Going further with LaTeX

Jamie Findlay
February 12, 2020

1 Bits and bobs

1.1 Housekeeping

- As you have no doubt noticed, running `pdflatex` doesn't just produce the PDF output file, but also some other files (including `.aux` and `.log`). (LaTeX uses these to keep track of various things like counters, etc.)
- This can make it rather confusing if you have more than one project in the same folder, so it may be worth thinking about the structure of your filing system before it all gets too out of hand ...

1.2 The local tex directory

- If you ever want to add your own packages (`.sty` files) or document classes (`.cls` files), and make them available anywhere on your system (they are always available to files in the same folder), then this is where you have to put them.
- Location on Mac OS:
  `/Users/<user name>/Library/texmf/tex/latex/`
- Location on Windows:
  `C:\Users\<user name>\texmf\tex\latex\` OR
  `C:\Users\<user name>\Local TeX files\tex\latex\`

1.3 CTAN

- CTAN is the 'Comprehensive TeX Archive Network', and it is the central repository for all sorts of TeX-related stuff, but most importantly packages.
- If you need to manually download a package (not that common any more but it might still happen), CTAN is a good place to look.
- It's also the place to look for documentation for packages, which will give you information on how to use them.

1.4 Unicode

- If you make frequent use of more 'exotic' characters (e.g. you write in a language that uses a lot of accents or other non-English characters), it might be worth investigating XeTeX (XeTeX) or LuaTeX (LuaTeX), which allow for native Unicode recognition (so you can write in e.g. Arabic script).

1.5 Other fonts

- The default LaTeX font (Computer Modern) is pretty nice, but LaTeX is capable of typesetting in a whole bunch of other fonts too. See `https://tug.org/FontCatalogue/` for a very comprehensive list. Many of these will work 'out of the box', simply by loading a particular package (e.g. `\usepackage{times}` will give you Times New Roman), but some might need you to install font files on your system, etc.

1.6 Running LaTeX from the command line

- When you installed your LaTeX distribution, you also acquired a number of commands that can be run from the command line. E.g. `pdflatex file.tex` will do the same thing as pressing 'Typeset' in TeXShop.

- If you're interested in this way of doing things, check out `latexmk` (`http://mg.readthedocs.io/latexmk.html`), which does a lot of nice things, such as continuously checking for changes while you work, tidying up all the extra files LaTeX produces, etc.

1.7 Converting between other file types

- If you need to e.g. convert `.tex` files into Word documents, or convert GitHub Markdown into PDF using LaTeX, check out `pandoc` (`https://pandoc.org`), which, as the name suggests, gives you a lot of options for converting between different file types.

1.8 The Oxford thesis template

- If you're thinking of writing your thesis in LaTeX, you can simply use the standard `book` or `report` classes, or you can use one of the many custom-made thesis classes out there (one good source for templates is ShareLaTeX: `https://www.sharelatex.com`).
- A particularly nice template is the modified `ociamthesis.cls` template by John McManigle which is available from his website here: `https://www.oxfordechoes.com/oxford-thesis-template/`. This is a modified version of a modified version of an original maths department template made by Keith Gillow (see John's site for more info).
- There is a fair amount of detailed information and instruction in the comments, but there is quite a lot going on under the hood, so it is not for the faint hearted, and it might be worth cutting your teeth on some smaller LaTeX documents first.
- (One potential pitfall straight off the bat is that by default the template uses `biber` and `biblatex` rather than the older but perhaps more widespread BibTeX for reference management – see below.)

2 Defining your own commands

- It can be convenient to define your own commands. These can be as complex as you like, but we'll stick to pretty simple examples here. As ever, check out the Wikibook and elsewhere for more.

  Syntax

  ```
  \newcommand{name}[number of arguments]{definition}
  ```

- The simplest use of this is just to make a shortcut for something you have to write frequently:

  Example

  ```
  \newcommand{\ds}{description theory of names}
  ```

- Now, wherever you put `\ds`, you will get 'description theory of names'.
- **Note:** Macros like this swallow up whitespace after them, so that if you write

  ```
  The \ds is wrong.
  ```

  you will get

  The description theory of namesis wrong.

- To fix this, you need to end the command with either `\ `, or `{}`. Thus, either

  ```
  The \ds\ is wrong.
  ```

  or

  ```
  The \ds{} is wrong.
  ```

  will correctly give

The description theory of names is wrong.

