用 Markdown+Pandoc+XeLaTeX 写作

An He

Published by LaTeX
https://github.com/annProg/pandoc-template

最后编译日期*, 2017* 年 *9* 月 *13* 日 *18:02:00*

前言

LaTeX 可以排版格式精美的书籍，但是学习成本较高，使用不便；Markdown 是一种简单易学的标记语言。如果能结合两者的优点，使用 Markdown 来写作，然后通过程序转换为 latex 源码，再编译为 PDF，那将是一件美妙的事情。幸运的是，已经有工具很好的实现了支持这一功能，那就是 Pandoc。

Pandoc 是由 John MacFarlane 教授开发的标记语言转换工具，实现了数十种标记语言之间的转换。Pandoc 还扩展了 Markdown 语法，比如标题表格等的 ID 属性，脚注等，并且可以直接嵌入 LaTeX 代码，这样在 Markdown 中就可以实现输入数学公式，交叉引用等功能。Pandoc 还支持自定义转换模板，通过命令`pandoc -D latex`可以输出默认的 LaTeX 模板，以此模板为基础，可以定制自己的模板。

本工具定义了一种 Markdown 源码组织规范，提供了一个转换脚本，用来更方便的使用 Pandoc 将 Markdown 转换为 PDF。另外还定义了一套 LaTeX 书籍模板，用来生成中文书籍。用户也可以在自己的工作目录修改此模板，并通过修改配置来引用自己的模板。

目录

表格

插图

第一章　使用说明

1.1　安装步骤

首先克隆代码库

```
git clone https://github.com/annProg/pandoc-template
```

然后将 pandoc-template 目录加入环境变量

建立工作目录

```
mkdir workdir
cd workdir
bookgen.sh init # 初始化工作环境
```

目录结构说明

```
.
    book                                    # 书籍模板，暂未用到
        book-template.latex
        pm-template.latex
    bookgen.sh                              # 转换脚本
    build                                   # 电子书构建目录
    config.default                          # pandoc默认转换配置
    html5                                   # html5电子书模板
    README.md
    resume                                  # 简历模板
    src                                     # Markdown源码目录
        images                              # 源码涉及插图目录
    zh-ctex                                 # ctexbook模板目录
        Pictures                            # 模板引用的图片资源
```

1.2　使用规范

1.2.1　源码命名规范

脚本中使用`ls src/*.md`列出所有的 Markdown 源码，要保证顺序正确，才能生成正确的 LaTeX 源码。因此，要求 Markdown 源码文件命名能够被 ls 以正确的顺序列出。例如，有少于十个的 Markdown

文件，可以使用 0~9 为前缀：

```
$ tree src/
src/
    0-title.md
    1-intro.md
    2-pandoc-markdown.md
    3-template.md
    images
```

如果 Markdown 文件数多于 10 个，则需要在前缀为个位数的前面补 0，与最大前缀数字位数保持一致，例如最后一个 Markdown 文件为`99-markdown.md`，那么个位数应形如 `01-first.md`。

1.2.2 编码规范

Markdown 源码文件需要使用 UTF-8 编码。以 Notepad++ 为例，依次选择格式，以 **UTF-8 无 BOM** 格式编码即可正确设置编码。

1.2.3 注意事项

Pandoc 扩展的 Markdown 语法要求在标题前留出一个空行，因此按章节拆分的多个 Markdown 文件，开头需要空一行，否则 pandoc 不能正确识别标题。

1.3 模板变量说明

可以使用 Yaml 语言[1] 定义模板中的变量，建议第一个 Markdown 文件专门用来定义变量，如代码1.1所示。

其中 title，author，date 变量也可以通过以下形式来定义：

```
% title
% author(s) (separated by semicolons)
% date
```

查看模板文件，可以获取所有变量（形如`$var$`)。也可以通过修改模板来添加自定义的变量。

1.4 转换命令

pandoc-template 目录加入环境变量后可以直接调用`bookgen.sh`:

```
bookgen.sh init  # 初始化工作环境
bookgen.sh pdf   # 生成pdf电子书
bookgen.sh html  # 生成html电子书
bookgen.sh pdf d # 调试模式，只使用一个代码高亮风格， html电子书也支持调试模式
```

