用R + Markdown进行写作

江航

April 8, 2016

Contents

版权声明

序言

这里整理了一些关于在ubuntu平台下使用markdown语言进行写作，并制作电子书的过程。其中的内容都是来自我的博客文章。整理以后，希望能够作为一份更好的参考。

主要涉及到的知识有：

1. markdown基本语法以及pandoc的一些扩展
2. pandoc的基本用法
3. 如果使用knitr和rmarkdown来动态生成文章内容
4. 如何使用igraph制作流程图（插图）
5. 如何利用markdown文章制作pdf书籍
6. 提供了一个简单的pandoc filter的例子，以后对markdown文章进行处理时可供参考。

我会在使用的过程中，不断完善其中的内容。如果你看了之后，有什么想法，可以写信告诉我 jianghang@bagualu.net。也可以在我的博客上留言。

谢谢关注。

Chapter 1

R + markdown简介

markdown是什么

markdown是由John Gruber提出的一种书写格式，于2004年12月推出它的1.0.1版本。到现在为止这个版本也一直没有更新过，也就是说这是一个基本稳定的版本。作者的博客地址在这里，其中包含了Markdown的语法说明，这个是英文版的。

在说明markdown是什么之前，先看一个markdown文件的实例:

```
# 灵根育孕源流出　心性修持大道生

**诗曰**

>混沌未分天地乱，茫茫渺渺无人见。
自从盘古破鸿蒙，开辟从兹清浊辨。
覆载群生仰至仁，发明万物皆成善。
欲知造化会元功，须看《西游释厄传》。

盖闻天地之数，有十二万九千六百岁为一元。将一元分为十二会，乃子、丑、寅、卯、辰、巳、午、未、申、酉、戌、亥之十二支也。每会该一万八百岁。且就一日而论：子时得阳气而丑则鸡鸣，寅不通光而卯则日出，辰时食后而巳则挨排，
```

这段markdown文档生成的效果如下:

灵根育孕源流出　心性修持大道生

诗曰

> 混沌未分天地乱，茫茫渺渺无人见。 自从盘古破鸿蒙，开辟从兹清浊辨。 覆载群生仰至仁，发明万物皆成善。 欲知造化会元功，须看《西游释厄传》。

盖闻天地之数，有十二万九千六百岁为一元。将一元分为十二会，乃子、丑、寅、卯、辰、巳、午、未、申、酉、戌、亥之十二支也。每会该一万八百岁。且就一日而论：子时得阳气而丑则鸡鸣，寅不通光而卯则日出，辰时食后而巳则挨排，

由此可见，markdown的源码和最终输出的结果比起来，除了格式不一样外，只是多出了一些标点符号。而这些标点符号就是markdown控制格式的方法。

如果说markdown有什么特征，那么markdown最大的特征就是简单。与微软的word相比，不知简单了多少倍，简直不是一个数量级的。标准的markdown中只是定义了10多种格式，并且格式的定义非常容易记忆，以至于你根本不需要化专门的时间来学习它。

那么有人会问，它会不会太简单了？的确有时候它太简单了，比如写作一些科技文献的时候，它提供的基本格式就不够用了。在这种情况下，出现了markdown的几种扩展，可以让人们来方便的处理这些格式。目前比较流行的扩展有pandoc扩展，其次是GitHub扩展。其中GitHub的扩展主要是用于在GutHub的网站上使用。由于GitHub用户很多，所以这个扩展也很流行。而pandoc是一个强大的格式转换工具，基本上支持各类常见格式之间的转换，其中就包括有markdown。而markdown的格式超级简单，这就使得人们能够用超级简单的markdown来生成看起来复杂的其他格式，比如word、html、pdf等。这样，因为pandoc使得人们能够使用markdown来生成doc和pdf文档，因此受到广泛的欢迎。随着pandoc的流行，出现了很多以pandoc为基础的外围工具，以此来方便人们对pandoc的使用，比如在R语言中的几个软件包Rmarkdown，knitr等就使用pandoc作为底层的支持来实现从markdown到各种其他格式的转换。

另外关于这些基于pandoc的扩展，有些已经实现的非常先进，可以帮助作者完成不少以前需要手动完成的工作，比如自动生成文档的部分内容，自动插入图片等等。

基于以上这些原因，本书中也将以pandoc扩展为主要的关注点。

R语言

本书的目的是为了讲解markdown，为什么还要提到R语言？R语言是什么东西，跟markdown有什么关系呢？

如果你使用markdown只是进行文学类创作，比如写写小说什么的，那么你根本不需要使用到markdown的扩展就可以工作了。在这种情况下，你可以不去考虑makrdown扩展以及R的问题。

如果你要使用markdown进行科技文献，科技书籍的写作，那么如果你了解并能够使用R语言，将会对你的工作效率有巨大的提升。

那么R语言是什么？R语言是一种针对统计分析和数据科学的开源的统计语言，它是一种基于向量的高级函数式编程语言。近年来，由于大数据的兴起，越来越多的人开始关注R语言。每天都有大量的志愿者为R软件的发展贡献代码。因此R本身也是一个迅速发展的语言。

由于pandoc的兴起，R社区的开发者发现了这个工具给写作带来的巨大便利，因此开发了一些以pandoc为基础的软件包，其代表是Rmarkdown和knitr。这些软件包让文档或书籍的作者可以方便的从markdown文件动态的生成最终的文档。这些工具使得写作的效率大幅提高。这也是我们将要关注的内容。

rmarkdown和knitr

前面介绍提到了rmarkdown和knitr是R的软件包，它们把能够在markdown中嵌入R语言，并且执行其中的R代码，而R代码的输出可以作为文档的一部分。甚至R代码生成的图片也可以自动嵌入到markdown文件中。

Rmarkdown + knitr可以执行R语言代码的这个功能使得markdown如虎添翼，具有非凡的表现力和极高的效率。含有可执行R代码的markdown文件有一个专门的扩展名，即rmd。

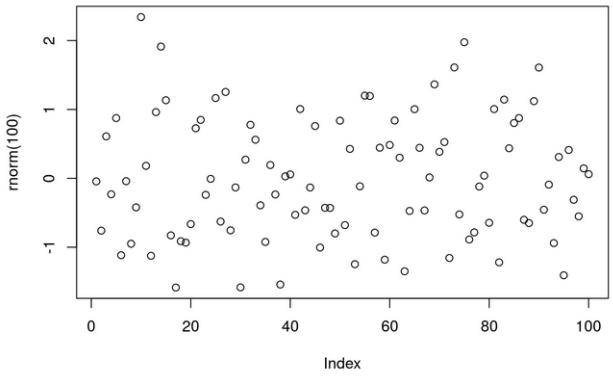
下面我们看一个rmarkdown的实例:

下面的代码将会画出100个正态分布的散点图，画出的图会变成文件的插图:

```
```{.r echo=FALSE,fig.path="./rfigures/p6172-"}
plot(rnorm(100))
```
```

实际生成的内容如下:

下面的代码将会画出100个正态分布的散点图，画出的图会变成文件的插图：



这里可以看到新生成的图片已经被插入文档中了，这样每次重新生成文档，文档的内容都会被更新，这样只要文档中的R代码没有错，该文档将永远保持使用最新的结果，你不需要为每次的代码改动来重新粘贴代码生成的结果。这将为您节省宝贵的时间。

为什么要使用markdown

从上面的介绍可以看出，markdown最大的特征就是简单，其次是基于markdown的各种外围工具使得利用markdown进行创作非常高效。这个高效体现在

1. 语法简单，作者基本上无需关心排版的问题，只要专心写作就可以了。
2. 对于科技文献的写作，R软件提供了文章动态生成的过程。使得作者不必要手动去拷贝粘贴代码的结果或者生成的图片等。
3. 易于修改，因为文档可以从markdown动态的生成。作者只需要修改markdown文件，然后就立刻能够看到最终的文章结果。

还有一个使用markdown的重要原因是，它是**免费**的，markdown只是一种格式定义，你可以使用你喜欢的编辑器来编写markdown文件，编写完的markdown文件可以通过免费的工具来转换成html、pdf、docx等。因此使用markdown你不必要花钱去使用某软的付费系统和付费软件。

怎样使用markdown

markdown文件是通常的文本文件，你使用vi，emacs或者windows下的记事本都可以编写markdown。写好的markdown如果上传到GitHub，那么markdown文件会自动被转换成html显示出来。在本地可以通过perl，R或者其他的工具比如pandoc来将markdown文件转为html或PDF以及docx。

- perl，在John Gruber的网站，提供了一个Markdown.pl，这是标准的markdown解析工具，它可以把markdown转为html

  使用方法是： ./Markdown.pl ./about.md

  这样将会把md文件转换为html，输出在标准输出。

  因为前面提到的，标准的markdown的功能有限，比如没有表格的支持等，所以一般使用中，不推荐直接使用这个perl脚本进行格式转换。

- pandoc，你也可以使用pandoc直接来将md文件转换为html

  使用方法： pandoc -f markdown -t html -o ./about.html ./about.md

  pandoc也可以把md转换为pdf文件，使用方法为 pandoc -f markdown -t latex -o ./about.pdf ./about.md

  pandoc可以支持多种类型的转换，通常你需要熟悉pandoc的各种参数，你可以手动敲这些命令。不过使用R的包会使这个过程简化。

- R ，以下是利用R将当前目录下的about.md文件转换为about.html文档的方法:

  Rscript -e “rmarkdown::render(‘./about.md’)”

  当然，为了上面的代码能够正确运行，你需要在安装R以后，安装rmarkdown这个包，这个包的安装方法是，在R的命令行键入install.packages("rmarkdown")。

  如果要生成pdf格式的文件，你需要在render后面指定更多的参数。这些后面再讨论。

怎样使用rmarkdown

那么rmarkdown的文件怎么编译呢，rmarkdown的文件，也就是.rmd文件就只能在R上编译（转换）了。编译的方法和编译markdown文件一样，只是这时候，文件的扩展名由md变成了rmd，这时候，rmd文件中的r代码就会被处理了，命令行如下:

```
Rscript -e "rmarkdown::render('./about.rmd')"
```

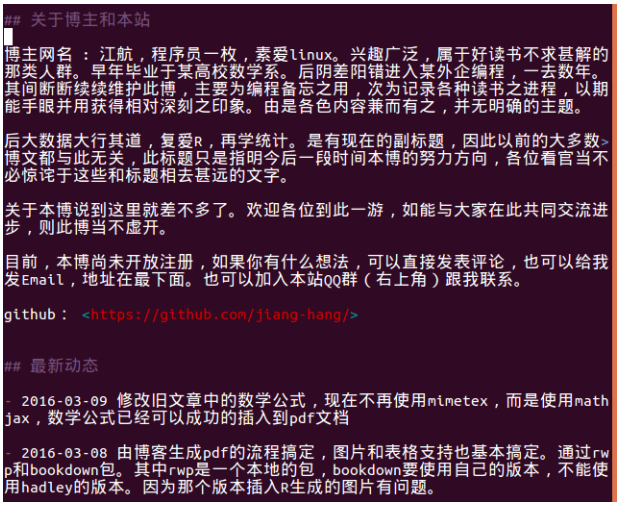
在真正使用markdown进行创作之前，你还需要做一些准备工作。如果你比较着急的想试试，可以直接跳到后面一章去看看markdown的基础语法。

准备工作

在真正使用markdown之前，我们还需要一些准备工作。这里的准备工作主要包括两个部分，一是选择一个编辑器，再就是选择一个编译工具，我们这里选择R语言。当然你也可以选择perl,但是我不推荐。

编辑器

Linux下的同学都应该有自己趁手的工具，vim或者emacs随便找一个就可以了。这些工具下面应该还有些支持markdown语法的插件。对于vim来说，有一款插件叫vim-markdown。这个插件只是提供了语法的高亮显示。如下图所示



