USA &  $\pi$ \\\hline %
Arthur Spirling & UK&  $\theta$ \\\hline%
Arnd Pagge & Germany &  $\rho$ \\\hline%
Subhashish Ray & India &  $\alpha$ \\\hline%
Mike Epstein & USA &  $\gamma$ \\\hline%
\end{tabular}
\end{center}

```

There are actually a few things to notice here:

- the `center` environment is used to center the table on the page. If we didn't use it, L<sup>A</sup>T<sub>E</sub>X would put it flush left
- the general form of the `tabular` environment is:

```
\begin{tabular}{column info} table \end{tabular}
```

We can go through these parts one-by-one.

- the `{column info}` specifies how to format the columns. Here, this was `|1|c|c|`. This means the first column will have its entries left aligned, and the other two will have centrally aligned entries. You can use `r` if you want things right aligned in that column.
- a line `|` in those braces causes a vertical line to divide the columns from top to bottom: leaving them out means no dividing line.
- the `\hline` commands give horizontal lines after that entry line of data. The `\hline` must come after the `\end-of-line` command. We have some `%` marks here to make the text-wrapping more convenient.
- each row will have ampersands—the `&` symbol—to demarcate the column entries for each column. There will always be one less per row than the total number of columns.
- notice how we had to switch into math mode to write the Greek letters. This is vital, otherwise L<sup>A</sup>T<sub>E</sub>X will just return errors: if you want to write some math in there, make sure you are in math mode.
- when the table is finished, and the `\end{center}` command is executed, there is no need to try and leave space before you start the next line. L<sup>A</sup>T<sub>E</sub>X will take care of that for you. If you try and put `\` after a table, it will tell you that

`! There's no line here to end`

## Table Numbering and Comments

The `tabular` environment above can be made more useful by adding `\begin{table}` and `\end{table}` before the `\begin{center}` and `\end{center}` commands respectively. Doing so means that you can add notes and table numbers.

- the `\caption{}` command tells L<sup>A</sup>T<sub>E</sub>X to number the table, while the braces allow you to write a title. An example is the following:

```
\begin{table}[h]
\begin{center}
\begin{tabular}{c|c|c|}
&Cooperate&Defect\\\hline%
Cooperate&3,3&1,4\\
Defect&4,1&2,2\\\hline
\end{tabular}
\end{center}
\caption{The Prisoner's Dilemma}
\end{table}
```

which produces something like:

	Cooperate	Defect
Cooperate	3,3	1,4
Defect	4,1	2,2

Table 1: The Prisoner's Dilemma

- a package we won't discuss here at length, but that is discussed in many L<sup>A</sup>T<sub>E</sub>X guides is `endfloat`. This package places all 'floats' (which are figures and tables) at the end of your document, each on a page by itself. This maybe useful if you have when submitting articles to certain journals that require such a format. To use this package, all you need do is add its name to the `\usepackage{ }` command in your preamble. So, for our documents, that it would now read

```
\usepackage{endfloat,amsmath, amssymb }
```

Now, when you write a table in your input file (say, Table 3), L<sup>A</sup>T<sub>E</sub>X will automatically write:

[Table 3 about here.]

You can carry on writing your text below.

## Thanks, abstracts and page-breaks

### Thanks

Recall that in the first tutorial when dealing with setting up an input file, we suggested the following format:

```
\documentclass[11pt]{article}
\title{My Day at the Department}
\author{Arthur Spirling}
\begin{document}
\maketitle
\noindent
```

We can use the `\thanks` command to note contributions and caveats. You can use more than one and an example is as follows:

```
\documentclass{article}
\title{Our Day at the Department\thanks{Thanks to participants of
the Watson seminar. The usual caveat applies.}}
\author{Richard Fenno \and Arthur Spirling}
\thanks{Spirling thanks David Carter and Justin Fox for enlightening
```

```
discussion.}}
\begin{document}
\maketitle
\noindent
```

which will produce a \* symbol for the first footnote by `Department`, a † for the second (by `Spirling`). There are several more symbols if needed.

- note that you write the thanks text in braces, within either the title or author braces.
- note the use of `\and` which will space the names properly.

## Abstracts

If you want an abstract for your article, you need to follow these steps:

- you write the abstract in the `abstract` environment which is just like the others we have discussed
- it is best if it is put after the `\maketitle` command in the input file and should be written something like the following:

```
\maketitle
\begin{abstract}
This paper explores the variables governing the attractiveness of
junior male faculty to self-delusionary female graduate students.
We find that, contrary to the rest of the known world, facial
handsomeness, charm, wit and dating availability of assistant
professors have zero effect on these mens' attractiveness to
ambitious women. We also find that graduate males perceive this
situation as baffling and irritating.
\end{abstract}
```

This produces an effect something like the following (though below your title):

### Abstract

This paper explores the variables governing the attractiveness of junior male faculty to self-delusionary female graduate students. We find that, contrary to the rest of the known world, facial handsomeness, charm, wit and dating availability of assistant professors have zero effect on these mens' attractiveness to ambitious women. We also find that graduate males perceive this situation as baffling and irritating.

- note that the abstract environment automatically indents the first line of the abstract. This is suppressed by the `\noindent` command as usual.

## Page-breaks

Page-breaks, such that the next line of the next paragraph appears on a blank page are easily achieved. Simply use the command `\pagebreak` where required.

## R output in L<sup>A</sup>T<sub>E</sub>X

The statistical program R is increasingly popular in our department. Unlike STATA though, there is no easy to use translation package that will take your statistical output and write it neatly in L<sup>A</sup>T<sub>E</sub>X. A way round this is the `verbatim` environment. This environment tells L<sup>A</sup>T<sub>E</sub>X to write what you have entered (including all punctuation and spacing) in exactly the way you entered it. It does so in the same font that you write WinEdt in: i.e. a type-writer style, courier. If you have some R output to put in L<sup>A</sup>T<sub>E</sub>X follow these steps:

- click down your left mouse key in R and highlight (in a fetching shade of blue) all the output you want to export
- right click your mouse: and select `copy`
- return to your L<sup>A</sup>T<sub>E</sub>X input file and write `\begin{verbatim}` where you want the output to begin
- right click the mouse in WinEdt and select `paste`
- write `\end{verbatim}`. When you run L<sup>A</sup>T<sub>E</sub>X the output will appear as it was in the R window
- note that L<sup>A</sup>T<sub>E</sub>X will leave space after the `verbatim` environment: there is no need to use the `\\` command

An example of the output you can show is as follows:

```
Call: lm(formula = y ~ x)
```

```
Residuals:
```

```
      1      2      3
8.056e-18 -1.611e-17  8.056e-18
```

```
Coefficients:
```

```
              Estimate Std. Error  t value Pr(>|t|)
(Intercept) 0.000e+00  3.014e-17  0.000e+00      1 x
1.000e+00  1.395e-17  7.167e+16  <2e-16 ***
```

```
---
```

```
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
Residual standard error: 1.973e-17 on 1 degrees of freedom
```

```
Multiple R-Squared:  1,      Adjusted R-squared:  1
```

```
F-statistic: 5.136e+33 on 1 and 1 DF,  p-value: < 2.2e-16
```

## PDFing Documents

PDF documents have *Portable Document Format*. This means that nearly any computer can open them (provided they have some free software installed).

PDF is the standard format for papers that are to be sent via email, or are to be kept on a web-site. Though they look exactly like DVI files of your work, they have the advantage that your audience doesn't need to know how L<sup>A</sup>T<sub>E</sub>X and MiKTeX work to open and read them. Putting your paper into PDF format is very easy with WinEdt, and you need to follow these steps:

- write your document in L<sup>A</sup>T<sub>E</sub>X as usual, then press the DVI button as you normally would.
- when you are happy with how your article looks, look for a button on your tool-bar in WinEdt that says `dvi` in blue with an arrow pointing to `pdf` in red. It is right of the DVI button, past the picture of the ghost and to the right of the strange red triangular loop
- press it. A black screen with white writing will begin to scroll which (hopefully) looks something like the following:

```
myday.dvi -> myday.pdf
[1] [2] [3] [4] [5] [6] [7] [8]
11350 bytes written
Press any key to continue....
```

- Press any key. L<sup>A</sup>T<sub>E</sub>X has now written your DVI file as a PDF. To see it, go to the `start` button, follow the `Program` and `words` pathway and select `Adobe Reader 6.0`.
- Now, open the directory you are working on your WinEdt file in. There should be a `.pdf` file there with the same *first name* as your input file.
- this is your PDF file. This can be attached to emails, or put on the web. Or, you can just print it out from there.

## PDFL<sup>A</sup>T<sub>E</sub>X

PDFL<sup>A</sup>T<sub>E</sub>X is a version of L<sup>A</sup>T<sub>E</sub>X that generates PDF output *directly* from your input code (i.e. you don't have to go via DVI). Some people find this more efficient: it 'cuts out the middle man' of having to preview your output in a viewer. Also, there is an argument that PDFL<sup>A</sup>T<sub>E</sub>X is more useful for presentations since it can accommodate hyperlinks and some other special features.

On the downside, it can make your documents a little (not much) slower to compile and means that WinEdt is operating in a slightly 'heavier' mode and (perhaps) more liable to crash (i.e. you need more virtual memory to keep everything open and working than with YAP).

The principles for using PDFL<sup>A</sup>T<sub>E</sub>X are almost identical to using DVI, so they are given only briefly here:

- write your document in WinEdt as usual. Now, instead of hitting the L<sup>A</sup>T<sub>E</sub>X button, look to its right. Three buttons along is a button that says PDF L<sup>A</sup>T<sub>E</sub>X where those two words are interlinked. You do **not** want to press the button that has PDF T<sub>E</sub>X intertwined (look closely!).
- press the PDF L<sup>A</sup>T<sub>E</sub>X button and a black box with white writing will begin to scroll. Press any key when it tells you to do so.
- Now, rather than pressing the DVI button with the magnifying glass, press the button with the red-loop triangle directly below the PDF L<sup>A</sup>T<sub>E</sub>X button. This will launch Adobe Reader 6.0 which is installed in the `star lab`. Your document will now be visible in the Adobe viewer.
- note: when you have finished viewing your document, you should **close** the viewer (don't just minimize it). Otherwise, L<sup>A</sup>T<sub>E</sub>X will complain when you use PDF L<sup>A</sup>T<sub>E</sub>X again that it needs another file name before it can write your output. This annoyance can be corrected with some code in your input file, but it seems a bit much for an “absolute beginners” course.