Introduction to LaTeX

Lecture IV: Graphs, Tables and Code

Liu Yihao

SJTU-UMJI Technology Department

April 16, 2020

Include Graphs

Before all, you need the `graphics` or `graphicx` package, where `graphicx` is an extended and enhanced one. So you are recommended to insert the command in the preamble of your document.

Command

```
\usepackage{graphicx}
```

Then you can use the command `\includegraphics` to insert images of many formats, including `jpg`, `png` images and even other `pdf` files. `eps` images should be supported by most modern LaTeX distributions as well.

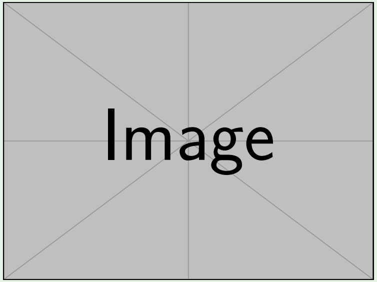
Command

```
\includegraphics[options]{filename}
```

There are some example images defined, you can insert them if the figure is not yet ready when writing LaTeX code. They are `example-image`, `example-image-golden`, `example-image-a`, `example-image-b` and etc.

Example

```
\includegraphics[width=0.4\textwidth]{example-image}
```



We usually use the `width` option to adjust the size of the image, according to a ratio of `\textwidth`, which means the maximum width of text here.

Options of Include Graphs

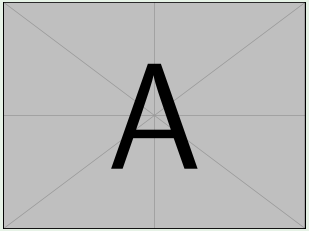
Here some useful `options` are listed:

- `height` - use any LaTeX measuring unit.
- `width` - use any LaTeX measuring unit.
- `scale` - scale the graph to this proportion
- `angle` - rotate the graph in anti-clockwise by this angle

LaTeX measuring unit can be `\textwidth`, `\linewidth`, `\textheight`, `\lineheight`, cm, pt, em, and etc..

Example

```
\includegraphics[width=4cm]%
{example-image-a}
```



The `figure` Environment

The `figure` environment provides a wrapper of image inserted by `\includegraphics`, which add caption and label (reference) to an image. They are especially useful in report and paper writing, here is a template of how to use the environment.

Command

```latex
\begin{figure}[position]
  \centering
  \includegraphics[options]{filename}
  \caption{caption}
  \label{fig:label}
\end{figure}
```

- `filename` - the filename or relative path of the graph you want to insert, usually placed in the same or child directory as the tex file
- `position` - we usually use `!htbp` or `!H` here, which will be introduced later in this chapter
- `caption` - the caption displayed above/under the graph
- `label` - used for references in a document (will be introduced later)

Labels and References

You can use `\ref` to have a reference of a figure by its label. The figures will be automatically numbered (like equations), and the reference is also a hyperlink.

Example

```
\begin{figure}[!htbp]
  \centering
  \includegraphics[
    width=0.8\textwidth,
    angle=90
  ]{example-image-b}
  \caption{Example Image B rotated by 90
  ↪  degree.}
  \label{fig:img-b}
\end{figure}
B was shown in Figure
\ref{fig:img-b}.
```

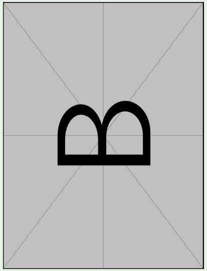


Figure 1: Example Image B rotated by 90 degree.

B was shown in Figure 1.

Floats and Positions

Floats are containers for things in a document that cannot be broken over a page. LaTeX by default recognizes `figure` and `table` (will be introduced later) floats.

If you don't provide the `position` option, LaTeX will try to help you find a place to set the figure. However, the position is often not ideal, so you need to add some specifiers yourselves.