在Linux下使用VIM编辑markdown文件实际上是很不方便的，根本原因在于中文的切换问题。你在输入中文的时候，如果要键入一个vim的命令，需要退出中文输入，或者在输入法上输入命令然后直接键入回车键，这是很不方便的。因此，我实际的使用过程中，并不怎么使用vim。而是使用了另一款编辑器，即gedit。这个编辑器中集成了markdown语法的高亮显示。使用起来也比较直接。中文输入的不方便是我这个vim粉放弃vim的根本原因。不过除了markdown以外的其他的编辑基本在vim中进行。

另外现在有一些可以实时看到编辑效果的编辑器，比如Windows下的MarkdownPad，Mac下的Mou，Linux下的ReText等编辑器，大家也可以自己去找一下。选择一个适合自己的编辑器就好了。

pandoc

如前所述，pandoc是一个格式转换工具，可以支持常见文档格式之间的相互转换。它支持很多的格式，使用pandoc --help可以看到它所支持的所有格式，这里列出其支持的格式供参考:

```
xuyang@ubuntu15:~/blog$ pandoc --help
pandoc [OPTIONS] [FILES]
Input formats:  docbook, docx, epub, haddock, html, json, latex, markdown,
                markdown_github, markdown_mmd, markdown_phpextra,
                markdown_strict, mediawiki, native, opml, org, rst, t2t,
                textile, twiki
Output formats: asciidoc, beamer, context, docbook, docx, dokuwiki, dzslides,
                epub, epub3, fb2, haddock, html, html5, icml, json, latex, man,
                markdown, markdown_github, markdown_mmd, markdown_phpextra,
                markdown_strict, mediawiki, native, odt, opendocument, opml,
                org, pdf*, plain, revealjs, rst, rtf, s5, slideous, slidy,
                texinfo, textile
                [*for pdf output, use latex or beamer and -o FILENAME.pdf]
```

另外，pandoc还支持很多变成语言的语法高亮显示，如下所示

```
xuyang@ubuntu15:~/blog$ pandoc --version
pandoc 1.13.2.1
Compiled with texmath 0.8.2.2, highlighting-kate 0.5.12.
Syntax highlighting is supported for the following languages:
    abc, actionscript, ada, agda, apache, asn1, asp, awk, bash, bibtex, boo, c,
    changelog, clojure, cmake, coffee, coldfusion, commonlisp, cpp, cs, css,
    curry, d, diff, djangotemplate, dockerfile, dot, doxygen, doxygenlua, dtd,
    eiffel, email, erlang, fasm, fortran, fsharp, gcc, glsl, gnuassembler, go,
    haskell, haxe, html, idris, ini, isocpp, java, javadoc, javascript, json,
    jsp, julia, latex, lex, lilypond, literatecurry, literatehaskell, lua, m4,
    makefile, mandoc, markdown, mathematica, matlab, maxima, mediawiki,
    metafont, mips, modelines, modula2, modula3, monobasic, nasm, noweb,
    objectivec, objectivecpp, ocaml, octave, opencl, pascal, perl, php, pike,
    postscript, prolog, pure, python, r, relaxng, relaxngcompact, rest, rhtml,
    roff, ruby, rust, scala, scheme, sci, sed, sgml, sql, sqlmysql,
    sqlpostgresql, tcl, tcsh, texinfo, verilog, vhdl, xml, xorg, xslt, xul,
    yacc, yaml, zsh
Default user data directory: /home/xuyang/.pandoc
Copyright (C) 2006-2014 John MacFarlane
Web:  http://johnmacfarlane.net/pandoc
This is free software; see the source for copying conditions.
There is no warranty, not even for merchantability or fitness
for a particular purpose.
```

关于pandoc的安装:

- ubuntu 15.10 sudo apt-get install pandoc
- ubuntu 15.10以前的版本在15.10以前的版本，ubuntu缺省安装的pandoc版本太低，以至于Rmarkdown拒绝调用它。所以在以前的版本上，你需要自己安装pandoc的最新版本，安装方法是:

```
sudo apt-get install cabal-install
cabal update
cabal install pandoc
#记得把这个添加到目录
export PATH=$HOME/.cabal/bin:$PATH
```

使用这个方法安装时，需要花很长的时间，估计要1个小时吧。主要原因是需要下载大量的haskell的包。所以推荐大家在ubuntu 15.10以后的平台上使用。

另外，在用上面的方法安装pandoc的时候，如果使用国内的镜像，速度会快很多，国内镜像的使用方法可以参考清华镜像的使用方法[1]，这里把它记录下来供参考:

第一次使用时，先执行cabal update，待生成~/.cabal/config之后，使用ctrl+c中断这个update，然后修改~/.cabal/config。修改方法为，将下面这行注释掉（用--注释）

remote-repo: hackage.haskell.org:http://hackage.haskell.org/packages/archive

修改为:

remote-repo: mirrors.tuna.tsinghua.edu.cn:http://mirrors.tuna.tsinghua.edu.cn/hackage

然后再执行cabal update和cabal install pandoc。使用国内镜像以后，整个安装过程大约在半小时左右，主要的时间用在编译上。

- 其他的系统暂时还没有尝试

texlive

texlive用来处理Latex，它把latex格式的文件转为pdf。如果需要从markdown生成pdf文件，或者是打算由多篇文章生成一本pdf格式的书籍，需要有texlive的支持。这里选择的是texlive，安装方法如下

```
sudo apt-get install texlive-full
```

这个安装也需要大约1小时左右，因为texlive也有很多的包需要安装。这样安装会多安装许多语言包，不过比起自己去一个一个选择包来说，这是最简单的方式。

另外，因为我们需要中文的支持，所以需要另一个latex引擎，xelatex，所以这个也需要安装，方法是

```
sudo apt-get install xelatex
```

R安装

- ubuntu 15.10

  直接安装就可以了sudo apt-get install r-base rbase-dev

安装成功以后，你应该可以直接键入R命令，如下所示:

```
xuyang@ubuntu15:~/blog$ R

R version 3.2.2 (2015-08-14) -- "Fire Safety"
Copyright (C) 2015 The R Foundation for Statistical Computing
Platform: x86_64-pc-linux-gnu (64-bit)

R是自由软件，不带任何担保。
在某些条件下你可以将其自由散布。
用'license()'或'licence()'来看散布的详细条件。

R是个合作计划，有许多人为之做出了贡献.
用'contributors()'来看合作者的详细情况
用'citation()'会告诉你如何在出版物中正确地引用R或R程序包。

用'demo()'来看一些示范程序，用'help()'来阅读在线帮助文件，或
用'help.start()'通过HTML浏览器来看帮助文件。
用'q()'退出R.

>
```

[1] https://mirrors4.tuna.tsinghua.edu.cn/help/hackage/

R包安装

R安装以后，还需要安装一个R的扩展包才可以工作，其中的rmarkdown,knitr是必须的。安装的方法为，在R的命令行键入

```
install.packages(c('rmarkdown','knitr'),repos="http://mirror.bjtu.edu.cn/cran/")
```

其中repos参数指定包的下载地址，这里指定的是国内的一个R镜像。

准备就绪

有了以上准备工作（pandoc，texlive，R，rmarkdown）以后，你就可以开始r + markdown的工作了。后面开始学习markdown的基础语法。

Chapter 2

markdown语法及pandoc扩展

这里参考markdown官方说明分类介绍markdown的各种语法以及相关的pandoc扩展，其中pandoc的扩展部分，参考内容为pandoc的使用说明。如果是pandoc扩展的部分，会在其中明确指出。在使用pandoc进行转换的过程中，如果想要取消某个扩展，可以使用markdown-<扩展的名字>，比如想要把标题前空行的扩展去掉，可以使用pandoc -f makrdown-blank_before_header -t html -O about.html about.md。另外如果只是想使用某个扩展而其他的扩展都不要，可以使用markdown_strict+<扩展的名字>，比如markdown_strict+blank_before_hader。其中的markdown_strict表示标准的不带任何扩展的markdown语法。

标题

对于标题，只需要在其前面加上#就可以了。#的数量指定了标题的级别。最深为六级。

```
# 1号标题
## 2号标题
### 3号标题
#### 4号标题
##### 5号标题
###### 6号标题
```

另外，对于1号和2号标题，有另外一种语法，代码如下:

```
1号标题
=======

2号标题
-------
```

这个效果和上面的1，2号标题一样。这种风格来自另一个轻量级的标记语言—setext[1]，它的全称是Structure Enhanced Text。因此这种标题被称为setext风格的标题。

pandoc 扩展：标题前空行 blank_before_header

pandoc中，需要在标题的上一行留出一个空白行。因为普通行文中，被#号顶头的可能性很大，强制要求前面留出一个空行，行文时出现#号开头的行就没什么问题了。

pandoc 扩展：标题属性 header_attributes

标题行后面可以使用这样的语法来添加标题属性，这个属性通常为HTML和Latex使用。其语法为

[1] https://en.wikipedia.org/wiki/Setext

```
# 标题{#id .class .class key=value key=value}
```

其中的id是最有用的，通常用于文内的交叉引用。使用id做文内引用的方法是 [标题名](#id)，这样在生成的html和pdf文档中就会有到这个标题的超链了。其中的标题名不必要和该标题的名字一样，只要保证#id一样就可以了。

pandoc扩展：自动生成ID auto_identifiers

在没有指定标题id的情况下，pandoc会自动为每个标题生成一个id，生成id的方法是

- 移除所有格式，链接等。
- 移除所有标点符号（下划线，减号除外）
- 用减号替代所有的空格
- 所有英文字母转为小写
- 移除第一个字符前的所有内容（ID 不能以数字或标点符号开头）。
- 如果剩下为空串，则使用section作为ID

由以上的规则知，如果标题为中文的情况下，基本上就什么都不剩啦。所以在使用pandoc的时候，通常要把这个扩展关掉。

如果指定了--section-divs选项，那么在生成的文档中，每个小节都会用div包住，并用该标题的id作为这个div的id。这样做的好处是，可以使用javascipt来对整个小节进行操作。

pandoc扩展：默认标题引用 implicit_header_references

todo

列表

无序列表

这是代码:

```
* item 1
* item 2
* item 3
```

效果如下:

- item 1
- item 2
- item 3

这里的*可以替换成+和-，效果一样。各个列表项之间可以加入空行。

有序列表

```
1. item 1
1. item 2
1. item 3
```

效果如下:

1. item 1
2. item 2
3. item 3

注意这里和无序列表的区别，前面的*号换成数字，这里数字的具体值并不重要，全部写成1就可以了，编译器会自动给它们编号的。

嵌套列表

无需和有序列表、无序列表之间以及有序列表之间都可以嵌套，如下述代码

```
* item 1
    1. sub item 1
    1. sub item 2
    1. sub item 3
* item 2
* item 3
```

效果如下:

- item 1
    1. sub item 1
    2. sub item 2
    3. sub item 3
- item 2
- item 3

注意这里的嵌套列表，需要缩进**四个**空格

多段列表

有时候，每个列表项都需要几段话来说明，这这种情况下，后面的段落需要有四个空格的缩进，段内需要换行时使用两个空格，如下所示

```
1. 这是item1 ， 假设我们有很长的很长的很长的很长的
    很长的很长的很长的很长的很长的很长的很长的说明

    这是说明的第二段，很长的很长的很长的很长的很长的
    很长的很长的很长的很长的很长的很长的说明

    这是说明的第三段，很长的很长的很长的很长的很长的
    很长的很长的很长的很长的很长的很长的说明

2. 这是项目2，它的说明不怎么长
```

效果如下:

1. 这是item1 ， 假设我们有很长的很长的很长的很长的
   很长的很长的很长的很长的很长的很长的很长的说明

   这是说明的第二段，很长的很长的很长的很长的很长的
   很长的很长的很长的很长的很长的很长的说明

   这是说明的第三段，很长的很长的很长的很长的很长的
   很长的很长的很长的很长的很长的很长的说明

2. 这是项目2，它的说明不怎么长

段落和换行符

段落之间有一个空行。如果没有空行，那么它们会被认为是一个段落。同一个段落中不同的行之间的换行符会被一个空格代替。这个对于中文而言，造成的后果是，段落中会莫名其妙的多出几个空格，看着奇怪。pandoc中有一个扩展能自动处理这种换行符，这个扩展叫east_asian_line_breaks，这个扩展在pandoc中没有被缺省的打开，所以需要手动打开，即使用-from markdown+east_asian_line_breaks，这个扩展是在pandoc的1.16版本中实现，此前的版本中是没有这个扩展的。因此，如果你使用这个扩展是发现这个扩展不认识，那你需要升级你的pandoc的版本。