[1] http://www.ruanyifeng.com/blog/2016/07/yaml.html

代码 1.1: code:template-var

```
---
title: 用Markdown+Pandoc+XeLaTeX写作
author:          # 作者（数组）
  - An He
date: \today     # 日期
copyright: true  # 是否生成版权页
lof: true        # 是否生成插图列表页
lot: true        # 是否生成表格列表页
homepage: https://github.com/annProg/pandoc-template
header-includes:
  - \usepackage{cleveref}
# preface用于生成前言
preface: '\LaTeX\ 可以排版格式精美的书籍，但是学习成本较高，使用不便；
 换行请在开头留出一个空格'
---
```

第二章　Pandoc Markdown 语法简介

Pandoc 的目标与原始 Markdown 的最初目标有着方向性的不同。在 Markdown 原本的设计中，HTML 是其主要输出对象；然而 Pandoc 则是针对多种输出格式而设计。因此，虽然 Pandoc 同样也允许直接嵌入 HTML 标签，但并不鼓励这样的作法，取而代之的是 Pandoc 提供了许多非 HTML 的方式，来让使用者输入像是定义列表、表格、数学公式以及脚注等诸如此类的重要文件元素。

Pandoc Markdown 语法介绍可以在 Pandoc 主页[1] 找到。以下翻译大部分部分摘自 tzengyuxiao 的翻译[2]，在此向译者表示感谢。

2.1　段落

一个段落指的是一行以上的文字，跟在一行以上的空白行之后。换行字元会被当作是空白处理，因此你可以依自己喜好排列段落文字。如果你需要强制换行，在行尾放上两个以上的空白字元即可。

Extension: escaped_line_breaks

一个反斜线后跟着一个换行字元，同样也有强制换行的效果。这也是在表格单元格中添加换行的唯一形式。

2.2　标题

有两种不同形式的标题语法，Setext 以及 Atx。Setext 风格的标题是由一行文字底下接着一行 = 符号（用于一阶标题）或-符号（用于二阶标题）所构成；Atx 风格的标题是由一到六个 # 符号以及一行文字所组成，你可以在文字后面加上任意数量的 # 符号。由行首起算的 # 符号数量决定了标题的阶层，如代码2.1所示。

Extension: blank_before_header

原始 markdown 语法在标题之前并不需要预留空白行。Pandoc 则需要（除非标题位于文件最开始的地方）。这是因为以 # 符号开头的情况在一般文字段落中相当常见，这会导致非预期的标题。例如：

```
I like several of their flavors of ice cream:
#22, for example, and #5.
```

[1] http://www.pandoc.org/MANUAL.html#pandocs-markdown

[2] http://pages.tzengyuxio.me/pandoc/

代码 2.1: code:markdownTitle

```
Setext A level-one header
==================

Setext A level-two header
------------------

# Atx level-one

## Atx level-two

### Atx  level-three
```

这也是前一章所述注意事项1.2.3的原因。

2.3 HTML 与 LaTeX 的标题标识符

Extension: header_attributes

在标题文字所在行的行尾，可以使用以下语法为标题加上属性：

```
{#identifier .class .class key=value key=value}
```

虽然这个语法也包含加入类别 (class) 以及键 / 值形式的属性 (attribute)，但目前只有标识符 (identifier/ID) 在输出时有实际作用（且只在部分格式的输出，包括：HTML, LaTeX, ConTeXt, Textile, AsciiDoc）。举例来说，下面是将标题加上 foo 标识符的几种方法：

```
# My header {#foo}

## My header ##    {#foo}

My other header   {#foo}
---------------
```

（此语法与 PHP Markdown Extra 相容。）

具有 unnumbered 类别的标题将不会被编号，即使–number-sections 的选项是开启的。单一连字符号 (-) 等同于.unnumbered，且更适用于非英文文件中。因此，

```
# My header {-}
```

与下面这行是等价的

```
# My header {.unnumbered}
```

2.4 引用

Markdown 使用 email 的习惯来建立引用区块。一个引用区块可以由一或多个段落或其他的区块元素（如列表或标题）组成，并且其行首均是由一个 > 符号加上一个空白作为开头。（> 符号不一定要位在该行最左边，但也不能缩进超过三个空白）。

```
> This is a block quote. This
> paragraph has two lines.
>
> 1. This is a list inside a block quote.
> 2. Second item.
```

效果如下：

> This is a block quote. This paragraph has two lines.
>
> 1. This is a list inside a block quote.
> 2. Second item.