- `h` - Place the float here, i.e., approximately at the same point it occurs in the source text (however, not exactly at the spot)
- `t` - Position at the top of the page.
- `b` - Position at the bottom of the page.
- `p` - Put on a special page for floats only.
- `!` - Override internal parameters LaTeX uses for determining "good" float positions.
- `H` - Places the float at precisely the location in the LaTeX code. Requires the float package, i.e., `\usepackage{float}`.

Include Multiple Graphs

A useful extension is the `subcaption` package, which provides a `subfigure` environment to add multiple subfigures in a figure.

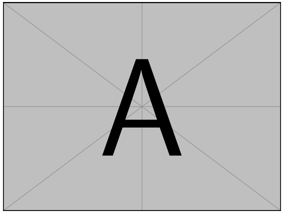
Note that there is also a package called `subfigure`, but is has been deprecated (not maintained), please do not use it. Another package called `subfig` provides the same commands as that of `subfigure` package. However, they can't be used together.

In simplicity, if there is some compatibility problem with your template after you tried the `subcaption` package, choose the `subfig` package.

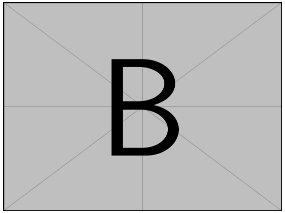
Here is an example with the `subcaption` package.

Example

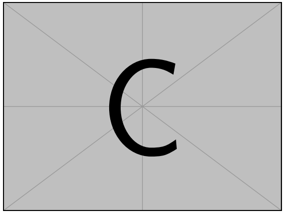
```latex
\begin{figure}
    \centering
    \begin{subfigure}{0.3\textwidth}
        \includegraphics[width=\textwidth]{example-image-a}
        \caption{Example Image A.}
        \label{fig:subcaption-a}
    \end{subfigure}
    ~
    \begin{subfigure}{0.3\textwidth}
        \includegraphics[width=\textwidth]{example-image-b}
        \caption{Example Image B.}
        \label{fig:subcaption-b}
    \end{subfigure}

    \begin{subfigure}{0.3\textwidth}
        \includegraphics[width=\textwidth]{example-image-c}
        \caption{Example Image C.}
        \label{fig:subcaption-c}
    \end{subfigure}
    \caption{Example Images}\label{fig:subcaption}
\end{figure}
```



(a) Example Image A.



(b) Example Image B.



(c) Example Image C.

Figure 2: Example Images

As shown in Figure 2, the figures can be arranged in columns and rows.

Between Figure 2a and Figure 2b, a ~ was added. You can add desired spacing between images, e. g. ~, `\quad`, `\qquad`, `\hfill` (fill all rest horizontal spaces) and etc..

Between Figure 2b and Figure 2c, a newline was added. It will force the subfigure onto a new line.

The references of subfigures can be used by their `\label` as well. For example, above references are generated by these commands:

Example

```
\ref{fig:subcaption}
\ref{fig:subcaption-a}
\ref{fig:subcaption-b}
\ref{fig:subcaption-c}
```

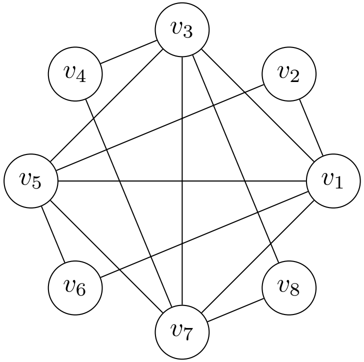
The tikz and pgf packages

The tikz and pgf packages can help you draw graphs in LaTeX for example:

Example

```
\begin{tikzpicture}[scale=2, bend angle=22.5]
\tikzstyle{every node}=[draw,shape=circle];
\foreach \i in {1,...,8}
{
\path (45*\i-45:1cm) node (v\i) {$v_\i$};
}
\draw
(v1) -- (v2) (v3) -- (v4) (v5) -- (v6) (v7) -- (v8)
(v1) -- (v3) (v3) -- (v5) (v5) -- (v7) (v7) -- (v1)
(v2) -- (v5) (v4) -- (v7) (v6) -- (v1) (v8) -- (v3)
(v1) -- (v5) (v3) -- (v7);
\end{tikzpicture}
```