一行结束时，如果要强制换行，可以在行尾输入两个及以上空格。不然，pandoc会以上面的方式进行段内换行的处理。

引用

使用>来表示引用

如

```
> 这是引用
> 这还是引用
```

效果如下:

> 这是引用这还是引用

在实际使用中，你可以只在第一行写上>，后面的行可以省略。直到碰到空行，这个引用会一直有效。引用可以嵌套，即引用中还可以有引用，使用方法，如下所示

```
> 你好，这是第一层引用

> > 这是第二层引用，很长的引用
这是第二层引用，很长的引用
这是第二层引用，很长的引用
```

以上代码产生的效果如下:

> 你好，这是第一层引用
>
> > 这是第二层引用，很长的引用这是第二层引用，很长的引用这是第二层引用，很长的引用

注意这里的第二层引用和第一层引用之间的空行。pandoc中有一个扩展叫blank_before_blockquote，也就是在区块引言之前要有一个空行。否则不会被认为是区块引言。

加粗、斜体及删除线以及intraword_underscores扩展

代码:

```
**加粗**的语法
第二种__加粗__的方法
这个是*斜体*
第二种_斜体_
这里是~~删除线~~
```

效果如下:

加粗的语法
第二种__加粗__的方法
这个是*斜体*
第二种_斜体_
这里是~~删除线~~

有上面的生成效果看出，其中使用*作为加粗和斜体的生效了，但是使用_作为加粗的却没有效果，原因在于pandoc的另一个扩展intraword_underscores，这个扩展的意思是pandoc不会把文字中间的_解释为加粗的语法，因为pandoc认为在文字中使用_是一个普遍的现象。如果解释为加粗反而会带来不便，因此如果要在文字中间使用强调（加粗）语法，请使用*。

链接

行内链接

代码:

```
[猎数博客](http://www.bagualu.net/wordpress)
```

效果如下:

猎数博客

链接的url后面可以跟title，格式为[文字](url title)，url和title之间有一个空格，title需要用引号或者小括号包起来如下所示:

代码:

```
[猎数博客](http://www.bagualu.net/wordpress "关注科技，数据挖掘")
```

效果:

猎数博客

参考链接

在参考链接中，链接和内容可以分开，这样有多个同样的链接时，只需要写一次，如下所示，其中链接的内容可以单独写在文章的下面。

```
- [文章1][1]
- [文章2][2]
- [文章1][1]

[1]: http://www.bagualu.net/wordpress/archives/5284
[2]: http://www.bagualu.net/wordpress/archives/5336
```

这段代码产生的效果如下:

- 文章1
- 文章2
- 文章1

其中第二个括号的数字可以为文字，这是用作区别各个参考链接的标识。比如[文章1][artical1]，第二个括号中的标签不区分大小写。

如果第二个放括号为空或者没有第二个方括号，也可以做一个参考链接，此时，第一个中括号的内容就时标签内容，如[my web]，在后面的参考链接部分只要写上[my web]: http://xxx.com就可以了。这是一个隐式的参考链接。

自动链接

在不指定链接文字的情况下，可以用这个语法直接生成一个链接

```
<http://www.bagualu.net/wordpress>
```

生成的链接效果为:

http://www.bagualu.net/wordpress

图片

代码:

```
![猎数博客](http://www.bagualu.net/wordpress/wp-content/themes/xuyang2/images/logo2.png)
```

效果如下:

和前面的链接一样，可以使用一个单独的中括号，然后在后面给出链接。也可以在链接后给出图片的alt。如

```
![猎数博客](url "alt")
```

或者`![猎数博客]`， 然后在最后面给出`[猎数博客]: http://www.bagualu.net "科技博客"`这样的图片链接。

水平线

代码:

```
---
***
```

效果如下:

注意，markdown中还有一种语法结构叫YAML元数据块，这种块以---开头，以---或...结尾，因此，在使用多条水平线的时候，有可能与这个块混淆，因此，建议在使用水平线的地方，使用多余三个的-，这样可避免这种混淆。

定义

pandoc定义扩展definition_lists

可以用下面的方式来给出一个定义

```
词语1

:    词语1的定义

    词语1的第二段定义
第二段定义是多行的

词语2

:    词语2的定义
```

这段代码产生的效果如下:

词语1 词语1的定义

词语1的第二段定义第二段定义是多行的

词语2 词语2的定义

其中词语所在行必须独占一行。词语后面的空行是可选的。

定义部分以:或者~开头，前面可以有一个或者两个空格。

一个词语可以有多个定义。每个定义可以有一个或多个块，其中可以包含代码，列表等。注意这里的**每个块**必须有四个空格的缩进（包括第一个块，四个空格从:或者～开始计算，不包括这两个标点符号前面的空格，也就是这两个符号之后要有四个空格，然后才是定义的内容），不然不会被认为是一个有效的定义列表。

pandoc实例扩展example_lists

实例扩展使用@开始，如下所示

```
(@)  第一个实例，编号为1。
(@good)  第二个实例，编号为2。
关于上面这个实例的更多解释
(@)  第三个实例，编号为3。

可以像这样引用前面定义的实例，如请参考(@good)
```

生成的效果如下:

---

(1) 第一个实例，编号为1。
(2) 第二个实例，编号为2。关于上面这个实例的更多解释
(3) 第三个实例，编号为3。

可以像这样引用前面定义的实例，如请参考(2)

代码

缩进代码块

一个以四个空格字符开头的段落被认为是代码，其中的字符不会被特殊处理，而是会被原样输出。注意其中的空行不需要使用四个空格。

行内代码

行内的代码使用`。 如`printf("hello world")`。

pandoc扩展： 栅栏代码块 fenced_code_blocks

大段的代码可使用```或者~~~包起来，如

```
```c
int main() {
    return 0;
}
```
```

如果在代码中使用到了``` ，可以在开头栅栏中使用更多的`，比如四个，然后在结束的地方使用同样多的`就可以了。

pandoc 扩展： 行块 line_blocks

行块是以|开头的连续行，注意这里|后面的空格。行与行间的距离在输出时会原样保留，每行前的空白也会保留。如下所示

```
| 你好，第一行
|   你好，这是第二行
|       第三行
```

你好，第一行
 你好，这是第二行
   第三行

这种语法有效的保留了行前的空格，在有些情况下可以对格式有很好的控制。

上下标扩展superscribe和subscript

上下标通过~和 ^ 来指定，这两个符号中间的内容为下标和上标，如下所示

```
H~2~0

2^10^ = 1024
```

效果如下:

H_2O

2^{10} = 1024

多段列表和缩进代码块

前面看到，多段列表后面的段落和缩进代码块都是使用四个空格来说明。那么在多段列表中要使用代码块应该怎么做呢?
假定有下面的多段列表

```
1. 这是一个多段列表的第一段

    这是一个多段列表的第二段，第二段继续，第二段
第二段，第二段。

    这是一个多段列表中的代码段，使用栅栏代码块

    ```c
    printf("hello world")
    ```

    不使用栅栏代码块的情况

    printf("hello world");

1. 列表的第二项，使用栅栏代码块，但是不缩进的情况

```c
printf("hello world");
```

1. 多段列表的第三项，跟在一个没有缩进的栅栏代码块后面，它的标号会从1开始。

    多段列表第三项的第二段
```

这段代码的产生的效果如下:

1. 这是一个多段列表的第一段

这是一个多段列表的第二段，第二段继续，第二段第二段，第二段。

这是一个多段列表中的代码段，使用栅栏代码块

```
printf("hello world")
```

不使用栅栏代码块的情况

printf(“hello world”);

2. 列表的第二项，使用栅栏代码块，但是不缩进的情况

```
printf("hello world");
```

1. 多段列表的第三项，跟在一个没有缩进的栅栏代码块后面，它的标号会从1开始。

   多段列表第三项的第二段

由上面的示例知，在多段列表中，要使用代码块，需要四个空格缩进，同时使用栅栏代码块，单独的缩进不会被认为是代码。另外，如果在使用栅栏代码块时不进行四个空格的缩进，会被认为是一个列表的结束，后面的列表项目不会继续编号，而是会从1开始重新编号。

表格

标准的markdown中没有表格的支持，关于表格的内容都属于是pandoc的扩展

pandoc表名扩展table_captions

这个扩展可以在表前或者表后使用一个独立的段来给出表格的名字，该段以Table:或者:开头。如后面所示

简单表格扩展simple_tables

简单表格的代码是这样的:

```
  右对齐    左对齐      中间对齐      缺省
--------  -------  ----------- --------
       1        2           2        3
   3            2           3        3

  Table:  简单的表格
```

这个代码产生的表格为:

Table 2.1: 简单的表格

右对齐	左对齐	中间对齐	缺省
1	2	2	3
3	2	3	3

表格的对齐方式为，表头与第二行虚线之间的对齐关系。表格下面必须有一个空白行。其中表头可以省略，此时必须在表格结束的地方添加一行虚线。如下所示

```
--------  -------  ----------- --------
       1        2           2        3
   3            2           3        3
--------  -------  ----------- --------

  Table:  简单的表格
```

Table 2.2: 简单的表格

1	2	2	3
3	2	3	3

多行表格扩展multiline_tabes

和上面的简单表格一样，多行表格只是让每个单元格或表头可以跨越多行。而表格的每行之间要有一个空行，另外，表头和表尾的虚线行不能省，如下所示

```
-------------------------------------------
  header     heander       多行的
    1          2            内容
---------- ----------  --------------------
  单元格1    单元格2       多行的
                          单元格

  单元格3    单元格4       单元格5
-------------------------------------------

: 多行的表格
```

其产生的表格如下:

Table 2.3: 多行的表格

header 1	heander 2	多行的内容
单元格1	单元格2	多行的单元格
单元格3	单元格4	单元格5

管道表格扩展pipe_tabes

管道表格可显示的指定各列的对齐方式，但是管道表格的单元格不可以是多行的，不能包含块级元素。下面是一个例子

```
| Right | Left | Default | Center |
|------:|:-----|---------|:------:|
|   12  |  12  |    12   |    12  |
|  123  |  123 |   123   |   123  |
|    1  |    1 |     1   |     1  |
: 管道表格，对齐方式通过上面虚线中的`:`指定
```

它产生的表格如下:

Table 2.4: 管道表格，对齐方式通过上面虚线中的:指定

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

表格中的虚线行和Setext风格的标题

由上面的介绍知，在表格中会使用到虚线，而虚线如果放在一句文字下面也是二号标题的Setext风格表示，因此这里需要把这两种情况区分开。可行区分的方法是，表头的虚线行上方留出一个空行，这样就不会被识别为标题行。

对于简单的表格，表头下方的虚线是分为多段的，而标题行下的虚线是一条没有断开的虚线行，因此这种情况下，二者也可以被区分开。

元数据块

这一小部分内容可以在后面学习过pandoc模板以后在来看。

在markdown中可以嵌入元数据块，这些块用来给定文档的一些属性，这些块分为两种，一种是使用在文章最前面的标题块，还有一种是使用在文中任意地方的，叫YAML块，分别介绍如下:

pandoc标题块扩展 pandoc_title_block

这个块放在文件的最前面，格式如下:

```
% 标题
% 作者，多个作者使用分号分开
% 日期
```

这样的内容将会被解析，解析的结果可能会出现在最后的文档中（比如在pdf或者html的标题以及作者部分，之所以说“可能”，是因为自定义模板中可能没有这些输出）。这三个元素也可以只有其中一个或者两个，在这种情况下，在相应的地方要留出空行，比如

```
% 我的标题
%
% 2016年4月6日
```

标题以及作者可以是多行的，第二行以后的文字需要缩进。多个作者可使用分号分开，也可使用多行格式，例如:

```
% 我的标题
  多行的标题
% 作者1
  作者2
% 2016年4月6日
```