- To include arguments in your commands, you simply use tags of the form `#n` to identify where each argument should go. For example:

  Example

  ```
  \newcommand{\op}[2]{\ensuremath{\langle #1, #2 \rangle}}
  ```

- Now we can more easily write out ordered pairs: `\op{x}{y}` gives $\langle x, y \rangle$.

- **Note:** `\newcommand` will not allow you to overwrite an existing command (so it is quite safe in that respect). If you explicitly want this behaviour, use `\renewcommand`, which has the same syntax.

3 Links

- Basic out-of-the-box LaTeX is pretty powerful, but to do lots of things you will need to call *packages*, which you do in the preamble (i.e. before `\begin{document}`).

  Syntax

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

- One example is handling hyperlinks, which uses the `hyperref` package.

- This gives you two commands: `\url{...}` and `\href{...}{...}`.

- `\url{...}` will turn its argument into a link (and display it in typewriter font by default – you can change this by using the `\urlstyle{...}` command, which takes a font style as its argument, e.g. `\urlstyle{rm}` will make your links appear in Roman font). Note that the target of the link will be identical to what is displayed.

- `\href{...}{...}` takes two arguments: the first is the target of the link, the second is the text to display.

  Example

  ```
  Here's a link to \url{http://www.google.com}.\\
  But here's a link to \href{http://www.google.com}{Google}.\\
  For more, contact \href{mailto:more@info.com}{\nolinkurl{more@info.com}}.
  ```

  Example output

  Here's a link to `http://www.google.com`.
  But here's a link to Google.
  For more, contact `more@info.com`.

- If you want to hide the boxes around links, pass the option `hidelinks` to the initial call of the `href` package (i.e. `\usepackage[hidelinks]{href}`). If you don't want them to change colour when clicked, you need the option `colorlinks = false`.

4 Tables

- Tables are, sadly, comparatively hard work in LaTeX. The basics are easy, but as soon as you want to do something a bit more complicated things can very quickly get out of hand.

  Syntax

  ```
  \begin{tabular}[position]{columns and separators}
  column1 & column2 \\
  column1 & column2
  \end{tabular}
  ```

Example

```
\begin{tabular}{| l | c | r |}
\hline
1   & 2   & 333 \\ \hline
444 & 5   & 6   \\ \hline
7   & 888 & 9   \\
\hline
\end{tabular}
```

Example output

1	2	333
444	5	6
7	888	9

- By default, LaTeX won't wrap text in `tabular` environments, which means that cells with a lot of writing in will spill over, even going off the page.

Example

```
\begin{tabular}{|l|l|}
\hline
Number & Long text\\
\hline
1 & This is some really long text which
    is going to end up going off the page if we're not careful---OH GOD! \\
\hline
\end{tabular}
```

Example output

Number	Long text
1	This is some really long text which is going to end up going off the page if we're not careful—OH

- To fix this, we use the `p` attribute and specify the desired width of the column in question:

Example

```
\begin{tabular}{|l|p{8cm}|}
\hline
Number & Long text\\
\hline
1 & This is some really long text which
    is going to end up going off the page if we're not careful---OH PHEW! \\
\hline
\end{tabular}
```

Example output

Number	Long text
1	This is some really long text which is going to end up going off the page if we're not careful—OH PHEW!

- Note that `tabular` environments aren't just for drawing actual tables. They can be used to align diagrams or other things too.

- How to span multiple columns:

Syntax

```
\multicolumn{number of columns}{alignment}{contents}
```

Example

```
\begin{tabular}{llrr}
\hline
      &     & \multicolumn{2}{c}{Style} \\
              \cline{3-4}
Class & Sex & FS & CS \\
\hline
MMC   & M   & 4  & 31 \\
      & F   & 0  & 0 \\
LMC   & M   & 27 & 17 \\
      & F   & 3  & 67 \\
\hline
\end{tabular}
```

Example output

		Style	
Class	Sex	FS	CS
MMC	M	4	31
	F	0	0
LMC	M	27	17
	F	3	67

- The package `multirow` provides an equivalent command for spanning multiple rows:

  ```
  \multirow{number of rows}{width}{contents}
  ```

  (Use `*` for the value of `width` to just use the natural width of the cell's contents.)