This will generate a simple graph which consists of eight nodes:



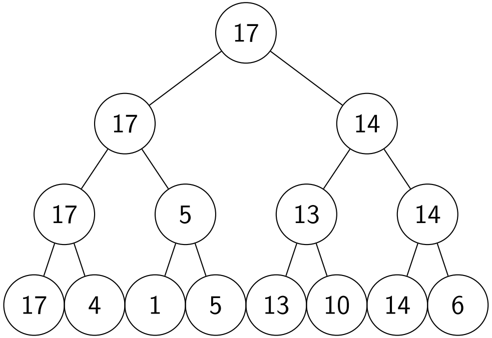
There may be a lecture about `tikz` and `pgf` in the future. If you are now interested in it, please refer to the pgf manuel by `texdoc tikz` or `texdoc pgf`.

Another example:

Example

```
\begin{tikzpicture}[scale=0.8]
\tikzstyle{every node}=[draw,shape=circle,minimum size=0.8cm];
\node {17}[sibling distance=4cm]
child { node {17}[sibling distance=2cm]
    child {
        node {17}[sibling distance=1cm]
        child { node {17} }
        child { node {4} }
    }
    child {
        node {5}[sibling distance=1cm]
        child { node {1} }
        child { node {5} }
    }
}
child { node {14}[sibling distance=2cm]
    child {
        node {13}[sibling distance=1cm]
        child { node {13} }
        child { node {10} }
        }
        child {
            node {14}[sibling distance=1cm]
            child { node {14} }
            child { node {6} }
        }
    };
\end{tikzpicture}
```

This will generate a binary tree:



The tabular Environment

Table is another common element in LaTeX, usually you will need the array package for enhanced functions of tables. You can insert the command in the preamble of your document.

Command

```
\usepackage{array}
```

Example

```
\begin{tabular}{|l|c|r|}
  \hline
  Title 1 & Title 2 & Title 3 \\
  \hline
  1 & 2 & 3 \\
  \hline
\end{tabular}
```

Title 1	Title 2	Title 3
1	2	3

The syntax is similar to the align environment in maths. & is used to split the columns are \\ is used to split the rows.

Column Format

Command

```
\begin{tabular}{format}
...
\end{tabular}
```

`format` can be set as follow:

- `|` - represents a vertical separate line between two columns
- `l` - align left in this column
- `c` - align center in this column
- `r` - align right in this column

Example

`|l|l|l|`

Title 1	Title 2	Title 3
1	2	3

`||c|cc||`

Title 1	Title 2	Title 3
1	2	3

With the help of the `array` package, more formats are available:

- `p{width}` - Equivalent to `\parbox[t]{width}`, vertically aligned bottom
- `b{width}` - Equivalent to `\parbox[b]{width}`, vertically aligned top
- `m{width}` - Equivalent to `\parbox{width}`, vertically aligned middle
- `>{decl.}` - Can be used after a letter option, inserts `decl` before the entry.
- `<{decl.}` - Can be used before a letter option, inserts `decl` after the entry.

`t` and `b` may be very confusing, but that's how they work in `\parbox`. With these new formats, the columns can be defined more flexibly.

Example

```latex
\begin{tabular}
{|p{1.2cm}|b{1.2cm}|m{1.2cm}|}
  \hline
  Aligned Bottom & Aligned Top &
  Aligned Middle \\
  \hline
  1 & 2 & 3 \\
  \hline
\end{tabular}
```

Aligned Bottom	Aligned Top	Aligned Middle
1	2	3

`t`, `b` and `m` only affect the vertical alignment. If you want to control the width and make the text horizontally centered as well, you can use `>{\centering}` to insert a `\centering` before the text in that column. You can also insert `>{$}` and `<{$}` to generate a column in math mode.