日期必须在一行，不能分行。

关于这些元素在最后输出文件中的位置，可以参考后面的pandoc模板部分。

pandoc YAML元数据扩展yaml_metadata_block

yaml块是用---开头，以---或者...结束的一个块，这中块可以出现在文中的任何地方，每个块开始的地方，必须在前面留出一个空行。yaml块也可以放在一个独立的文件中，在使用pandoc的时候，把这个yaml文件作为输入就可以了。pandoc在处理多个输入文件的时候，总是把这些文件合成一个文件，然后再进行处理，因此一个单独的yaml文件是不会有问题的。但是请注意一定要以---把yaml块包起来。

下面是一个处理独立yaml文件的命令行

```
pandoc aa.md metadata.yaml -s -o aa.html
```

yaml块中定义的变量会加入到文件的metadata中。如果一个变量被多个块定义，那么它的值为**第一个**被定义的地方，后面的定义无效。**另外如果一个变量以下划线结束，它将不会被pandoc处理**。

下面是一个元数据块的例子，第一行中包含:，因此需要一个引号包起来，在abstract定义中，使用|来开始一个缩进的块。

```
---
title:  'This is the title: it contains a colon'
author:
- name: Author One
  affiliation: University of Somewhere
- name: Author Two
  affiliation: University of Nowhere
tags: [nothing, nothingness]
abstract: |
  This is the abstract.

  It consists of two paragraphs.
...
```

在上面的这个块中，作者部分有自定义的域，因此需要一个自定义的模板来显示它们，下面是一个模板的例子:

```
$for(author)$
$if(author.name)$
$author.name$$if(author.affiliation)$ ($author.affiliation$)$endif$
$else$
$author$
$endif$
$endfor$
```

数学公式

标准的markdown中没有数学公式的支持,不过可以通过MathJax的支持来在页面上显示数学公式，比如下面的例子:

```
$x^2 + y^2 = z^2$
```

效果如下:

$x^2 + y^2 = z^2$

要使用，可在页面中添加下面的代码:

```
<script type="text/x-mathjax-config">
  MathJax.Hub.Config({
    extensions: ["tex2jax.js"],
    jax: ["input/TeX","output/HTML-CSS"],
    tex2jax: {inlineMath: [["$","$"],["\\(","\\)"]]}
  });
</script>
<script type="text/javascript" src="/MathJax/MathJax.js"></script>
```

你需要在你网站的根目录下有MathJax的包。

对于pdf而言，不需要使用javascript，直接使用pandoc生成就可以了，比html要方便。

嵌入html代码

在markdown文件中可以嵌入html代码，对于嵌入的html代码有两种情况

段级html代码嵌入及pandoc扩展markdown_in_html_blocks

嵌入的html代码段必须在头上和结尾出与markdown的其他部分用一个空行分开，并且起始tag和结束tag行前不要有空格或者tab。其次，在标准的markdown语法中，html代码段中间的markdown语法不会被处理。比如你用html代码段插入一个表格

```
<table>
<tr><td>[博客](http://www.bagualu.net)</td><td>**col2**</td></tr>
</table>
```

其中的链接和**不会被标准的markdown处理。而在pandoc中有一个扩展叫markdown_in_html_blocks ， 打开这个扩展，可以使html代码块中markdown被解析。在pandoc中，上面的代码会生成链接和强调。

<table>
<tr><td>博客</td><td>col2</td></tr>
</table>

由此可见html代码段中间的markdown语法已经被处理。

行级的html代码嵌入

除了在段级的html代码，在行级嵌入html代码，可以直接使用标签就可以了。行级标签下的markdown语法会被处理。

pandoc扩展raw_html

严格说这不算是一个扩展，因为标准markdown中就有这个支持，这里做成一个扩展是为了可以方便的关掉它。对于嵌入的html格式，当输出为latex时，其中的html代码会被直接丢掉。由此可见，要在markdown中插入表格是一个比较麻烦的事情，因为不同格式下，其代码是不一样的。

pandoc扩展raw_tex

除了在markdown中嵌入html格式外，也可以在其中嵌入latex的代码。但是对于latex格式而言，其中的markdown格式是肯定不会被解释的。也就是说没有一个对应的markdown_in_tex_blocks的扩展来解释tex中的markdown语法。

pandoc引用扩展citation

这是关于如何添加参考文献的内容。

pandoc脚注扩展footnotes

pandoc可以使用下面的方式来产生脚注

```
这是一个脚注的例子[^1]，第二个脚注的例子[^note2]

[^1]: 脚注的内容，这个内容不必要在文章最后，只要不包含在其他的块级元素中就可以了
[^note2]: 脚注的内容，更多内容更多内容更多内容
    第二行以后的脚注内容需要缩进
    更多内容更多内容更多内容

    更多内容更多内容

脚注结束的地方，不缩进就可以了
```

查看其实际产生的结果

这是一个脚注的例子[2]，第二个脚注的例子[3]

脚注结束的地方，不缩进就可以了

[2]脚注的内容

[3]脚注的内容，更多内容更多内容更多内容第二行以后的脚注内容需要缩进更多内容更多内容更多内容
更多内容更多内容

YAML语法简单介绍

前面提到了markdown中的元数据块可以通过YAML语法来指定。

YAML全称为（Yet Another Markup Language），也是一种标签语言。是以数据为表达目标的语言。通过空格和分行来分隔数据单元。下面是一个实例:

```
house:
  family:
    name: Doe
    parents:
      - John
      - Jane
    children:
      - Paul
      - Mark
      - Simone
  address:
    number: 34
    street: Main Street
    city: Nowheretown
    zipcode: 12345
```

其中连续的项目通过-来表示，map结构的key/value结构使用:来分隔。要求同一级别的数据前面缩进的空格数要相同。

在上面的例子中，Simone可使用hourse.family.children[3]来引用。

其他的相关内容:

- 要添加注释，使用#符号。
- 布尔值使用 true 和 false
- 空值使用 null 或者 ~
- 数值支持
    1. 整数 ，如12
    2. 八进制数， 如012
    3. 16进制数， 如0xC
    4. 浮点数 ， 如1.24
    5. 指数 ， 如1.2e3
    6. 无穷: .inf

结束前，我们看这样一个例子:

```
title:
 - type: main
   text: My Book
 - type: subtitle
   text: An investigation of metadata
```

在这个例子中，要访问“My Book”,应该使用title[1].text。这里的两个-表示title下是一个数组结构。

Chapter 3

用pandoc把markdown转换为html

markdown 到 html 的简单转换

第一个转换

markdown到html的转换是我们目前最感兴趣的，这里列出一些常用的方法。假设有以下的markdown文档abc.md，　　内容如下:

```
    # 标题1

    ## 子标题1

    你好，这是一个测试
```

使用命令pandoc -f markdown -t html -o abc1.html abc.md，生成的html文档abc1.html为:

```
<h1 id="标题1">标题1</h1>
<h2 id="子标题1">子标题1</h2>
<p>你好，这是一个测试</p>
```

如果想要一个完整的文档，可以添加-s参数。即pandoc -f markdown -t html -o abc2.html -s abc.md，生成的文档为:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
  <meta http-equiv="Content-Style-Type" content="text/css" />
  <meta name="generator" content="pandoc" />
  <title></title>
  <style type="text/css">code{white-space: pre;}</style>
</head>
<body>
<h1 id="标题1">标题1</h1>
<h2 id="子标题1">子标题1</h2>
<p>你好，这是一个测试</p>
</body>
</html>
```

在生成的HTML中包含其他文件

如果要在markdown文件中包含其他的文件，可以使用pandoc的参数下面三个参数:

```
  -H FILENAME           --include-in-header=FILENAME
  -B FILENAME           --include-before-body=FILENAME
  -A FILENAME           --include-after-body=FILENAME
```

这三个参数的意义显而易见，就是在头上，或者在文档最前面、最后面包含一部分内容。假设有一个头文件header.html为这个

```
<title>这是标题</title>
```

那么,pandoc -f markdown -t html -B ./header.html -o abc3.html -s abc.md生成的文件为:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
  <meta http-equiv="Content-Style-Type" content="text/css" />
  <meta name="generator" content="pandoc" />
  <title></title>
  <style type="text/css">code{white-space: pre;}</style>
</head>
<body>
<title>这是标题</title>
<h1 id="标题1">标题1</h1>
<h2 id="子标题1">子标题1</h2>
<p>你好，这是一个测试</p>
</body>
</html>
```

这里可以看见header.html中的内容被加入了。如果这里插入markdown格式的内容为是不会被处理的，它会原样放进去。

生成目录

pandoc可以自动文文章生成目录，方法是将各个级别的标题抽取出来，作成一个列表。

pandoc中，这个叫toc，全称为table of contents，要生成toc，可以在pandoc的命令行指定参数 --toc或者--table-of-contents， 缺省情况下，目录的深度是3级，即--toc-depth 3， 你可以利用这个参数指定生成目录的深度。对于html而言，生成的目录链接会指向各个标题的id。如果各个标题的id都被正确生成了，那么目录应该是没有问题的。如果你发现有些标题没有生成正确的id或者是生成的链接有问题，请检查一下关于id生成的问题。

html中的标题属性

前面在markdown语法的pandoc扩展中有提到一个标题属性扩展，标题中的id属性会用来生成html标题的id，有时候你会碰到这种情况，即不是每个标题都生成了id，以至于生成目录(toc)中的有些链接是无效的。原因是pandoc的另一个扩展ascii_identifiers。如果这个扩展被开启，那么id会通过对标题进行处理以后得到的，处理的方法是留下ASCII字符，空格之类的符号用-代替。如果某个标题的内容是中文，那么就有可能没有id了，因为pandoc各种去掉之后，就什么都不剩了。这样导致生成的目录链接有问题，就是纯中文的标题会没有链接。

解决这个问题的方法是要么使ascii_identifiers扩展无效，要么就是在标题后面指定一个id，方法如下:

```
## 这是一个标题 {#title1}

### 这是下一级标题 {#subtitle1}
```

在标题后面的花括号中，指定一个id，生成的html中，该标题就会使用给定的id。而不会再去作处理。上面的markdown生成的html为:

```
<h2 id="title1">这是一个标题</h2>
<h3 id="subtitle1">这是下一级标题</h3>
```

如果不想手动来设置每个标题的id，你可以选择将ascii_identifiers这个扩展去掉。

交叉引用

前面提到了生成目录的方法，其中目录就是根据id来指向某个标题。在目录中是自动生成的。同样的，在文中，我们也可以手动的来建立这些链接，这样就生成了文中的内链。使用的方式是[标题名](#id)，这样在生成的html中，就会生成相关的链接。以上面的两个标题为例，假设我想要引用上面的二级标题，可以在文中写请参考[下一级标题](#subtitle)。

pandoc 模板

在了解了pandoc把markdown转换为html的基本过程之后，可以更加详细地来看看一个完整的html文档是如何生成的。

pandoc使用一个html模板来生成一个html页面，我们可以使用命令pandoc -D html来查看缺省的模板，它看起来像这样:

```
pandoc -D html
#> <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dt
#> <html xmlns="http://www.w3.org/1999/xhtml"$if(lang)$ lang="$lang$" xml:lang="$lang$"$endif$$if(dir)$ dir="$dir$"$endif
#> <head>
#>   <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
#>   <meta http-equiv="Content-Style-Type" content="text/css" />
#>   <meta name="generator" content="pandoc" />
#> $for(author-meta)$
#>   <meta name="author" content="$author-meta$" />
#> $endfor$
#> $if(date-meta)$
#>   <meta name="date" content="$date-meta$" />
#> $endif$
#> $if(keywords)$
#>   <meta name="keywords" content="$for(keywords)$$keywords$$sep$, $endfor$" />
#> $endif$
#>   <title>$if(title-prefix)$$title-prefix$ – $endif$$pagetitle$</title>
#>   <style type="text/css">code{white-space: pre;}</style>
#> $if(quotes)$
#>   <style type="text/css">q { quotes: "“" "”" "‘" "’"; }</style>
#> $endif$
#> $if(highlighting-css)$
#>   <style type="text/css">
#> $highlighting-css$
#>   </style>
#> $endif$
#> $for(css)$
#>   <link rel="stylesheet" href="$css$" type="text/css" />
#> $endfor$
#> $if(math)$
#>   $math$
#> $endif$
#> $for(header-includes)$
#>   $header-includes$
#> $endfor$
#> </head>
#> <body>
#> $for(include-before)$
#> $include-before$
#> $endfor$
```

```
#> $if(title)$
#> <div id="$idprefix$header">
#> <h1 class="title">$title$</h1>
#> $if(subtitle)$
#> <h1 class="subtitle">$subtitle$</h1>
#> $endif$
#> $for(author)$
#> <h2 class="author">$author$</h2>
#> $endfor$
#> $if(date)$
#> <h3 class="date">$date$</h3>
#> $endif$
#> </div>
#> $endif$
#> $if(toc)$
#> <div id="$idprefix$TOC">
#> $toc$
#> </div>
#> $endif$
#> $body$
#> $for(include-after)$
#> $include-after$
#> $endfor$
#> </body>
#> </html>
```

