Learning LaTeX

Anzong Zheng

February 14, 2015

Contents

Chapter 1

Table of contents

Command to tell LaTeXto show table of contents

```
\tableofcontents
```

Listing 1.1: Add table of contents

Add certain title to the TOC.

```
\section*{Abstract}
\addcontentsline{toc}{chapter}{Abstract}
```

Listing 1.2: Add table of contents

Chapter 2

Code listing

LaTeX is widely used in science and programming has become an important aspect in several areas of science, hence the need for a tool that properly displays code. In this article is explained how to use the standard **verbatim** environment as well as the package **listings**, which provide more advanced code formatting features.

2.1 Introduction

Displaying code in LaTeXis straightforward. For instance, using the **lstlisting** environment:

```
import numpy as np

def incmatrix(genl1,genl2):
    m = len(genl1)
    n = len(genl2)
    M = None #to become the incidence matrix
    VT = np.zeros((n*m,1), int)  #dummy variable

    #compute the bitwise xor matrix
    M1 = bitxormatrix(genl1)
    M2 = np.triu(bitxormatrix(genl2),1)

    for i in range(m-1):
        for j in range(i+1, m):
            [r,c] = np.where(M2 == M1[i,j])
                for k in range(len(r)):
                    VT[(i)*n + r[k]] = 1;
                    VT[(i)*n + c[k]] = 1;
                    VT[(j)*n + r[k]] = 1;
                    VT[(j)*n + c[k]] = 1;

                    if M is None:
                        M = np.copy(VT)
                    else:
                        M = np.concatenate((M, VT), 1)

```

```
27                      VT = np.zeros((n*m,1), int)
28
29     return M
```

In this example, the outupt ignores all LaTeX commands and the text is printed keeping all the line breaks and white spaces typed. To use the ***lstlisting*** environment you have to add the next line to the preamble of your document:

```
\usepackage{listings}
```

2.2 The verbatim environment

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

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

Just as in the example at the introduction, all text is printed keeping line breaks and white spaces. There's a starred version of this command whose output is slightly different.

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

In this case white spaces are emphasized with a special symbol.

Verbatim-like text can also be used in a paragraph by means of the \verb command.

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+...+ uses + as delimiter.

2.3 Using listings to highlight code

In the introduction a basic example of the package **listings** was presented, let's see a second example:

```
1 import numpy as np
2
3 def incmatrix(genl1,genl2):
4     m = len(genl1)
5     n = len(genl2)
6     M = None #to become the incidence matrix
```

```
7       VT = np.zeros((n*m,1), int)  #dummy variable
8
9       #compute the bitwise xor matrix
10      M1 = bitxormatrix(genl1)
11      M2 = np.triu(bitxormatrix(genl2),1)
12
13      for i in range(m-1):
14          for j in range(i+1, m):
15              [r,c] = np.where(M2 == M1[i,j])
16              for k in range(len(r)):
17                  VT[(i)*n + r[k]] = 1;
18                  VT[(i)*n + c[k]] = 1;
19                  VT[(j)*n + r[k]] = 1;
20                  VT[(j)*n + c[k]] = 1;
21
22                  if M is None:
23                      M = np.copy(VT)
24                  else:
25                      M = np.concatenate((M, VT), 1)
26
27                  VT = np.zeros((n*m,1), int)
28
29      return M
```

The additional parameter inside brackets [language=Python] enables code highlighting for this particular programming language (Python), special words are in boldface font and comments are italicized. See the reference guide for a complete list of supported programming languages.

2.4 Importing code from a file

Code is usually stored in a source file, therefore a command that automatically pulls code from a file becomes very handy.

```
The next code will be directly imported from a file

\lstinputlisting[language=Octave]{BitXorMatrix.m}
```

The command \lstinputlisting[language=Octave]{BitXorMatrix.m} imports the code from the file *BitXorMatrix.m*, the additional parameter in between brackets enables language highlighting for the Octave programming language. If you need to import only part of the file you can specify two comma-separated parameters inside the brackets. For instance, to import the code from the line 2 to the line 12, the previous command becomes

```
\lstinputlisting[language=Octave, firstline=2, lastline=12]{BitXorMatrix.m}
```

If firstline or lastline is omitted, it's assumed that the values are the beginning of the file, or the bottom of the file, respectively.