Example

```latex
\begin{tabular}{|>{\centering}m{2cm}|>{$}b{2cm}<{$}|}
  \hline
  Row of Text &
  \text{Row of Maths} \\
  \hline
  First & x \\
  Second & x^2 \\
  \hline
\end{tabular}
```

Row of Text	Row of Maths
First	x
Second	x^2

If a column type will be used many times, and also very long, you can define a new column type by yourselves. You can use

Command

```latex
\newcolumntype{new type}{>{some declarations}{old type}<{some more declarations}}
```

If you want to repeat a format for multiple times, you can use `*{num}{format}`. Here's an example of the usage of `\newcolumntype` with multiple columns form.

Example

```
\newcolumntype{C}{>{$}c<{$}}
\newcolumntype{L}{>{$}l<{$}}
\newcolumntype{R}{>{$}r<{$}}

\begin{tabular}{|L| *{2}{C|} R|}
  \hline
  \text{First} & \text{Second} &
  \text{Second} & \text{Third} \\
  \hline
  x & x^2 & x^2 & x^3 \\
  \hline
  y & y^2 & y^2 & y^3 \\
  \hline
\end{tabular}
```

First	Second	Second	Third
x	x^2	x^2	x^3
y	y^2	y^2	y^3

Horizontal Lines

We usually need horizontal lines in tables. As shown in the examples above, you can add a `\hline` at the beginning of a row.

If you only want to draw a partial line, use `\cline[start-end]`.

Example

```latex
\begin{tabular}{c|l|c|r}
  \hline\hline
  & Title 1 & Title 2 & Title 3 \\
  \cline{2-4}
  Table & 1 & 2 & 3 \\
  \cline{2-4}
  & 4 & 5 & 6 \\
  \hline\hline
\end{tabular}
```

	Title 1	Title 2	Title 3
Table	1	2	3
	4	5	6

Here we draw a table with a multirow, but it only works with multirows of odd row number. A more convenient method of drawing multirows will be introduced.

Combine Rows and Columns

There are two commands being used to combine rows and columns

Command

`\multicolumn{ncols}{format}{text}`

- `ncols` - the number of columns to be merged
- `format` - the format of the merged column, excluding the left `|` (eg. `c|`)
- `text` - the text in the merged column

`\multirow{nrows}{width}[fixup]{text}`

- `nrows` - the number of rows to be merged
- `width` - the width of the merged rows (use `*` for auto)
- `fixup` - the vertical position of the text (optional, default in the center)
- `text` - the text in the merged row

To use the `\multirow` command, you need to insert the package `multirow` in the preamble of your document.

Example

```latex
\centering
\begin{tabular}{|c|c|c|c|c|}
  \hline
  \multirow{4}{*}{Table} & Title 1 & Title 2 & Title 3 & Title 4 \\
  \cline{2-5}
  & \multicolumn{2}{c|}{Text 1} &
  \multicolumn{2}{c|}{\multirow{3}{*}{Text 3}} \\
  \cline{2-3}
  & \multicolumn{2}{c|}{Text 2} & \multicolumn{2}{c|}{} \\
  \cline{2-3}
  & Text 4 & Text 5 & \multicolumn{2}{c|}{} \\
  \hline
\end{tabular}
```

Table	Title 1	Title 2	Title 3	Title 4
	Text 1		Text 3	
	Text 2			
	Text 4	Text 5		

Just leave blank in the rest rows of `\multirow`.

Table Generators

With `\multirow` and `\multicolumn`, we can almost draw tables of any style, but this coding process can never be as easy as the graphic one, like making tables in Word or Excel. Is there any ways to convert graphic tables into LaTeX codes directly?

- Use LaTeX Table Generator: `http://www.tablesgenerator.com/`
- LaTeX Complex Table Editor: `https://www.latex-tables.com/`
- Excel2latex: `https://ctan.org/tex-archive/support/excel2latex/`

The table Environment

The `table` environment is used to arrange the place of a tabular, similar to the `figure` environment. Here is a template of how to use the environment.