在看了这个文档之后，你会发现其中有些html标签，还有一些特殊的标签和代码。下面来看看pandoc是如何解释这些特殊标签和代码的。

在进行模板处理的时候，pandoc中会自动生成一些变量，还有一些变量是用户用各种方式指定的。在模板中要引用一个变量使用$变量名$,模板中可以使用条件判断，形式如下:

```
$if(title)$
  <title>$title$</title>
$else$
  <title>缺省的标题</title>
$endif$
```

上面的意思是，如果定义了title变量，那么就输出title的内容，否则就输出缺省的标题。

另外有些变量中含有多个内容，比如需要包含的内容，此时可以使用for循环来打印其中的内容，如下所示

```
$for(header-includes)$
$header-includes$
$endfor$
```

另外，如果要在模板中打印$，使用$$就可以了。

有了这些知识，基本上就可以看明白模板中的内容了。基于此，我们也可以定义自己的模板。自定义模板写好之后，可以使用--template=FILENAME来指定模板文件。

pandoc模板中可以使用的变量

前面知道了如何在模板中使用变量，那么在pandoc中到底有哪些变量可以使用和如何设置呢？对于不同的输出文档而言，使用pandoc -D <format> 可以输出该格式的模板。其中可以看到模板中使用了哪些变量。对于其中使用的变量一般可以使用YAML元数据块来设定。也可以在命令行中使用-M参数来指定。

其中有一个重要的变量body，它是整个markdown的内容。这个是无法通过YAML或者命令行来设置的。

参考文档

- pandoc src on github
- pandoc README.pdf
- pandoc.org (打开慢!!)
- rmarkdown

Chapter 4

用pandoc把markdown转化为pdf文档

本文的目的是由markdown生成pdf格式的文件。其基本转换流程如下:

可以看到这里需要Latex的支持。Latex中需要设置中文的字体。不然，生成的pdf中，中文将无法显示。

这里先简单的介绍下Latex以及其中文支持。

Latex环境

- ubuntu 15.10

```
xuyang@ubuntu15:~/blog$ uname -a
Linux ubuntu15 4.2.0-16-generic #19-Ubuntu SMP Thu Oct 8 15:35:06
   UTC 2015 x86_64 x86_64 x86_64 GNU/Linux
```

  安装的时候，选择语言为简体中文。该系统的安装过程详见ubuntu 15.10安装过程及其上的软件包

- Latex

  Latex安装的是texlive-full包，这是个比较暴力的方法。该包的安装需要较长的时间（估计1个小时左右）

第一个Latex文档

在进行Latex介绍之前，先看一个实例。这是一个包含中文的最简单的latex文件:

```
\documentclass[12pt]{article}
\usepackage{CJK}
\begin{document}
\begin{CJK}{UTF8}{gbsn}
你好世界
\end{CJK}
\begin{CJK}{UTF8}{gkai}
你好世界
\end{CJK}
\end{document}
```

可以使用latex aa.tex ； dvipdfm aa.dvi ；生成aa.pdf文件。效果如图所示:

Latex格式简介

基于上面的这一份简单文档，我们非常简单的介绍一下Latex格式。对Latex格式有一些基本的了解以后，就可以方便的去修改pandoc的模板，以生成自己想要的效果。

Table 4.1: texdoc模式选项

模式	选项	长格式选项
view	-w	view
list	-m	mix
mix	-l	list
showall	-s	showall

参考文档为一份不太简短的LATEX2 ε 介绍

源文件的结构

- 每个文档必须以\documentclass[options]{class}开始，指定文档类，比如book,article等

  比如: \documentclass[11pt,twoside,a4paper]{article}

- 开始语句后，可以引入各种宏包，用以控制各种排版格式，引入宏包的格式为\usepackage[options]{package}。宏包引入以后，就可以在后面的正文中使用包中定义的宏来控制文档的内容和格式。

  有很多宏包是和Latex一起发布的，要查询给定宏包的说明文档，可以使用texdoc命令。比如texdoc fontenc会打开fontenc的文档。关于texdoc的使用方法可以使用texdoc texdoc看到。

- 页面样式\pagestyle{style} ， style值为plain,headings,empty三者之一，其中plain在页脚显示页码;headings在页眉中显示章节名及页码，页脚空白;empty页眉和页脚空白。

- 文档开始\begin{document}

- 文档内容， 其中可以包含其他tex文档的内容，这个在大型项目（比如一本书）中很有用，你可以把每一章做成一个tex文件，然后在书的tex文件中包含各个章节的内容。语法是\include{filename}

- 文档结束\end{document}

使用texdoc获得帮助

上面的介绍算是最简单的介绍，在实际的使用过程中，这是远远不够的，因此我们需要学会怎样去查阅tex的文档。

- 基本命令使用texdoc的基本命令是texdoc <package-name>，这个命令将打开包<package-name>的文档。这个package-name 可以指定多个。

- 模式

  texdoc提供多种模式，你可以选择一种模式来浏览latex的文档。

  - view模式，显示指定包的文档
  - list模式，列出相关的文档供你选择，然后你选择一个需要文档打开
  - mix模式，混合上面的两种模式，如果只有一个相关文档，那就打开这个文档，否则就会列出相关的文档
  - showall模式，列出比list模式更多的相关文档。

  要使用给定的模式来浏览文档，使用命令行texdoc [mode] <name>

  其中，模式参数为

举例来说，你想要查询latex中关于title的文档，可以使用texdoc -l title,输出如下

```
xuyang@ubuntu15:~/mybook$ texdoc -l title
 1 /usr/share/texmf/doc/context/third/title/title-doc.pdf
 2 /usr/share/texlive/texmf-dist/doc/latex/titleref/titleref.pdf
   = Package documentation
 3 /usr/share/texmf/doc/context/third/title/README
 4 /usr/share/texlive/texmf-dist/doc/latex/tufte-latex/graphics/be-title.pdf
 5 /usr/share/texlive/texmf-dist/doc/latex/tufte-latex/graphics/ei-title.pdf
```

```
 6 /usr/share/texlive/texmf-dist/doc/bibtex/economic/itaxpf-ex-title.pdf
 7 /usr/share/texlive/texmf-dist/doc/latex/tufte-latex/graphics/vdqi-title.pdf
 8 /usr/share/texlive/texmf-dist/doc/latex/tufte-latex/graphics/ve-title.pdf
Please enter the number of the file to view, anything else to skip:
```

如果使用-s模式，将会输出长达60多项的选择。

如果要在所有的tex文档中进行搜索，可以使用texdoc -l <keyword>进行搜索，这样会列出所有文件名中包含<keyword>的文档，你可以选择一份进行阅读。

Latex中文支持

列出可用的字体

使用命令fc-list,

要列出所有的中文字体，可使用命令fc-list -f "%{family}\n" :lang=zh，以下是我安装了一些字体以后的输出

```
AR PL UMing CN
AR PL UKai TW MBE
黑体,SimHei
AR PL UKai HK
Droid Sans Fallback
AR PL UKai CN
楷体_GB2312,KaiTi_GB2312
AR PL UKai TW
AR PL UMing HK
仿宋_GB2312,FangSong_GB2312
AR PL UMing TW
华文行楷,STXingkai
AR PL UMing TW MBE
```

其中，,后面是该字体的别名。

安装新的字体

可以从windows下拷贝一些中文字体过来，一下是使用这些字体的步骤

1. mkdir -p /usr/share/fonts/myfonts 在fonts目录下创建一个新的目录myfonts
2. 把字体文件考进去，比如SIMFANG.TTF、SimHei.sfd、SIMHEI.TTF等等
3. 进入这个字体目录，也就是/usr/share/fonts/myfonts，运行下面的命令:
    1. sudo mkfontscale create an index of scalable font files for X。 这个命令将会在当前目录下创建一个fonts.scale文件
    2. sudo mkfontdir create an index of X font files in a directory。这个命令会会在当前目录下创建一个fonts.dir文件
    3. sudo fc-cache -fv build font information cache files。
4. 重启电脑，然后运行fc-list，在输出里面应该就可以看到新添加的字体，比如

    ```
    /usr/share/fonts/myfonts/SIMFANG.TTF: 仿宋_GB2312,FangSong_GB2312:style=Regular
    /usr/share/fonts/myfonts/SIMHEI.TTF: 黑体,SimHei:style=Regular
    /usr/share/fonts/myfonts/SIMKAI.TTF: 楷体_GB2312,KaiTi_GB2312:style=Regular
    ```

如此，新的字体就可以使用了。

在pandoc中指定中文字体

这篇文档是用markdown写的，pandoc可以直接把markdown转为pdf，试着直接使用命令:

pandoc -f markdown -t latex --latex-engine=xelatex -o ubuntu_latex.md ubuntu_latex.md

注意，这里的参数--latex-engine=xelatex是必须的，否则latex会抱怨说有utf8编码不认识。利用上面的命令可以生成一个pdf，但是其中只有英文，没有中文!

我利用pandoc -D latex仔细查看了下其latex模板文件，其中有这样的行

```
$if(mainfont)$
    \setmainfont{$mainfont$}
$endif$
$if(sansfont)$
    \setsansfont{$sansfont$}
$endif$
$if(monofont)$
    \setmonofont[Mapping=tex-ansi]{$monofont$}
$endif$
$if(mathfont)$
    \setmathfont(Digits,Latin,Greek){$mathfont$}
$endif$
```

这里可以看到pandoc模板中是可以指定字体的，而pandoc的-M参数可以用来指定pandoc的 meta data。因此使用下面的命令行:

```
pandoc -f markdown -t latex  -M mainfont:KaiTi_GB2312 -M sansfont:FangSong_GB2312 -M
 monofont:SimHei --latex-engine=xelatex -o aa.pdf ubuntu_latex.md
```

其中，指定的字体是前面利用fc-list看到的中文字体。最终生成的pdf文件中中文可以正常显示。

现在看到的问题是，pdf中过长的行没有被自动换行而导致行后面的文字都丢掉了。但是英文的换行没有问题。

pandoc中的中文换行问题

上面问题在与中文行没有被换行，参考了一下Tzeng　Yuxio同学的模板，中文的换行通过\XeTeXlinebreaklocale "zh"实现。而在pandoc的缺省模板中是没有这个设置的。因此考虑修改缺省的latex模板。方法如下:

1. 导出缺省的模板 pandoc -D latex > template.latex
2. 在模板的这个部分下面加上这一行\XeTeXlinebreaklocale "zh"

   ```
   $if(mathfont)$
       \setmathfont(Digits,Latin,Greek){$mathfont$}
   $endif$
   ```

3. 在上面的命令行中指定使用新的模板。

新的命令行如下:

```
pandoc -f markdown -t latex -s -M mainfont:KaiTi_GB2312 -M sansfont:FangSong_GB2312 -M
    monofont:SimHei --latex-engine=xelatex -o aa.pdf
    -M geometry:"top=1in, inner=1in,outer=1in,bottom=1in, headheight=3ex, headsep=2ex"
    --template=./template.tex ubuntu_latex.md
```