- Worth checking out the `booktabs` package, which makes your tables look a lot more professional:

		Style	
Class	Sex	FS	CS
MMC	M	4	31
	F	0	0
LMC	M	27	17
	F	3	67

		Style	
Class	Sex	FS	CS
MMC	M	4	31
	F	0	0
LMC	M	27	17
	F	3	67

- Note that the `table` environment is different: this marks off a table that will be numbered, have a caption, appear in a table of contents, etc. Tables and figures are 'floats', which means LaTeX will display them where it thinks is best (the default preference is for the top of the page).

Example

```
\begin{table}[h]
\centering

\begin{tabular}{ll}
Foo & Bar \\ \hline
1   & 0.5 \\
2   & 274.2
\end{tabular}

\caption{An example of a table}
\label{table example}
\end{table}
```

Example output

Foo	Bar
1	0.5
2	274.2

Table 1: An example of a table

- Note the `\label{...}` command. This can be used to label all sorts of things, e.g. sections, figures, example sentences. Now that it has a label, we can refer to it by using the `\ref{...}` command, which will return the number associated with that object:

  Example

  ```
  Now we can talk about Table \ref{table example},
  and even mention that it is on p.~\pageref{table example}.
  ```

  Example output

  Now we can talk about Table 1, and even mention that it is on p. 6.

- Note that you need the `hyperref` package for these cross-references to also be clickable.
- For more on tables, see the Wikibook, or one of the many online guides. This tutorial is quite thorough, for example: `http://www.texnology.com/teachingSamp.pdf`. The `booktabs` documentation (`http://bit.ly/2lp9QgT`) is also very good on table layout in general.
- If you need to import existing spreadsheets into LaTeX, I've written a simple Bash find-and-replace script which will convert `.csv` files into LaTeX markup: `https://github.com/findlayjy/scripts/blob/master/csv2latex`. Running `csv2latex file.csv` on the command line outputs the code to both a `.txt` file and to the clipboard. (YMMV on a Windows machine.)

5 Example numbers and glossing

- Various packages available (`gb4e`, `linguex`, `expex`, ...). Try them out and see what you like.
- `linguex` is very straightforward:

  Example

  ```
  Here is some text.

  \ex. Here is an example.

  \ex.
  \a. Here is a sub-example.
  \b. And another.
  \b. *And a other.
  \bg. Et voici un exemple avec une glose. \\
  and here.is a.\textsc{masc} example with a.\textsc{fem} gloss \\
  'And here is an example with a gloss.'

  And here is some text which follows.
  ```

  Example output

  Here is some text.

  (1) Here is an example.

  (2) a. Here is a sub-example.
      b. And another.

c. *And a other.
d. Et voici un exemple avec une glose.
and here.is a.MASC example with a.FEM gloss
'And here is an example with a gloss.'

And here is some text which follows.

- As well as the usual `\label{...}` and `\ref{...}` way of referring to examples, `linguex` gives you the handy commands `\Next` and `\Last` to refer to the next example coming up and the immediately preceding one (as well as `\NNext` and `\LLast` for the next but one and last but one).

6 IPA

- Use the `tipa` package.
- This provides a bunch of macros and, more usefully, a lot of shortcuts that can be used inside a set of commands or environments:

1. `\textipa{...}`
2. `{\tipaencoding ...}`
3. `\begin{IPA}`
   `...`
   `\end{IPA}`

- According to the authors, "about 92% of the symbols can be inputted within three keystrokes"(!)
- A lot of the shortcuts are guessable, but check out the documentation for a full list (a good list is found in the appendix of this document: `http://bit.ly/2l7ZzTP`).

Example

```
\textipa{[""f@"nEtIks Iz ku:l]}
```

Example output

[ˌfəˈnɛtɪks ɪz kuːl]

- Some special macros:
  - `\;` is used for small capitals.

    Example

    ```
    \textipa{\;B \;E \;A \;H \;L \;R}
    ```

    Example output

    ʙ ᴇ ᴀ ʜ ʟ ʀ
  - `\:` is used for retroflex symbols.

    Example

    ```
    \textipa{\:d \:l \:n \:r \:s \:z}
    ```

    Example output

    ɖ ɭ ɳ ɽ ʂ ʐ
  - `\!` is used for implosives and clicks.

    Example

    ```
    \textipa{\!b \!d \!g \!j \!G \!o}
    ```

    Example output

    ɓ ɗ ɠ ʄ ɢ ʘ

- `\*` is used for turned characters, miscellaneous special characters, and to escape the IPA coding for individual characters.