Command

```latex
\begin{table}[position]
  \centering
  \begin{tabular}{format}
    ...
  \end{tabular}
  \caption{caption}
  \label{table:label}
\end{table}
```

The `position`, `caption`, `label` are same as those in the `figure` environment.

Recall the Positions

We usually want to place the graphs or tables just below or above the content where we mention them, but even when we type `[h]` in position, you can not ensure that it will appear at the ideal position, and there are several methods to make up for this. You can try them one by one:

1. Change `[h]` to `[!h]`
2. Change `[!h]` to `[!H]`
3. Use `\newpage` to move the following content to the next page

Usually you don't need to pay too much attention about where the figures and tables are exactly are because you can use `\ref` to reference them. And the numbering of figures and tables will strictly follow the order of their code.

`figure` and `table` in Two-column Documents

If you are writing a document using two columns (i.e. you started your document with something like `\documentclass[twocolumn]{article}`), you might have noticed that you can't use floating elements that are wider than the width of a column (using a LaTeX notation, wider than `0.5\textwidth`), otherwise you will see the figure or table overlapping with text.

If you really have to use such wide elements, the only solution is to use the "starred" variants of the floating environments:

Command

```
\begin{figure*}[position]
  ...
\end{figure*}
```

```
\begin{table*}[position]
  ...
\end{table*}
```

Those "starred" versions work like the standard ones, but they will be as wide as the page, so you will get no overlapping.

The `array` Environment

When you use `tabular` in maths environment, the text format in the `tabular` won't be italic. However, there is a replacement of `tabular`, which is the `array` environment.

Command

```latex
\begin{array}{format}
  ...
\end{array}
```

The options and usages of these two environment are exactly the same.

Though the environment is not provided by the `array` package (it's built-in one), you are also recommended to use this package for enhancements.

The algorithm Environment

LaTeX has several packages for typesetting algorithms in form of "pseudocode". They provide stylistic enhancements over a uniform style (i.e., all in typewriter font) so that constructs such as loops or conditionals are visually separated from other text. The pseudocode is usually put in an algorithm environment. Include it by adding the command to your document's preamble.

Command

```
\usepackage{algorithm}
```

Then you can use the algorithm environment, which acts similar as the figure and table environments.

Command

```
\begin{algorithm}[position]
  \caption{caption}
  \label{algorithm:label}
  <the actual pseudocode environment>
\end{algorithm}
```

The algorithmic Package

One of the packages, the algorithmic, defines the algorithmic environment. Include it by adding the command to your document's preamble.

Command

```
\usepackage{algorithmic}
```

The basic commands are:

Command

```
\STATE <text>
\IF{<condition>} \STATE {<text>} \ELSE \STATE{<text>} \ENDIF
\IF{<condition>} \STATE {<text>} \ELSIF{<condition>} \STATE{<text>} \ENDIF
\FOR{<condition>} \STATE {<text>} \ENDFOR
\FOR{<condition> \TO <condition> } \STATE {<text>} \ENDFOR
\FORALL{<condition>} \STATE{<text>} \ENDFOR
\WHILE{<condition>} \STATE{<text>} \ENDWHILE
\REPEAT \STATE{<text>} \UNTIL{<condition>}
\LOOP \STATE{<text>} \ENDLOOP
\REQUIRE <text>, \ENSURE <text>, \RETURN <text>, \PRINT <text>
\AND, \OR, \XOR, \NOT, \TO, \TRUE, \FALSE, \COMMENT{<text>}
```