注意，这里除了指定模板外，还指定了geometry，因为我发现即使中文换行了，还有一些代码到了页面以外，故而加上了geometry参数，这个命令行以后，生成的pdf文档基本上是一个看上来还可以的文档。不过字体的设置看起来不怎么合理，还有无序列表前的点没有出现，还需要进一步的调整。首先是要处理下无序列表。

无序列表前没有点号的原因

使用上面的命令行，无序列表前的点号总是出不来，四处查找，发现[1]这个问题是字体文件导致的。使用的字体中没有点号，所以无序列表的点号没有了。解决方案是使用比较新的字体文件。另一个比较简单的方法是在latex文件的导言区添加这一行指定这个点号

\renewcommand\labelitemi{\ensuremath{\bullet}}

对我而言，具体的添加位置是\XeTeXlinebreaklocale "zh"这一行的后面。

之后，生成的无序列表就没问题了。

pandoc中的pdf模板

pandoc对于latex输出使用的缺省模板如下

```
pandoc -D latex
#> \documentclass[$if(fontsize)$$fontsize$,$endif$$if(lang)$$babel-lang$,$endif$$if(papersize)$$papersize$paper,$endif$$f
#> $if(fontfamily)$
#> \usepackage[$for(fontfamilyoptions)$$fontfamilyoptions$$sep$,$endfor$]{$fontfamily$}
#> $else$
#> \usepackage{lmodern}
#> $endif$
#> $if(linestretch)$
#> \usepackage{setspace}
#> \setstretch{$linestretch$}
#> $endif$
#> \usepackage{amssymb,amsmath}
#> \usepackage{ifxetex,ifluatex}
#> \usepackage{fixltx2e} % provides \textsubscript
#> \ifnum 0\ifxetex 1\fi\ifluatex 1\fi=0 % if pdftex
#>   \usepackage[$if(fontenc)$$fontenc$$else$T1$endif$]{fontenc}
#>   \usepackage[utf8]{inputenc}
#> $if(euro)$
#>   \usepackage{eurosym}
#> $endif$
#> \else % if luatex or xelatex
#>   \ifxetex
#>     \usepackage{mathspec}
#>   \else
#>     \usepackage{fontspec}
#>   \fi
#>   \defaultfontfeatures{Ligatures=TeX,Scale=MatchLowercase}
#> $if(euro)$
#>   \newcommand{\euro}{  }
#> $endif$
#> $if(mainfont)$
#>     \setmainfont[$for(mainfontoptions)$$mainfontoptions$$sep$,$endfor$]{$mainfont$}
#> $endif$
#> $if(sansfont)$
#>     \setsansfont[$for(sansfontoptions)$$sansfontoptions$$sep$,$endfor$]{$sansfont$}
#> $endif$
#> $if(monofont)$
#>     \setmonofont[Mapping=tex-ansi$if(monofontoptions)$,$for(monofontoptions)$$monofontoptions$$sep$,$endfor$$endif$]{$
#> $endif$
```

[1] 参考博客文章 http://blog.sina.com.cn/s/blog_5e16f1770100uh63.html

```
#> $if(mathfont)$
#>     \setmathfont(Digits,Latin,Greek)[$for(mathfontoptions)$$mathfontoptions$$sep$,$endfor$]{$mathfont$}
#> $endif$
#> $if(CJKmainfont)$
#>     \usepackage{xeCJK}
#>     \setCJKmainfont[$for(CJKoptions)$$CJKoptions$$sep$,$endfor$]{$CJKmainfont$}
#> $endif$
#> \fi
#> % use upquote if available, for straight quotes in verbatim environments
#> \IfFileExists{upquote.sty}{\usepackage{upquote}}{}
#> % use microtype if available
#> \IfFileExists{microtype.sty}{%
#> \usepackage{microtype}
#> \UseMicrotypeSet[protrusion]{basicmath} % disable protrusion for tt fonts
#> }{}
#> $if(geometry)$
#> \usepackage[$for(geometry)$$geometry$$sep$,$endfor$]{geometry}
#> $endif$
#> \usepackage{hyperref}
#> $if(colorlinks)$
#> \PassOptionsToPackage{usenames,dvipsnames}{color} % color is loaded by hyperref
#> $endif$
#> \hypersetup{unicode=true,
#> $if(title-meta)$
#>             pdftitle={$title-meta$},
#> $endif$
#> $if(author-meta)$
#>             pdfauthor={$author-meta$},
#> $endif$
#> $if(keywords)$
#>             pdfkeywords={$for(keywords)$$keywords$$sep$; $endfor$},
#> $endif$
#> $if(colorlinks)$
#>             colorlinks=true,
#>             linkcolor=$if(linkcolor)$$linkcolor$$else$Maroon$endif$,
#>             citecolor=$if(citecolor)$$citecolor$$else$Blue$endif$,
#>             urlcolor=$if(urlcolor)$$urlcolor$$else$Blue$endif$,
#> $else$
#>             pdfborder={0 0 0},
#> $endif$
#>             breaklinks=true}
#> \urlstyle{same}  % don't use monospace font for urls
#> $if(lang)$
#> \ifnum 0\ifxetex 1\fi\ifluatex 1\fi=0 % if pdftex
#>   \usepackage[shorthands=off,$for(babel-otherlangs)$$babel-otherlangs$,$endfor$main=$babel-lang$]{babel}
#> $if(babel-newcommands)$
#>   $babel-newcommands$
#> $endif$
#> \else
#>   \usepackage{polyglossia}
#>   \setmainlanguage[$polyglossia-lang.options$]{$polyglossia-lang.name$}
#> $for(polyglossia-otherlangs)$
#>   \setotherlanguage[$polyglossia-otherlangs.options$]{$polyglossia-otherlangs.name$}
#> $endfor$
#> \fi
```

```
#> $endif$
#> $if(natbib)$
#> \usepackage{natbib}
#> \bibliographystyle{$if(biblio-style)$$biblio-style$$else$plainnat$endif$}
#> $endif$
#> $if(biblatex)$
#> \usepackage$if(biblio-style)$[style=$biblio-style$]$endif${biblatex}
#> $if(biblatexoptions)$\ExecuteBibliographyOptions{$for(biblatexoptions)$$biblatexoptions$$sep$,$endfor$}$endif$
#> $for(bibliography)$
#> \addbibresource{$bibliography$}
#> $endfor$
#> $endif$
#> $if(listings)$
#> \usepackage{listings}
#> $endif$
#> $if(lhs)$
#> \lstnewenvironment{code}{\lstset{language=Haskell,basicstyle=\small\ttfamily}}{}
#> $endif$
#> $if(highlighting-macros)$
#> $highlighting-macros$
#> $endif$
#> $if(verbatim-in-note)$
#> \usepackage{fancyvrb}
#> \VerbatimFootnotes % allows verbatim text in footnotes
#> $endif$
#> $if(tables)$
#> \usepackage{longtable,booktabs}
#> $endif$
#> $if(graphics)$
#> \usepackage{graphicx,grffile}
#> \makeatletter
#> \def\maxwidth{\ifdim\Gin@nat@width>\linewidth\linewidth\else\Gin@nat@width\fi}
#> \def\maxheight{\ifdim\Gin@nat@height>\textheight\textheight\else\Gin@nat@height\fi}
#> \makeatother
#> % Scale images if necessary, so that they will not overflow the page
#> % margins by default, and it is still possible to overwrite the defaults
#> % using explicit options in \includegraphics[width, height, ...]{}
#> \setkeys{Gin}{width=\maxwidth,height=\maxheight,keepaspectratio}
#> $endif$
#> $if(links-as-notes)$
#> % Make links footnotes instead of hotlinks:
#> \renewcommand{\href}[2]{#2\footnote{\url{#1}}}
#> $endif$
#> $if(strikeout)$
#> \usepackage[normalem]{ulem}
#> % avoid problems with \sout in headers with hyperref:
#> \pdfstringdefDisableCommands{\renewcommand{\sout}{}}
#> $endif$
#> $if(indent)$
#> $else$
#> \IfFileExists{parskip.sty}{%
#> \usepackage{parskip}
#> }{% else
#> \setlength{\parindent}{0pt}
#> \setlength{\parskip}{6pt plus 2pt minus 1pt}
```

```
#> }
#> $endif$
#> \setlength{\emergencystretch}{3em}  % prevent overfull lines
#> \providecommand{\tightlist}{%
#>   \setlength{\itemsep}{0pt}\setlength{\parskip}{0pt}}
#> $if(numbersections)$
#> \setcounter{secnumdepth}{5}
#> $else$
#> \setcounter{secnumdepth}{0}
#> $endif$
#> $if(subparagraph)$
#> $else$
#> % Redefines (sub)paragraphs to behave more like sections
#> \ifx\paragraph\undefined\else
#> \let\oldparagraph\paragraph
#> \renewcommand{\paragraph}[1]{\oldparagraph{#1}\mbox{}}
#> \fi
#> \ifx\subparagraph\undefined\else
#> \let\oldsubparagraph\subparagraph
#> \renewcommand{\subparagraph}[1]{\oldsubparagraph{#1}\mbox{}}
#> \fi
#> $endif$
#> $if(dir)$
#> \ifxetex
#>   % load bidi as late as possible as it modifies e.g. graphicx
#>   $if(latex-dir-rtl)$
#>   \usepackage[RTLdocument]{bidi}
#>   $else$
#>   \usepackage{bidi}
#>   $endif$
#> \fi
#> \ifnum 0\ifxetex 1\fi\ifluatex 1\fi=0 % if pdftex
#>   \TeXXeTstate=1
#>   \newcommand{\RL}[1]{\beginR #1\endR}
#>   \newcommand{\LR}[1]{\beginL #1\endL}
#>   \newenvironment{RTL}{\beginR}{\endR}
#>   \newenvironment{LTR}{\beginL}{\endL}
#> \fi
#> $endif$
#> $for(header-includes)$
#> $header-includes$
#> $endfor$
#>
#> $if(title)$
#> \title{$title$$if(thanks)$\thanks{$thanks$}$endif$}
#> $endif$
#> $if(subtitle)$
#> \providecommand{\subtitle}[1]{}
#> \subtitle{$subtitle$}
#> $endif$
#> $if(author)$
#> \author{$for(author)$$author$$sep$ \and $endfor$}
#> $endif$
#> $if(institute)$
#> \institute{$for(institute)$$institute$$sep$ \and $endfor$}
```

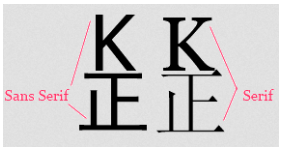
```
#> $endif$
#> \date{$date$}
#>
#> \begin{document}
#> $if(title)$
#> \maketitle
#> $endif$
#> $if(abstract)$
#> \begin{abstract}
#> $abstract$
#> \end{abstract}
#> $endif$
#>
#> $for(include-before)$
#> $include-before$
#>
#> $endfor$
#> $if(toc)$
#> {
#> $if(colorlinks)$
#> \hypersetup{linkcolor=$if(toccolor)$$toccolor$$else$black$endif$}
#> $endif$
#> \setcounter{tocdepth} {$toc-depth$}
#> \tableofcontents
#> }
#> $endif$
#> $if(lot)$
#> \listoftables
#> $endif$
#> $if(lof)$
#> \listoffigures
#> $endif$
#> $body$
#>
#> $if(natbib)$
#> $if(bibliography)$
#> $if(biblio-title)$
#> $if(book-class)$
#> \renewcommand\bibname{$biblio-title$}
#> $else$
#> \renewcommand\refname{$biblio-title$}
#> $endif$
#> $endif$
#> \bibliography{$for(bibliography)$$bibliography$$sep$,$endfor$}
#>
#> $endif$
#> $endif$
#> $if(biblatex)$
#> \printbibliography$if(biblio-title)$[title=$biblio-title$]$endif$
#>
#> $endif$
#> $for(include-after)$
#> $include-after$
#>
#> $endfor$
```

```
#> \end{document}
```