有一个「偷懒」的形式：你只需要在引用区块的第一行行首输入 > 即可，后面的行首可以省略符号：

```
> This is a block quote. This
paragraph has two lines.

> 1. This is a list inside a block quote.
2. Second item.
```

由于区块引用可包含其他区块元素，而区块引用本身也是区块元素，所以，引用是可以嵌套入其他引用的。

```
> This is a block quote.
>
>> A block quote within a block quote.
```

Extension: blank_before_blockquote

原始 markdown 语法在区块引用之前并不需要预留空白行。Pandoc 则需要（除非区块引用位于文件最开始的地方）。这是因为以 > 符号开头的情况在一般文字段落中相当常见（也许由于断行所致），这会导致非预期的格式。因此，除非是指定为 markdown_strict 格式，不然以下的语法在 pandoc 中将不会产生出嵌套区块引用：

```
> This is a block quote.
>> Nested.
```

2.5 代码

缩进式代码块

由四个空格或一个 tab 缩进的文本取做代码块，区块中的特殊字符、空格和换行都会被保留，而缩进的空格和 tab 会在输出中移除，但在代码块中的空行不必缩进。

```
#!/bin/bash

echo "Hello Markdown"
echo "Hello LaTeX"
```

围栏式代码块

Extension: fenced_code_blocks

除了标准的缩进式代码块之外，Pandoc 还支持围栏式代码块，代码块以三个或三个以上的 ~ 符号行开始，以等于或多于开始行 ~ 个数符号行结束，若是代码块中含有 ~，只需使开始行和结束行中的 ~ 符号个数多于代码块中的即可

```
~~~~~
~~~~
code here
~~~~
~~~~~~
```

Extension: backtick_code_blocks

与`fenced_code_blocks`相同，只不过使用反引号 ‘ 替换波浪线 ~ 而已

Extension: fenced_code_attributes

代码 2.2: code:fencedcode

```
100 ~~~~ {#code:mycode .haskell .numberLines startFrom="100"}
101 qsort []     = []
102 qsort (x:xs) = qsort (filter (< x) xs) ++ [x] ++
103                qsort (filter (>= x) xs)
104 ~~~~~~
```

这里的 mycode 为 ID，haskell 与 numberLines 是类别，而 startsFrom 则是值为 100 的属性。numberLines 和 startFrom 配合使用可以显示代码行号，如果没有指定 startFrom，则默认从 1 开始。有些输出格式可以利用这些信息来作语法高亮。目前使用到这些信息的输出格式仅有 HTML 与 LaTeX。如果指定的输出格式及语言类别有语法高亮支持，那么上面那段代码区块将会以高亮并带有行号的方式呈现。

仅指定高亮语言时，可以简写为以下形式：

```
~~~haskell
```

```
qsort [] = []
~~~
```

2.6 行区块

Extension: line_blocks

行区块是一连串以竖线 (|) 加上一个空格所构成的连续行。行与行间的区隔在输出时将会以原样保留，行首的空白字元数目也一样会被保留；反之，这些行将会以 markdown 的格式处理。这个语法在输入诗句或地址时很有帮助。

```
| The limerick packs laughs anatomical
| In space that is quite economical.
|    But the good ones I've seen
|    So seldom are clean
| And the clean ones so seldom are comical

| 200 Main St.
| Berkeley, CA 94718
```

如果有需要的话，书写时也可以将完整一行拆成多行，但后续行必须以空白作为开始。下面范例的前两行在输出时会被视为一整行：

```
| The Right Honorable Most Venerable and Righteous Samuel L.
  Constable, Jr.
| 200 Main St.
| Berkeley, CA 94718
```

效果：

The limerick packs laughs anatomical
In space that is quite economical.
But the good ones I’ve seen
So seldom are clean
And the clean ones so seldom are comical

200 Main St.
Berkeley, CA 94718

这是从 reStructuredText 借来的语法。

2.7 列表

2.7.1 无序列表

无序列表是以项目符号作列举的列表。每条项目都以项目符号 (*, + 或-) 作开头。下面是个简单的例子：

```
* one
* two
* three
```

这会产生一个「紧凑」列表。如果你想要一个「宽松」列表，也就是说以段落格式处理每个项目内的文字内容，那么只要在每个项目间加上空白行即可：

```
* one

* two

* three
```