  Example

  ```
  \textipa{\*f \*r \*n \*l \*I \*E}
  ```

  Example output

  ɟ ɹ ɲ ɬ I E

- **Note:** `linguex` doesn't play nicely with `tipa` and has to be loaded after it. (In fact, `linguex` doesn't get on with several packages, so probably best to load it last.)

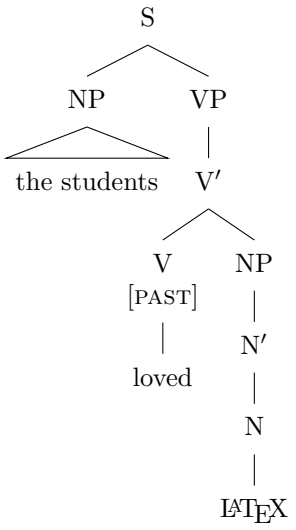
7 Syntactic trees

- Lots of different options here, but one package that strikes a good balance between being easy to use but also very powerful is `forest` (call the package with the `[linguistics]` option).

  Example

  ```
  \usepackage[linguistics]{forest}
  ...
  \begin{forest}
  [S
      [NP [the students, roof]]
      [VP
          [V$'$
              [{V\\$[\textsc{past}]$} [loved]]
              [NP [N$'$ [N [\LaTeX]]]]
          ]
      ]
  ]
  \end{forest}
  ```

  Example output



- The `forest` documentation (`http://bit.ly/2UypwAj`) is very detailed, and you're not likely to need anything approaching the package's full complexity. The 'tutorial' in the second section is good, though. A more concise guide for linguists is available here: `http://bit.ly/2vV7yxL`.

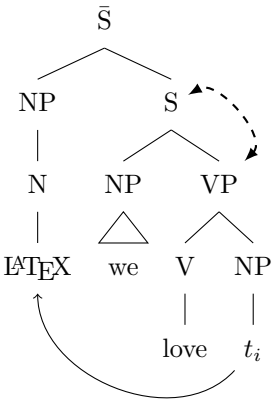
- `forest` is based on Ti*k*Z, which is a *very* powerful drawing package. It is basically its own programming language, just embedded in LaTeX. Here is a good getting started guide for Ti*k*Z: `http://bit.ly/2H2wuFH`, and the full documentation is here: `http://bit.ly/2Smf0cS`.

- For our purposes, though, the important thing is that any drawing command from Ti*k*Z will also work in `forest`:

Example

```
\begin{forest}
[$\bar{\textrm{S}}$
    [NP [N [\LaTeX, name = target]]]
    [S, name = s
        [NP [we, roof]]
        [VP, name = vp
            [V [love]]
            [NP [$t_i$, name = source]]
        ]
    ]
]
%
\draw[->] (source) to[out=south west, in=south] (target);
\draw[<->, thick, dashed,>=latex] (vp) to[out= 45, in = 30] (s);
\end{forest}
```

Example output



8 Citations and bibliography

- More up-to-date people than me will tell you to use `biblatex`; I use the older (but probably still more widespread) BibTeX and `natbib` combo.
- Whatever you use, the underlying logic is the same. You have a database (a `.bib` file) where you store information about your bibliographical items. Each of them has a key associated with it.
- When citing something in your document, you just use an appropriate `\cite` command, with the key as its argument.
- LaTeX will then add an in-line citation of the requested type, while adding the full bibliographical entry to your list of references at the end of the document (or wherever you tell LaTeX to put it).

- Entries in a `.bib` file look like the following:

```
@article{nunberg:idioms,
  author = {Geoffrey Nunberg and Ivan A. Sag and Thomas Wasow},
  title = {Idioms},
  year = {1994},
  journal = {Language},
  volume = {70},
  number = {3},
  pages = {491--538},
  url = {https://doi.org/10.1353/lan.1994.0007},
}

@book{grewendorf:ergativity,
  author = {G\"{u}nther Grewendorf},
  title = {Ergativity in {German}},
  year = {1989},
  publisher = {Foris},
  address = {Dordrecht, NL},
}

@incollection{kamp:drt,
  author = {Hans Kamp},
  title = {A Theory of truth and semantic representation},
  year = {1981},
  booktitle = {Formal methods in the study of language},
  editor = {Jeroen Groenendijk and Theo M. B. Janssen and Martin Stokhof},
  pages = {277--322},
  publisher = {Mathematical Centre Tracts},
  address = {Amsterdam, NL},
}
```

- There are lots of guides online that give details on what different entry types are recognised and what fields each should contain (e.g. this one: `http://bit.ly/2H1Ulp5`).
- If you don't want to have to type all the fields in by hand, or you want a nicer UI, then BibDesk is worth a look. But any decent reference manager should allow you to export your database as a `.bib` file, too.
- You must choose what style your bibliography will be in with the `\bibliographystyle{...}` command. The argument of this command is the name of a `.bst` file, and these can be customised.