关于字体的更多说明

前面我们看到有写关于字体的设置，如setmainfont,setsansfont,setmonofont等。其中的mainfont为正文罗马族，sansfont意为正文无衬线族，monofont为正文等宽族。这些为fontspec包中的设置。另外有一个包叫xeCJK，可以使用texdoc xecjk来查看这个包的文档。这个包由国人维护，内容也是中文的。其中有关于如何查找和使用中文字体的说明。

在西方字体中，分为两大类，Sans Serif和Serif，其中Serif的意思是在笔画起始处有额外的修饰。而Sans Serif字体则是没有修饰的。这样对比起来，Serif就相对于中文的楷体，而Sans Serif就相当于是黑体。其区别如下所示



除此之外，另外一种字体叫等宽字体，即monofont，指每个字的宽度一致，通常用来显示代码。

设置标题的字体为黑体

下面来看一个具体的问题，如何将文章的标题（包括章节的标题）设置为黑体。宏包titlesec可以重新定义各级标题的样式。可以参考这两篇博文[2], [3]。

根据文章[4]的建议，我把setmainfont{KaiTi_GB2312}改为setmainfont[BoldFont=SimHei]{KaiTi_GB2312}之后，可以看到生成的pdf文件标题全部变成了黑体，实现了想要的效果。

[2] http://blog.sina.com.cn/s/blog_5e16f1770100lqn7.html
[3] http://blog.csdn.net/u012675539/article/details/49919121
[4] http://blog.csdn.net/u012675539/article/details/49919121

Chapter 5

由markdown生成幻灯片

在使用markdown制作html和pdf之后，另一个经常使用到文档的场景就是PPT了，也就是幻灯片。现在我们在linux下，希望能通过pandoc的帮助来制作一个ppt。

一个实例

因为pandoc本身就支持一种幻灯片格式beamer，现在我们来做一个四页的幻灯片，看看效果如何:

```
# markdown简介

* 什么是markdown

* 如何使用markdown


# markdown语法

* 标题

* 引用

* 链接

* 图片

# 用markdown制作pdf

* 如何生成pdf

* 模板的使用

# 用markdown制作网站

* 生成html的例子

* html的模板
```

假设将上面的文件存为aa.md，那么通过命令行

```
pandoc aa.md -f markdown -t beamer -o aa.pdf --latex-engine=xelatex
```

可以生成aa.pdf，而且生成的pdf和我们想象的一样，该ppt是四页的。

但是和前面生成pdf时碰到的问题一样，中文没有，我们还是需要去研究下它的模板。它的模板是这样的

```
pandoc -D beamer
#> \documentclass[$if(fontsize)$$fontsize$,$endif$$if(lang)$$babel-lang$,$endif$$if(handout)$handout,$endif$$if(beamer)$i
#> \setbeamertemplate{caption}[numbered]
#> \setbeamertemplate{caption label separator}{: }
#> \setbeamercolor{caption name}{fg=normal text.fg}
#> \beamertemplatenavigationsymbols$if(navigation)$$navigation$$else$empty$endif$
#> $if(fontfamily)$
#> \usepackage[$for(fontfamilyoptions)$$fontfamilyoptions$$sep$,$endfor$]{$fontfamily$}
#> $else$
#> \usepackage{lmodern}
#> $endif$
#> \usepackage{amssymb,amsmath}
#> \usepackage{ifxetex,ifluatex}
#> \usepackage{fixltx2e} % provides \textsubscript
#> \ifnum 0\ifxetex 1\fi\ifluatex 1\fi=0 % if pdftex
#>   \usepackage[$if(fontenc)$$fontenc$$else$T1$endif$]{fontenc}
#>   \usepackage[utf8]{inputenc}
#> $if(euro)$
#>   \usepackage{eurosym}
#> $endif$
#> \else % if luatex or xelatex
#>   \ifxetex
#>     \usepackage{mathspec}
#>   \else
#>     \usepackage{fontspec}
#>   \fi
#>   \defaultfontfeatures{Ligatures=TeX,Scale=MatchLowercase}
#> $if(euro)$
#>   \newcommand{\euro}{ }
#> $endif$
#> $if(mainfont)$
#>     \setmainfont[$for(mainfontoptions)$$mainfontoptions$$sep$,$endfor$]{$mainfont$}
#> $endif$
#> $if(sansfont)$
#>     \setsansfont[$for(sansfontoptions)$$sansfontoptions$$sep$,$endfor$]{$sansfont$}
#> $endif$
#> $if(monofont)$
#>     \setmonofont[Mapping=tex-ansi$if(monofontoptions)$,$for(monofontoptions)$$monofontoptions$$sep$,$endfor$$endif$]{$
#> $endif$
#> $if(mathfont)$
#>     \setmathfont(Digits,Latin,Greek)[$for(mathfontoptions)$$mathfontoptions$$sep$,$endfor$]{$mathfont$}
#> $endif$
#> $if(CJKmainfont)$
#>     \usepackage{xeCJK}
#>     \setCJKmainfont[$for(CJKoptions)$$CJKoptions$$sep$,$endfor$]{$CJKmainfont$}
#> $endif$
#> \fi
#> $if(theme)$
#> \usetheme{$theme$}
#> $endif$
#> $if(colortheme)$
#> \usecolortheme{$colortheme$}
#> $endif$
#> $if(fonttheme)$
```

```
#> \usefonttheme{$fonttheme$}
#> $endif$
#> $if(mainfont)$
#> \usefonttheme{serif} % use mainfont rather than sansfont for slide text
#> $endif$
#> $if(innertheme)$
#> \useinnertheme{$innertheme$}
#> $endif$
#> $if(outertheme)$
#> \useoutertheme{$outertheme$}
#> $endif$
#> % use upquote if available, for straight quotes in verbatim environments
#> \IfFileExists{upquote.sty}{\usepackage{upquote}}{}
#> % use microtype if available
#> \IfFileExists{microtype.sty}{%
#> \usepackage{microtype}
#> \UseMicrotypeSet[protrusion]{basicmath} % disable protrusion for tt fonts
#> }{}
#> $if(lang)$
#> \ifnum 0\ifxetex 1\fi\ifluatex 1\fi=0 % if pdftex
#>   \usepackage[shorthands=off,$for(babel-otherlangs)$$babel-otherlangs$,$endfor$main=$babel-lang$]{babel}
#> $if(babel-newcommands)$
#>   $babel-newcommands$
#> $endif$
#> \else
#>   \usepackage{polyglossia}
#>   \setmainlanguage[$polyglossia-lang.options$]{$polyglossia-lang.name$}
#> $for(polyglossia-otherlangs)$
#>   \setotherlanguage[$polyglossia-otherlangs.options$]{$polyglossia-otherlangs.name$}
#> $endfor$
#> \fi
#> $endif$
#> \newif\ifbibliography
#> $if(natbib)$
#> \usepackage{natbib}
#> \bibliographystyle{$if(biblio-style)$$biblio-style$$else$plainnat$endif$}
#> $endif$
#> $if(biblatex)$
#> \usepackage$if(biblio-style)$[style=$biblio-style$]$endif${biblatex}
#> $if(biblatexoptions)$\ExecuteBibliographyOptions{$for(biblatexoptions)$$biblatexoptions$$sep$,$endfor$}$endif$
#> $for(bibliography)$
#> \addbibresource{$bibliography$}
#> $endfor$
#> $endif$
#> $if(listings)$
#> \usepackage{listings}
#> $endif$
#> $if(lhs)$
#> \lstnewenvironment{code}{\lstset{language=Haskell,basicstyle=\small\ttfamily}}{}
#> $endif$
#> $if(highlighting-macros)$
#> $highlighting-macros$
#> $endif$
#> $if(verbatim-in-note)$
#> \usepackage{fancyvrb}
```

```
#> \VerbatimFootnotes % allows verbatim text in footnotes
#> $endif$
#> $if(tables)$
#> \usepackage{longtable,booktabs}
#> \usepackage{caption}
#> % These lines are needed to make table captions work with longtable:
#> \makeatletter
#> \def\fnum@table{\tablename~\thetable}
#> \makeatother
#> $endif$
#> $if(graphics)$
#> \usepackage{graphicx,grffile}
#> \makeatletter
#> \def\maxwidth{\ifdim\Gin@nat@width>\linewidth\linewidth\else\Gin@nat@width\fi}
#> \def\maxheight{\ifdim\Gin@nat@height>\textheight0.8\textheight\else\Gin@nat@height\fi}
#> \makeatother
#> % Scale images if necessary, so that they will not overflow the page
#> % margins by default, and it is still possible to overwrite the defaults
#> % using explicit options in \includegraphics[width, height, ...]{}
#> \setkeys{Gin}{width=\maxwidth,height=\maxheight,keepaspectratio}
#> $endif$
#>
#> % Prevent slide breaks in the middle of a paragraph:
#> \widowpenalties 1 10000
#> \raggedbottom
#>
#> $if(section-titles)$
#> \AtBeginPart{
#>   \let\insertpartnumber\relax
#>   \let\partname\relax
#>   \frame{\partpage}
#> }
#> \AtBeginSection{
#>   \ifbibliography
#>   \else
#>     \let\insertsectionnumber\relax
#>     \let\sectionname\relax
#>     \frame{\sectionpage}
#>   \fi
#> }
#> \AtBeginSubsection{
#>   \let\insertsubsectionnumber\relax
#>   \let\subsectionname\relax
#>   \frame{\subsectionpage}
#> }
#> $endif$
#>
#> $if(links-as-notes)$
#> % Make links footnotes instead of hotlinks:
#> \renewcommand{\href}[2]{#2\footnote{\url{#1}}}
#> $endif$
#> $if(strikeout)$
#> \usepackage[normalem]{ulem}
#> % avoid problems with \sout in headers with hyperref:
#> \pdfstringdefDisableCommands{\renewcommand{\sout}{}}
```

```
#> $endif$
#> \setlength{\emergencystretch}{3em}  % prevent overfull lines
#> \providecommand{\tightlist}{%
#>   \setlength{\itemsep}{0pt}\setlength{\parskip}{0pt}}
#> $if(numbersections)$
#> \setcounter{secnumdepth}{5}
#> $else$
#> \setcounter{secnumdepth}{0}
#> $endif$
#> $if(dir)$
#> \ifxetex
#>   % load bidi as late as possible as it modifies e.g. graphicx
#>   $if(latex-dir-rtl)$
#>   \usepackage[RTLdocument]{bidi}
#>   $else$
#>   \usepackage{bidi}
#>   $endif$
#> \fi
#> \ifnum 0\ifxetex 1\fi\ifluatex 1\fi=0 % if pdftex
#>   \TeXXeTstate=1
#>   \newcommand{\RL}[1]{\beginR #1\endR}
#>   \newcommand{\LR}[1]{\beginL #1\endL}
#>   \newenvironment{RTL}{\beginR}{\endR}
#>   \newenvironment{LTR}{\beginL}{\endL}
#> \fi
#> $endif$
#> $for(header-includes)$
#> $header-includes$
#> $endfor$
#>
#> $if(title)$
#> \title{$title$}
#> $endif$
#> $if(subtitle)$
#> \subtitle{$subtitle$}
#> $endif$
#> $if(author)$
#> \author{$for(author)$$author$$sep$ \and $endfor$}
#> $endif$
#> $if(institute)$
#> \institute{$for(institute)$$institute$$sep$ \and $endfor$}
#> $endif$
#> \date{$date$}
#>
#> \begin{document}
#> $if(title)$
#> \frame{\titlepage}
#> $endif$
#>
#> $for(include-before)$
#> $include-before$
#>
#> $endfor$
#> $if(toc)$
#> \begin{frame}
```

```
#> \tableofcontents[hideallsubsections]
#> \end{frame}
#>
#> $endif$
#> $body$
#>
#> $if(natbib)$
#> $if(bibliography)$
#> $if(biblio-title)$
#> $if(book-class)$
#> \renewcommand\bibname{$biblio-title$}
#> $else$
#> \renewcommand\refname{$biblio-title$}
#> $endif$
#> $endif$
#> \begin{frame}[allowframebreaks]{$biblio-title$}
#> \bibliographytrue
#> \bibliography{$for(bibliography)$$bibliography$$sep$,$endfor$}
#> \end{frame}
#>
#> $endif$
#> $endif$
#> $if(biblatex)$
#> \begin{frame}[allowframebreaks]{$biblio-title$}
#> \bibliographytrue
#> \printbibliography[heading=none]
#> \end{frame}
#>
#> $endif$
#> $for(include-after)$
#> $include-after$
#>
#> $endfor$
#> \end{document}
```