项目符号不能直接从行首最左边处输入，而必须以一至三个空白字元作缩进。项目符号后必须跟着一个空白字元。

列表项目中的接续行，若与该项目的第一行文字对齐（在项目符号之后），看上去会较为美观：

```
* here is my first
  list item.
* and my second.
```

但 markdown 也允许以下「偷懒」的格式：

```
* here is my first
list item.
* and my second.
```

四个空白规则

一个列表项目可以包含多个段落以及其他区块等级的内容。然而，后续的段落必须接在空白行之后，并且以四个空白或一个 tab 作缩进。因此，如果项目里第一个段落与后面段落对齐的话（也就是项目符号前置入两个空白），看上去会比较整齐美观：

```
  * First paragraph.

    Continued.

  * Second paragraph. With a code block, which must be indented
    eight spaces:

        { code }
```

列表项目也可以包含其他列表。在这情况下前置的空白行是可有可无的。嵌套列表必须以四个空白或一个 tab 作缩进：

```
* fruits
    + apples
```

```
        - macintosh
        - red delicious
    + pears
    + peaches
* vegetables
    + brocolli
    + chard
```

上一节提到，markdown 允许你以「偷懒」的方式书写，项目的接续行可以不和第一行对齐。不过，如果一个列表项目中包含了多个段落或是其他区块元素，那么每个元素的第一行都必须缩进对齐。

```
+ A lazy, lazy, list
item.

+ Another one; this looks
bad but is legal.

    Second paragraph of second
list item.
```

注意：尽管针对接续段落的「四个空白规则」是出自于官方的 markdown syntax guide，但是作为对应参考用的 Markdown.pl 实作版本中并未遵循此一规则。所以当输入时若接续段落的缩进少于四个空白时，pandoc 所输出的结果会与 Markdown.pl 的输出有所出入。

在 markdown syntax guide 中并未明确表示「四个空白规则」是否一体适用于所有位于列表项目里的区块元素上；规范文件中只提及了段落与代码区块。但文件暗示了此规则适用于所有区块等级的内容（包含嵌套列表），并且 pandoc 以此方向进行解读与实作。

2.7.2 有序列表

有序列表与无序列表相类似，唯一的差别在于列表项目是以列举编号作开头，而不是项目符号。

在原始 markdown 中，列举编号是阿拉伯数字后面接着一个句点与空白。数字本身代表的数值会被忽略，因此下面两个列表并无差别：

```
1.  one
2.  two
3.  three
```

上下两个列表的输出是相同的。

```
5.  one
7.  two
1.  three
```

Extension: fancy_lists

与原始 markdown 不同的是，Pandoc 除了使用阿拉伯数字作为有序列表的编号外，也可以使用大写或小写的英文字母，以及罗马数字。列表标记可以用括号包住，也可以单独一个右括号，抑或是句号。

如果列表标记是大写字母接着一个句号，句号后请使用至少两个空白字元。

Extension: startnum

除了列表标记外，Pandoc 也能判读列表的起始编号，这两项资讯都会保留于输出格式中。举例来说，下面的输入可以产生一个从编号 9 开始，以单括号为编号标记的列表，底下还跟着一个小写罗马数字的子列表：

```
 9)  Ninth
10)  Tenth
11)  Eleventh
       i. subone
      ii. subtwo
     iii. subthree
```

当遇到不同形式的列表标记时，Pandoc 会重新开始一个新的列表。所以，以下的输入会产生三份列表：

```
(2) Two
(5) Three
1.  Four
*   Five
```

如果需要预设的有序列表标记符号，可以使用 `#.`：

```
#.  one
#.  two
#.  three
```

2.7.3 定义列表

Extension: definition_lists

Pandoc 支援定义列表，其语法的灵感来自于 PHP Markdown Extra 以及 reStructuredText：

```
Term 1

:   Definition 1

Term 2 with *inline markup*

:   Definition 2

        { some code, part of Definition 2 }

    Third paragraph of definition 2.
```

每个专有名词 (term) 都必须单独存在于一行，后面可以接着一个空白行，也可以省略，但一定要接上一或多笔定义内容。一笔定义需由一个冒号或波浪线作开头，可以接上一或两个空白作为缩进。定义本身的内容主体（包括接在冒号或波浪线后的第一行）应该以四个空白缩进。一个专有名词可以有多个定义，而每个定义可以包含一或多个区块元素（段落、代码区块、列表等），每个区块元素都要缩进四个空白或一个 tab。