Example

```
\begin{algorithm}[H]
  \caption{Calculate $y = x^n$}
  \label{algorithm:n-square}
  \begin{algorithmic}
    \REQUIRE $n \geq 0 \vee x \neq 0$
    \ENSURE $y = x^n$
    \STATE $y \leftarrow 1$
    \IF{$n < 0$}
      \STATE $X \leftarrow 1 / x$
      \STATE $N \leftarrow -n$
    \ELSE
      \STATE $X \leftarrow x$
      \STATE $N \leftarrow n$
    \ENDIF
    \WHILE{$N \neq 0$}
      \IF{$N$ is even}
        \STATE $X \leftarrow X \times X$
        \STATE $N \leftarrow N / 2$
      \ELSE[$N$ is odd]
        \STATE $y \leftarrow y \times X$
        \STATE $N \leftarrow N - 1$
     \ENDIF
    \ENDWHILE
  \end{algorithmic}
\end{algorithm}
```

Algorithm 1 Calculate $y = x^n$

```
Require: $n \geq 0 \vee x \neq 0$
Ensure: $y = x^n$
  $y \leftarrow 1$
  if $n < 0$ then
      $X \leftarrow 1/x$
      $N \leftarrow -n$
  else
      $X \leftarrow x$
      $N \leftarrow n$
  end if
  while $N \neq 0$ do
      if $N$ is even then
          $X \leftarrow X \times X$
          $N \leftarrow N/2$
      else[$N$ is odd]
          $y \leftarrow y \times X$
          $N \leftarrow N - 1$
      end if
  end while
```

The algorithmicx Package

Another package algorithmicx provides more functionalities, but it is not compatible with the algorithmic package. Include it by adding the command to your document's preamble.

Command

```
\usepackage{algpseudocode}
```

Note that `\usepackage{algorithmicx}` only defines some common macros and it is not enough. Don't insert `\usepackage{algorithmic}` in this situation.

The main difference of these two packages is that all of the command name are changed, so that only the first letter in a word is capital. For example, `\STATE` is changed to `\State` and `\ENDFOR` is changed to `\EndFor`.

The command `\begin{algorithmic}` can be given the optional argument of a positive integer, which if given will cause line numbering to occur at multiples of that integer. E.g. `\begin{algorithmic}[5]` will enter the algorithmic environment and number every fifth line.

Example

```
\begin{algorithm}[H]
  \caption{Euclids algorithm}
  \label{algorithm:euclid}
  \begin{algorithmic}[1]
    \Procedure{Euclid}{$a,b$}\Comment{The g.c.d. of a and b}
    \State $r\gets a\bmod b$
      \While{$r\not=0$}\Comment{We have the answer if r is 0}
        \State $a\gets b$
        \State $b\gets r$
        \State $r\gets a\bmod b$
      \EndWhile\label{euclidendwhile}
      \State \textbf{return} $b$\Comment{The gcd is b}
    \EndProcedure
  \end{algorithmic}
\end{algorithm}
```

Algorithm 2 Euclids algorithm

```
1: procedure EUCLID($a, b$)                          ▷ The g.c.d. of a and b
2:     $r \leftarrow a \bmod b$
3:     while $r \neq 0$ do                            ▷ We have the answer if r is 0
4:         $a \leftarrow b$
5:         $b \leftarrow r$
6:         $r \leftarrow a \bmod b$
7:     end while
8:     return $b$                                     ▷ The gcd is b
9: end procedure
```

Algorithms can also be listed like figures and tables, by the command:

Command

```
\listofalgorithms
```

The verbatim Environment

The default tool to display code in LaTeX is verbatim, which generates an output in monospaced font.

Example

```
\begin{verbatim}
Text enclosed inside \texttt{verbatim} environment
is printed directly
and all \LaTeX{} commands are ignored.
\end{verbatim}
```

```
Text enclosed inside \texttt{verbatim} environment
is printed directly
and all \LaTeX{} commands are ignored.
```

There's a starred version (`verbatim*`) whose output is slightly different.

```
Text␣enclosed␣inside␣\texttt{verbatim}␣environment
is␣printed␣directly
and␣all␣\LaTeX{}␣commands␣are␣ignored.
```

The verb Command

Verbatim-like text can also be used inline with the command `\verb`

Example

```
In the directory \verb|C:\Windows\system32| you can find a lot of Windows
system applications.

The \verb+\ldots+ command produces \ldots
```

In the directory `C:\Windows\system32` you can find a lot of Windows system applications.
The `\ldots` command produces . . .