  Example

```
\usepackage{natbib}
\bibliographystyle{sp}
```

- `natbib` comes with a few defaults (such as `plainnat`), but personally I like the `.bst` file for the journal *Semantics & Pragmatics*,[1] since it is based on the *de facto* standard in linguistics, the 'unified style sheet for linguistics journals' (`https://www.linguisticsociety.org/resource/unified-style-sheet`).
- As with all TEX-related files, you need to put this either in the folder of the project you're working on, or in your system's local TEX directory. For BibTeX-related content (like `.bst` files), this is found at the same path as we saw in section 1.2, but with `bibtex` in place of the final `latex`.
- At the end of your document, where you want the bibliography to appear, you use `\bibliography` with the name of the `.bib` file (minus the `.bib`) as its argument.

[1] It's called `sp.bst`, and I've uploaded a copy to WebLearn.

Example

```
\usepackage{natbib}
\bibliographystyle{sp}
...
\bibliography{test}
```

- When it comes to actually citing, `natbib` gives you two main commands: `\citet{...}` and `\citep{...}`, the former for citing in running text, the latter for parenthetical citations.

  Example

  ```
  Idioms are hard \citep{nunberg:idioms}.
  However, we must agree with \citet{kamp:drt} that DRT is cool.
  ```

  Example output

  Idioms are hard (Nunberg et al. 1994). However, we must agree with Kamp (1981) that DRT is cool.

- Specific page references are passed as options to the cite commands, e.g. `\citep[279]{kamp:drt}` produces 'Kamp (1981: 219)'.

- There's a lot more power and customisability in `natbib`, so have a play around. There are plenty of good guides online, including this one: `http://merkel.texture.rocks/Latex/natbib.php`.

- So, a minimal document might look like this:

  Example

  ```
  \documentclass{article}

  \usepackage{natbib}
  \bibliographystyle{sp}

  \begin{document}

  Here's a claim which needs justification \citep{something-which-justifies-it}.

  \bibliography{bibliography-name}

  \end{document}
  ```

- To get everything doing The Right Thing, you will initially need to run `latex`, then `bibtex`, then 2x `latex`.[2]

- (The first time, LaTeX makes a note in the `.aux` file of all of the citation commands; then BibTeX reads that file and makes a `.bbl` file containing all the entries correctly ordered; then LaTeX reads the `.bbl` file and makes a note of all the proper forms of the citations; finally, it uses these notes to fill in the text.)

- If you're using a front-end like TeXShop or Texmaker, to run BibTeX you need to either select it from the dropdown menu next to the compile button, and then hit compile, or else use a shortcut: for TeXShop it's Cmd-Shift-B, for Texmaker it's F11.

[2] This is one of the advantages of using something like `latexmk`: it will run whatever needs running as many times as it needs running.

9 AVMs

- Use the `avm` package (`https://nlp.stanford.edu/manning/tex/` – also on Weblearn).
- This provides the `avm` environment, which allows you to draw AVMs fairly intuitively.

Syntax

```
\[ attr1 & val1 \\
   attr2 & val2 \]
```

Example

```
\avmfont{\sc}
\avmvalfont{\rm}

\begin{avm}
\[
pred & ‘seem\sc$\langle$xcomp$\rangle$subj’\\
subj & \@{1}\[ pred & ‘bird’\\
               spec & \[ pred & ‘the’\] \\
               adj  & \{
                        \[ pred & ‘happy’\] \\
                        \[ pred & ‘brown’\]
                      \}
             \]\\
xcomp &      \[ pred & ‘eat$\langle$subj,obj$\rangle$’\\
                subj & \@1 \\
                obj  & \[ pred & ‘worm’\\
                           spec & \[ pred & ‘a’\]
                       \]
             \]\\
tense & \sc past
\]
\end{avm}
```