如果你在定义内容后面留下空白行（如同上面的范例），那么该段定义会被当作段落处理。在某些输出格式中，这意谓著成对的专有名词与定义内容间会有较大的空白间距。在定义与定义之间，以及定义与下个专有名词间不要留空白行，即可产生一个比较紧凑的定义列表：

```
Term 1
  ~ Definition 1
Term 2
  ~ Definition 2a
  ~ Definition 2b
```

2.7.4 编号范例列表

Extension: example_lists

这个特别的列表标记 @ 可以用来产生连续编号的范例列表。列表中第一个以 @ 标记的项目会被编号为'1'，接着编号为'2'，依此类推，直到文件结束。范例项目的编号不会局限于单一列表中，而是文件中所有以 @ 为标记的项目均会次序递增其编号，直到最后一个。举例如下：

```
(@)  My first example will be numbered (1).
(@)  My second example will be numbered (2).

Explanation of examples.

(@)  My third example will be numbered (3).
```

编号范例可以加上标签，并且在文件的其他地方作参照：

```
(@good)  This is a good example.

As (@good) illustrates, ...
```

标签可以是由任何英文字母、底线或是连字符号所组成的字串。

2.7.5 紧凑与宽松列表

在与列表相关的「边界处理」上，Pandoc 与 Markdown.pl 有着不同的处理结果。考虑如下代码：

```
+   First
+   Second:
    -   Fee
    -   Fie
```

```
    -   Foe

+   Third
```

Pandoc 会将以上列表转换为「紧凑列表」（在 “First”, “Second” 或 “Third” 之中没有

标签），而 markdown 则会在 “Second” 与 “Third”（但不包含 “First”）里面置入 <p> 标签，这是因为 “Third” 之前的空白行而造成的结果。Pandoc 依循着一个简单规则：如果文字后面跟着空白行，那么就会被视为段落。既然 “Second” 后面是跟着一个列表，而非空白行，那么就不会被视为段落了。至于子列表的后面是不是跟着空白行，那就无关紧要了。（注意：即使是设定为 markdown_strict 格式，Pandoc 仍是依以上方式处理列表项目是否为段落的判定。这个处理方式与 markdown 官方语法规范里的描述一致，然而却与 Markdown.pl 的处理不同。）

2.7.6 结束一个列表

如果你在列表之后放入一个缩排的代码区块，会有什么结果？

```
-   item one
-   item two

    { my code block }
```

问题大了！这边 pandoc（其他的 markdown 实作也是如此）会将 { my code block } 视为 item two 这个列表项目的第二个段落来处理，而不会将其视为一个代码区块。

要在 item two 之后「切断」列表，你可以插入一些没有缩排、输出时也不可见的内容，例如 HTML 的注解：

```
-   item one
-   item two

<!-- end of list -->

    { my code block }
```

当你想要两个各自独立的列表，而非一个大且连续的列表时，也可以运用同样的技巧：

```
1.  one
2.  two
3.  three

<!-- -->

1.  uno
2.  dos
3.  tres
```

2.8 分隔线

一行中若包含三个以上的 *, -或 _ 符号（中间可以以空白字元分隔），则会产生一条分隔线：

```
*  *  *  *

---------------
```

2.9 表格

有四种表格的形式可以使用。前三种适用于等宽字型的编辑环境，例如 Courier。第四种则不需要直行的对齐，因此可以在比例字型的环境下使用。

2.9.1 简单表格

Extension: simple_tables,table_captions

简单表格看起来像这样子：

```
  Right     Left     Center     Default
-------     ------ ----------   -------
     12     12        12            12
    123     123       123          123
      1     1          1             1

Table:  Demonstration of simple table syntax.
```

表 2.1: Demonstration of simple table syntax.