The command `\verb|C:\Windows\system32|` prints the text inside the delimiters `|` in verbatim format. Any character, except letters and `*`, can be used as delimiter. For instance `\verb+\ldots+` uses `+` as delimiter.

The `listings` Package

A better form of code listing can be done by the `listings` package. To use it, simply insert the command in the preamble of your document.

Command

```
\usepackage{listings}
```

It provides a `lstlisting` environment.

Command

```
\begin{lstlisting}[language=name]
  ...
\end{lstlisting}
```

You can also input source code from file.

Command

```
\lstinputlisting[language=name]{filename}
```

Example

```
\begin{lstlisting}[language=Python]
import numpy as np

def incmatrix(genl1,genl2):
    m = len(genl1)
    n = len(genl2)
\end{lstlisting}
```

```
import numpy as np

def incmatrix(genl1,genl2):
    m = len(genl1)
    n = len(genl2)
```

You can add code coloring and styling by some complicated configurations, see the Overleaf tutorial ▸ Link.

The documentation of the `listings` package can be found in ▸ Link.

The minted Package

All of the code in this lecture are highlighted by the `minted` package. To use it, simply insert the command in the preamble of your document.

Command

```
\usepackage{minted}
```

This is a very special package, it depends a program out of LaTeX called `pygmentize`, which is a code highlighting package written in `Python`.

You can install the package through `pip` (assuming you have `Python` 2 or 3 and `pip` installed) in your terminal:

Command

```
pip install Pygments
```

And then you can examine in your terminal whether `pygmentize` is your `PATH` by directly running it. You also need to add an option `-shell-escape` to your LaTeX compiler because LaTeX need this permission to run other programs on shell.

The minted Environment

You can use the `minted` environment to insert a block of code in the specific language.

Command

```
\begin{minted}[options]{language}
  ...
\end{minted}
```

You can use the command in the terminal to find the supported languages.

Command

```
pygmentize -L lexers
```

There is also a list of languages on the online document Link. Note that if you want to insert plain text, use the `text` language which doesn't have any highlight.

The Inline `minted`

For a single line of source code, you can alternatively use a shorthand notation:

Command

```
\mint[options]{language}|...|
```

Here we use a pair of `|`, same as the usage of the `\verb` command, which is also an inline verbatim command.

Or you can also use

Command

```
\mintinline[options]{language}|...|
```

Here `|` can also be replaced with `{}`, a pair of `+`, etc., the key is there should not exist the same delimiter inside the code.

Input File with `minted`

When you have a source code file alone, you can use the command to input the file.

Command

```
\inputminted[options]{language}{filename}
```

There are some commonly used options (not only for this command):

- `linenos` - Turn on line numbers
- `breaklines` - Automatically break long lines in `minted` environment and `\mint`, and wrap longer lines in `\mintinline`.
- `fontsize` - The size of the font to use, as a size command, e.g. `fontsize=\footnotesize`.
- `tabsize` - The number of spaces a tab is equivalent to. (default is 8, but often set to 4)
- `firstline` - The first line to show. (default is 1, useful when showing part of a file)
- `lastline` - The last line to show. (default is the last line of the input)

Using Different Styles

You can use various styles of highlighting scheme provided by `pygmentize`.

Command

```
\usemintedstyle{name}
```

You can use the command in the terminal to find the supported styles.

Command

```
pygmentize -L styles
```

There is also a demo of languages and styles on the online demo ▸ Link. The `autumn` style is used in this lecture.

In the end, `XeLaTeX` might be the best LaTeX compiler working with the `minted` package. It also supports typesetting with Chinese, if you meet problems when using the default `pdflatex` compiler, switch into `XeLaTeX` may solve your issues.

The documentation of the `minted` package can be found in ▸ Link.