2.5 Code styles and colours

Code formatting with the **listing** package is highly customisable. Let's see an example

```
\documentclass{article}
\usepackage[utf8]{inputenc}

\usepackage{listings}
\usepackage{color}

\definecolor{codegreen}{rgb}{0,0.6,0}
\definecolor{codegray}{rgb}{0.5,0.5,0.5}
\definecolor{codepurple}{rgb}{0.58,0,0.82}
\definecolor{backcolour}{rgb}{0.95,0.95,0.92}

\lstdefinestyle{zstyle}{
    basicstyle=\small\tt,
    backgroundcolor=\color{backcolour},
    commentstyle=\color{codegreen},
    keywordstyle=\color{magenta},
    numberstyle=\tiny\color{codegray},
    stringstyle=\color{codepurple},
    %basicstyle=\footnotesize,
    breakatwhitespace=false,
    breaklines=true,
    captionpos=b,
    keepspaces=true,
    numbers=left,
    numbersep=5pt,
    showspaces=false,
    showstringspaces=false,
    showtabs=false,
    tabsize=2
}

\lstset{style=zstyle}

\begin{document}
The next code will be directly imported from a file

\lstinputlisting[language=Octave]{BitXorMatrix.m}
\end{document}
```

Listing 2.1: Code styles and colours

As you see, the code colouring and styling greatly improves readability.

In this example the package **color** is imported and then the command \definecolor{}{}{} is used to define new colours in rgb format that will later be used. The package **xcolor** also works for this. For more information see: using colours in LaTeX

There are essentially two commands that generate the style for this example:

`\lstdefinestyle{mystyle}{...}`

Defines a new code listing style called "mystyle". Inside the second pair of braces the options that define this style are passed; see the reference guide for a full description of these and some other parameters.

`\lstset{style=mystyle}`

Enables the style "mystyle". This command can be used within your document to switch to a different style if needed.

2.6 Captions and the list of Listings

Just like in floats (tables and figures), captions can be added to listing for a more clear presentation.

```
import numpy as np

    def incmatrix(genl1,genl2):
    m = len(genl1)
    n = len(genl2)
    M = None #to become the incidence matrix
    VT = np.zeros((n*m,1), int)  #dummy variable

    #compute the bitwise xor matrix
    M1 = bitxormatrix(genl1)
    M2 = np.triu(bitxormatrix(genl2),1)

    for i in range(m-1):
        for j in range(i+1, m):
            [r,c] = np.where(M2 == M1[i,j])
            for k in range(len(r)):
                VT[(i)*n + r[k]] = 1;
                VT[(i)*n + c[k]] = 1;
                VT[(j)*n + r[k]] = 1;
                VT[(j)*n + c[k]] = 1;

                if M is None:
                    M = np.copy(VT)
                else:
                    M = np.concatenate((M, VT), 1)

                VT = np.zeros((n*m,1), int)

    return M
```

Listing 2.2: Python example

Adding the comma-separated parameter caption=Python example inside the brackets, enables the caption. This caption can be later used in the list of Listings.