Right	Left	Center	Default
12	12	12	12
123	123	123	123
1	1	1	1

表头与资料列分别以一行为单位。直行的对齐则依照表头的文字和其底下虚线的相对位置来决定：

- 如果虚线与表头文字的右侧有切齐，而左侧比表头文字还长，则该直行为靠右对齐。
- 如果虚线与表头文字的左侧有切齐，而右侧比表头文字还长，则该直行为靠左对齐。
- 如果虚线的两侧都比表头文字长，则该直行为置中对齐。
- 如果虚线与表头文字的两侧都有切齐，则会套用预设的对齐方式（在大多数情况下，这将会是靠

左对齐）。

- 表格底下必须接着一个空白行，或是一行虚线后再一个空白行。表格标题为可选的（上面的范例中有出现）。标题需是一个以 Table:（或单纯只有:）开头作为前缀的段落，输出时前缀的这部份会被去除掉。表格标题可以放在表格之前或之后。

表头也可以省略，在省略表头的情况下，表格下方必须加上一行虚线以清楚标明表格的范围。例如：

```
-------     ------ ----------   -------
     12     12        12             12
    123     123       123           123
      1     1          1              1
-------     ------ ----------   -------
```

当省略表头时，直行的对齐会以表格内容的第一行资料列决定。所以，以上面的表格为例，各直行的对齐依序会是靠右、靠左、置中以及靠右对齐。

多行表格 Extension: multiline_tables,table_captions

多行表格允许表头与表格资料格的文字能以复数行呈现（但不支援横跨多栏或纵跨多列的资料格）。以下为范例：

表 2.2: Here's the caption. It, too, may span multiple lines. 看起来很像简单表格，但两者间有以下差别：

Centered Header	Default Aligned	Right Aligned	Left Aligned
First	row	12.0	Example of a row that spans multiple lines.
Second	row	5.0	Here's another one. Note the blank line between rows.

在表头文字之前，必须以一列虚线作为开头（除非有省略表头）。必须以一列虚线作为表格结尾，之后接一个空白行。资料列与资料列之间以空白行隔开。在多行表格中，表格分析器会计算各直行的栏宽，并在输出时尽可能维持各直行在原始文件中的相对比例。因此，要是你觉得某些栏位在输出时不够宽，你可以在 markdown 的原始档中加宽一点。

和简单表格一样，表头在多行表格中也是可以省略的：

表 2.3: Here's a multiline table without headers. 多行表格中可以单只包含一个资料列，但该资料列之后必须接着一个空白行（然后才是标示表格结尾的一行虚线）。如果没有此空白行，此表格将会被解读成简单表格。

First	row	12.0	Example of a row that spans multiple lines.

Second	row	5.0	Here's another one. Note the blank line between rows.

格框表格 Extension: grid_tables,table_captions

格框表格看起来像这样：

: Sample grid table.

+———+———+———————+ | Fruit | Price | Advantages | +===============+===============
| Bananas | $1.34 | - built-in wrapper | | | - bright color | +———+———+———————+ | Oranges | $2.10 | - cures scurvy | | | - tasty | +———+———+———————+ 以 = 串成的一行区分了表头与表格本体，这在没有表头的表格中也是可以省略的。在格框表格中的资料格可以包含任意的区块元素（复数段落、代码区块、清单等等）。不支援对齐，也不支援横跨多栏或纵跨多列的资料格。格框表格可以在 Emacs table mode 下轻松建立。

管线表格 Extension: pipe_tables,table_captions

管线表格看起来像这样：

表 2.4: Demonstration of simple table syntax. 这个语法与 PHP markdown extra 中的表格语法相同。开始与结尾的管线字元是可选的，但各直行间则必须以管线区隔。上面范例中的冒号表明了对齐方式。表头可以省略，但表头下的水平虚线必须保留，因为虚线上定义了资料栏的对齐方式。

Right	Left	Default	Center
12	12	12	12
123	123	123	123
1	1	1	1

因为管线界定了各栏之间的边界，表格的原始码并不需要像上面例子中各栏之间保持直行对齐。所以，底下一样是个完全合法（虽然丑陋）的管线表格：

fruit| price ——|——: apple|2.05 pear|1.37 orange|3.09 管线表格的资料格不能包含如段落、清单之类的区块元素，也不能包含复数行文字。

注意：Pandoc 也可以看得懂以下形式的管线表格，这是由 Emacs 的 orgtbl-mod 所绘制：

One	Two
my	table
is	nice

主要的差别在于以 + 取代了部分的 。其他的 orgtbl 功能并未支援。如果要指定非预设的直行对齐形式，你仍然需要

2.10 插图

2.11 表格

Pandoc 扩展的 Markdown 语法可以为代码块添加 ID 属性及语言类型属性，形如`{#id .language}`，其中 ID 属性可以用来做交叉引用，使用`\ref{id}`在正文中引用代码块。例如代码块**??**。