Example output

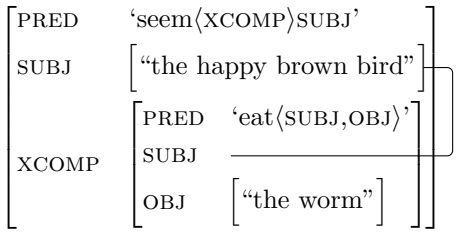
$$\begin{bmatrix} \text{PRED} & \text{‘seem}\langle\text{XCOMP}\rangle\text{SUBJ’} \\ \text{SUBJ} & \boxed{1}\begin{bmatrix} \text{PRED} & \text{‘bird’} \\ \text{SPEC} & \left[\text{PRED} \quad \text{‘the’}\right] \\ \text{ADJ} & \left\{\left[\text{PRED} \quad \text{‘happy’}\right], \left[\text{PRED} \quad \text{‘brown’}\right]\right\} \end{bmatrix} \\ \text{XCOMP} & \begin{bmatrix} \text{PRED} & \text{‘eat}\langle\text{subj,obj}\rangle\text{’} \\ \text{SUBJ} & \boxed{1} \\ \text{OBJ} & \begin{bmatrix} \text{PRED} & \text{‘worm’} \\ \text{SPEC} & \left[\text{PRED} \quad \text{‘a’}\right] \end{bmatrix} \end{bmatrix} \\ \text{TENSE} & \text{PAST} \end{bmatrix}$$

- If you want to draw lines in AVMs instead of using index boxes, you need to put the AVM inside a Ti*k*Z node. You can then use `\subnode{...}{...}` to create nodes inside this node. This requires you to load a Ti*k*Z library called `tikzmark`.
- Ti*k*Z libraries are like LaTeX packages; you load them with the command `\usetikzlibrary{...}` in your preamble.

Example

```
\usepackage{tikz}
\usetikzlibrary{tikzmark}
...
\begin{tikzpicture}[remember picture, anchor = center,inner sep = 0pt]
\node{
\begin{avm}
\[
    pred  & ‘seem\sc$\langle$xcomp$\rangle$subj’\\
    subj  & \subnode{subj}{\[\rm ‘‘the happy brown bird’’\]}\\
    xcomp & \[ pred & ‘eat\sc$\langle$subj,obj$\rangle$’\\
               subj & \subnode{xcomp-subj}{\phantom{x}}\\
               obj  & \[\rm‘‘the worm’’\] \]
\]
\end{avm}
};
\draw[shorten <= -1, rounded corners = 2pt]
    (subj.east) -- +(east:10pt) |- (xcomp-subj.west);
\end{tikzpicture}
```

Example output



10 Graphics

- Use the `graphicx` package.
- Lots of powerful features (see documentation), but we'll look at two frequently useful ones for now.
- Importing graphics:

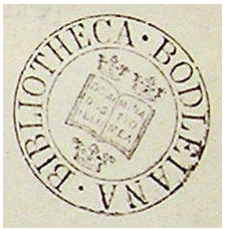
  Syntax

  ```
  \includegraphics[attr1=val1, attr2=val2, ...]{filename}
  ```

  Example

  ```
  \includegraphics[scale=0.5]{bod}
  ```

  Example output



- **Beware:** If you are compiling via `latex` rather than `pdflatex`, the image file *must* be an EPS (Encapsulated PostScript) file. If you are using `pdflatex` it can be a JPG, PNG, or PDF.

- Resizing diagrams, etc.:

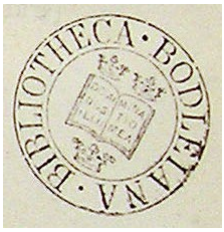
  Syntax

  `\scalebox{horizontal scale}[vertical scale]{object}`

  Example

  `\scalebox{0.5}{\includegraphics{bod}}`

  Example output

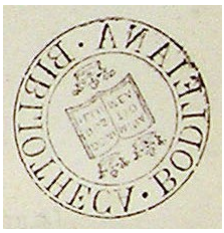
  

- Using a negative value for one of the scales allows you to flip the object.

  Example

  `\scalebox{0.5}[-0.5]{\includegraphics{bod}}`

  Example output

- There is also the `resizebox` command for fitting to a particular size:

  Syntax

  `\resizebox{horizontal length}{vertical length}{object}`

- Use `!` as the horizontal or vertical length to have it scale with the other one.
- E.g. use `\resizebox{\linewidth}{!}{object}` to fit the object in question to the width of the line (great for making sure figures, trees, diagrams fit on the page).
- *Tip:* If you want to store your figures somewhere other than the local directory, put `\graphicspath{{...}}` (note the extra `{}`s around the path) in the preamble, and this will set a different default path for `\includegraphics{...}`.