\lstlistoflistings

2.7 Reference guide

Supported languages

supported languages (and its dialects if possible, dialects are specified in brackets and default dialects are italized):

Table 2.1: supported languages of listings

ABAP (R/2 4.3, R/2 5.0, R/3 3.1, R/3 4.6C, R/3 6.10)	ACSL
Ada (2005, 83, 95)	Algol (60, 68)
Ant	Assembler (Motorola68k, x86masm)
Awk (gnu, POSIX)	bash
Basic (Visual)	C (ANSI, Handel, Objective, Sharp)
C++ (ANSI, GNU, ISO, Visual)	Caml (light, Objective)
CIL	Clean
Cobol (1974, 1985, ibm)	Comal 80
command.com (WinXP)	Comsol
csh	Delphi
Eiffel	Elan
erlang	Euphoria
Fortran (77, 90, 95)	GCL
Gnuplot	Haskell
HTML	IDL (empty, CORBA)
inform	Java (empty, AspectJ)
JVMIS	ksh
Lingo	Lisp (empty, Auto)
Logo	make (empty, gnu)
Mathematica (1.0, 3.0, 5.2)	Matlab
Mercury	MetaPost
Miranda	Mizar
ML	Modula-2
MuPAD	NASTRAN
Oberon-2	OCL (decorative, OMG)
Octave	Oz
Pascal (Borland6, Standard, XSC)	Perl
PHP	PL/I
Plasm	PostScript
POV	Prolog
Promela	PSTricks
Python	R
Reduce	Rexx
RSL	Ruby
S (empty, PLUS)	SAS
Scilab	sh
SHELXL	Simula (67, CII, DEC, IBM)
SPARQL	SQL
tcl (empty, tk)	TeX (AlLaTeX, common, LaTeX, plain, p
VBScript	Verilog
VHDL (empty, AMS)	VRML (97)
XML	XSLT

2.7.1 Options to customize code listing styles

- **backgroundcolor** - colour for the background. External ***color*** or ***xcolor*** package needed.
- **commentstyle** - style of comments in source language.
- **basicstyle** - font size/family/etc. for source (e.g. basicstyle=\ttfamily\small)
- **keywordstyle** - style of keywords in source language (e.g. keywordstyle=\color {red})
- **numberstyle** - style used for line-numbers
- **numberserp** - distance of line-numbers from the code
- **stringstyle** - style of strings in source language
- **showspaces** - emphasize spaces in code (true/false)
- **showstringspaces** - emphasize spaces in strings (true/false)
- **showtabs** - emphasize tabulators in code (true/false)
- **numbers** - position of line numbers (left/right/none, i.e. no line numbers)
- **prebreak** - displaying mark on the end of breaking line (e.g. prebreak=\raisebox{0ex}[0ex][0ex]{\ensuremath{
- **captionpos** - position of caption (t/b)
- **frame** - showing frame outside code (none/leftline/topline/bottomline/lines/single/shadowbox)
- **breakwhitespace** - sets if automatic breaks should only happen at whitespaces
- **breaklines** - automatic line-breaking
- **keepspaces** - keep spaces in the code, useful for indetation tabsize - default tabsize
- **escapeinside** - specify characters to escape from source code to LATEX (e.g. escapeinside={%*}{*)})
- **rulecolor** - Specify the colour of the frame-box

2.8 Add more keywords

Keywords are closely related to languages, so it should not be specified when setting language style. It should be used like:

```
1 \begin{lstlisting}[language=Python, morekeywords={as}, ↩
    caption=Python example]
```

Or if one language is extensively used, then do like:

```
\lstset{style=zstyle, language=python, morekeywords={as}}
\begin{lstlisting}
import numpy as np

def incmatrix(genl1,genl2):
    m = len(genl1)
    n = len(genl2)
    M = None #to become the incidence matrix
    VT = np.zeros((n*m,1), int)  #dummy variable

    #compute the bitwise xor matrix
    M1 = bitxormatrix(genl1)
    M2 = np.triu(bitxormatrix(genl2),1)

    for i in range(m-1):
        for j in range(i+1, m):
        [r,c] = np.where(M2 == M1[i,j])
            for k in range(len(r)):
                VT[(i)*n + r[k]] = 1;
                VT[(i)*n + c[k]] = 1;
                VT[(j)*n + r[k]] = 1;
                VT[(j)*n + c[k]] = 1;

                if M is None:
                    M = np.copy(VT)
                else:
                    M = np.concatenate((M, VT), 1)

                VT = np.zeros((n*m,1), int)

    return M
```