在这个模板中，我们没有像pdf模板中那样看到有设置字体的地方。看起来我们需要自己设定字体了。

修改模板

查看一下，把字体设置放在\newcommand{\euro}{}的后面，其中字体设置部分的内容和pdf模板中一样

这样首先利用pandoc -D beamer > beamer.tpl导出原来的模板文件，然后在其中添加如下内容（最后一行是原有的内容）：

```
    \setmainfont[BoldFont=SimHei]{KaiTi_GB2312}
    \setsansfont{SimHei}
    \setmonofont[Mapping=tex-ansi]{FangSong_GB2312}
    \XeTeXlinebreaklocale "zh"
% use upquote if available, for straight quotes in verbatim environments
```

更新这个模板以后，使用命令

```
pandoc -f markdown -t beamer -o aa.pdf --latex-engine=xelatex --template=./beamer.tpl ./aa.md
```

如此生成的PPT基本上就是我们想要的了，中文可以正常显示。如下所示:


aa.pdf
2
共 4 页
缩略图
1
2
3
4
markdown语法
标题
引用
链接
图片

Chapter 6

用knitr动态生成markdown文件的内容

knitr是R的一个包，作者是谢益辉，目前就职于RStudio。这个包用来动态生成文档的内容，markdown在没有碰到knitr之前，只是显示了它简洁的一面。当它碰到knitr的时候，就立刻显示了markdown最强大的一面。内容可以动态生成，而且可随着代码的变化自动更新。

用knitr动态生成内容

在markdown中利用knitr动态生成内容的方法是，使用一个含有特殊标记的代码块，knitr会运行其中的代码，并抓取该代码的输入，最后把代码的输出作为文档的一部分。如下所示

```
```{.r echo=FALSE}
print("hello knitr")
```
```

它将会产生一段内容插入到当前的markdown文件中。如下所示:

```
#> [1] "hello knitr"
```

其中的.r告诉knitr使用R语言引擎来执行程序，echo=FALSE是knitr能够识别的一个参数，这个意思是，这段代码本身不会出现在最后的文档中。

现在knitr可以支持多种语言引擎，目前知道的有r,bash,pl,python等。

后面的参数可以控制代码的执行以及输出的格式等，常见的参数有：（多个参数用,分开）

- eval 是否求值当前代码块
- echo 是否显示源代码
- results ‘markup’ , ‘asis’
- tidy 是否美化R代码
- fig.width fig.height out.width out.height 设备和输出图片的大小
- fig.path 输出图片的前缀，如’./rfigures/p123- ‘,将生成’./rfigures/p123-…png’
- include 是否在结果中包含当前块的结果
- collapse 输出和代码之间是否分开，缺省为 “FALSE “,即分开
- comment 输出前的注释符，缺省为’ ##’

除了在代码块上指定参数以外，也可以使用ops_chunk$set(tidy=FALSE) 来设置这些参数。如果设定这些参数以后，会在当前文件中一直有效。所以，如果为了使代码块更简单，可以使用这个函数来设置一些参数。

另外，knitr中除了每个代码块的参数外，还有包级别的参数，可以使用opts_knit$set(progress=TRUE,verbose=TRUE)这样的方式来设置。常见的包级别参数有:

- verbose 是否输出更多信息

所有的knitr参数，可以参考这里: http://yihui.name/knitr/options/#package_options

Table 6.1:

Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species
5.1	3.5	1.4	0.2	setosa
4.9	3.0	1.4	0.2	setosa
4.7	3.2	1.3	0.2	setosa
4.6	3.1	1.5	0.2	setosa
5.0	3.6	1.4	0.2	setosa
5.4	3.9	1.7	0.4	setosa

用knitr插入动态表格

在knitr中生成表格，可以使用knitr::kable。

```
```{.r}
knitr::kable(head(iris))
```
```

注意这里的kable是带有参数的，即生成html和latex时要用不同的参数调用，因此这个函数不宜直接放在rmd文件中。最好的方式是使用一个函数来封装这个函数，在封装函数中判定当前的目标格式，然后再利用不同的参数来调用这个函数。在实际的使用中，我用blogtable来封装这个函数，这个函数是这样定义的:

```
blogtable<-function(x)
{
        if(is_latex()) {
                knitr::kable(x,format="latex",align="c")
        }else{
                knitr::kable(x,format="html",table.attr = "class=\"table table-bordered\"", align="c")
        }
}
```

后面的table.attr用来指定表格的css，这样html中生成的每个表格会居中显示。这样，我就可以在我的rmd文件中这样使用它了:

```
```{.r}
rwp::blogtable(head(iris))
```
```

其中的rwp是我另外做的一个包，用以封装一些常用的函数。这句话生成的表格是这样的:

也可以在knitr块中使用pander包生成markdown的表格，pander可支持除了表以外的更多东西，也可以支持更多的表格，通常的表格应该使用knitr::kable就可以了。

下面是knitr中使用pander的方法

```
```{.r results='asis'}
pander::pandoc.table(head(iris),style='simple')
```
```

这样生成的表格基本和knitr::kable一样。下面是pandoc.table的函数声明，可以看出这是一个非常复杂的函数:

```
Description:

     Creates a Pandoc's markdown style table with optional caption and
     some other tweaks. See 'Details' below.

Usage:

     pandoc.table.return(t, caption, digits = panderOptions("digits"),
       decimal.mark = panderOptions("decimal.mark"),
```

Table 6.2:

link	index
[a](http://www.bagualu.net/wordpress/archives/6223)	1
[b](http://www.bagualu.net/wordpress/archives/6220)	2
[c](http://www.bagualu.net/wordpress/archives/6210)	3
[d](http://www.bagualu.net/wordpress/archives/6205)	4
[e](http://www.bagualu.net/wordpress/archives/6213)	5

```
        big.mark = panderOptions("big.mark"), round = panderOptions("round"),
        missing = panderOptions("missing"), justify, style = c("multiline",
        "grid", "simple", "rmarkdown"),
        split.tables = panderOptions("table.split.table"),
        split.cells = panderOptions("table.split.cells"),
        keep.trailing.zeros = panderOptions("keep.trailing.zeros"),
        keep.line.breaks = panderOptions("keep.line.breaks"),
        plain.ascii = panderOptions("plain.ascii"),
        use.hyphening = panderOptions("use.hyphening"),
        emphasize.rownames = panderOptions("table.emphasize.rownames"),
        emphasize.rows, emphasize.cols, emphasize.cells, emphasize.strong.rows,
        emphasize.strong.cols, emphasize.strong.cells, emphasize.italics.rows,
        emphasize.italics.cols, emphasize.italics.cells, emphasize.verbatim.rows,
        emphasize.verbatim.cols, emphasize.verbatim.cells, ...)
```

更多pander包的使用说明，看这里： http://mirror.bjtu.edu.cn/cran/web/packages/pander/index.html

如何在表格中使用格式标签

常见的用例是需要在表格中显示超链。即，假设我们有这样的一个data.frame.

```
aa=c(6223,6220,6210,6205,6213)
bb=letters[1:5]
link=paste0("http://www.bagualu.net/wordpress/archives/",aa)
mdlink=paste0("[",bb,"](",link,")")
dd=data.frame(link=mdlink,index=1:5)
rwp::blogtable(dd)
```

以上代码在html格式下，可以正常的显示，即，表格中的第一行是生成了超链，但是在latex情况下，超链却没有生成，表格的第一列是原始的markdown源码。其中的原因是pandoc中有一个扩展叫makrdown_in_html_blocks ，但是没有一个类似的makrdown_in_tex_blocks，这样在html中的markdown可以被识别，但是latex格式下的markdown不会被识别。

如何在latex格式下，生成带有超链的表格呢？一个可行的方式是生成表格各单元格的latex表示，然后在kable中指定参数escape=FALSE，如果不指定escape参数，在latex的表格中也不会有链接，因为其中的特殊字符会被escape掉。

用这种方式来解决问题的画，生成表格的地方就需要两个版本，一个html格式的一个latex格式的。

```
aa=c(6223,6220,6210,6205,6213)
bb=letters[1:5]
link=paste0("http://www.bagualu.net/wordpress/archives/",aa)
if(rwp::is_latex()){
    latexlink=paste0("\\href{",link,"}{",bb,"}")
    dd=data.frame(link=latexlink,index=1:5)
    knitr::kable(dd,format="latex",escape=FALSE)
}else{
    mdlink=paste0("[",bb,"](",link,")")
    dd=data.frame(link=mdlink,index=1:5)
```

```
    rwp::blogtable(dd) #knitr::kable(dd) + css
}
```

link	index
a	1
b	2
c	3
d	4
e	5

对于以上这个例子，这个解决方案是没有问题的。但是仔细分析一下，可以发现这个解决方案是有问题的，也就是说kable的escape被设置成了FALSE,而一旦这个值被设置为FALSE，dataframe中所有的内容都不会被escape，如果dataframe中其他列出现tex中的特殊字符，那就会出问题。因为kable中不能对每个列控制是否escaple，所以kable基本不能处理这种情况。

一个可行的处理方法是在kable上面再加一层，这里对dataframe中的每一列分别设定是否需要做escape，对需要做escape的列做escape，然后在对这个dataframe调用kable,，此时设定escape为FALSE即可。对于kable上面包的这一层，代码是这样的:

```
blogtable<-function(x, escape=TRUE, none_escape_column=NULL)
{
        #only data frame is supported
        if(is.data.frame(x)){
            if(! is.null(none_escape_column) ) {
                x=partial_escape_dataframe(x,except=none_escape_column)
                escape=FALSE
            }
            if(is_latex()) {
                    knitr::kable(x,format="latex",align="c",escape=escape)
            }else{
                knitr::kable(x,format="html",table.attr = "class=\"table table-bordered\"", align="c", escape=escape)
            }
        }else{
                stop("blogtable supports only data.frame")
        }
}

partial_escape_dataframe<-function(x,except=none_escape_column)
{
    if(is.null(except)){
        stop("error in partial-escape-dataframe")
    }
    if(! is.numeric(except))
    {
        nn=names(x)
        needescap=setdiff(nn,except)
    }else{
        needescap=setdiff(1:dim(x)[2],except)
    }

    for(idx in needescap){
        x[,idx]=escape_wrap(x[,idx])
    }
    x
}
```

这样在调用blogtable时，只要指定none_escape_column就可以了，一个调用示例如下:

Table 6.3: 几篇博文

link	index
a	1
b	2
c	3
d	4
e	5

```
aa=c(6223,6220,6210,6205,6213)
bb=letters[1:5]
link=paste0("http://www.bagualu.net/wordpress/archives/",aa)
rlink=rwp::link(link,bb)
dd=data.frame(link=rlink,index=1:5)
rwp::blogtable(dd,none_escape_column=1)
```

其中rwp::link的定义如下:

```
link<-function(href,name)
{
        if(is_latex()){
                paste0("\\href{",href,"}{",name,"}")
        }else{
                paste0("<a href='",href,"'>",name,"</a>")
        }
}
```

这样对于每种二者有区别的格式都写一个类似link的函数，这样就能在文档中使用一份代码了。

为表格添加标题

kable中有一个参数叫caption，这个参数用来指定表格的标题，现在，我们把这个参数也加到blogtable中去。试验一个实例:

```
aa=c(6223,6220,6210,6205,6213)
bb=letters[1:5]
link=paste0("http://www.bagualu.net/wordpress/archives/",aa)
rlink=rwp::link(link,bb)
dd=data.frame(link=rlink,index=1:5)
rwp::blogtable(dd,none_escape_column=1,caption="几篇博文")
```

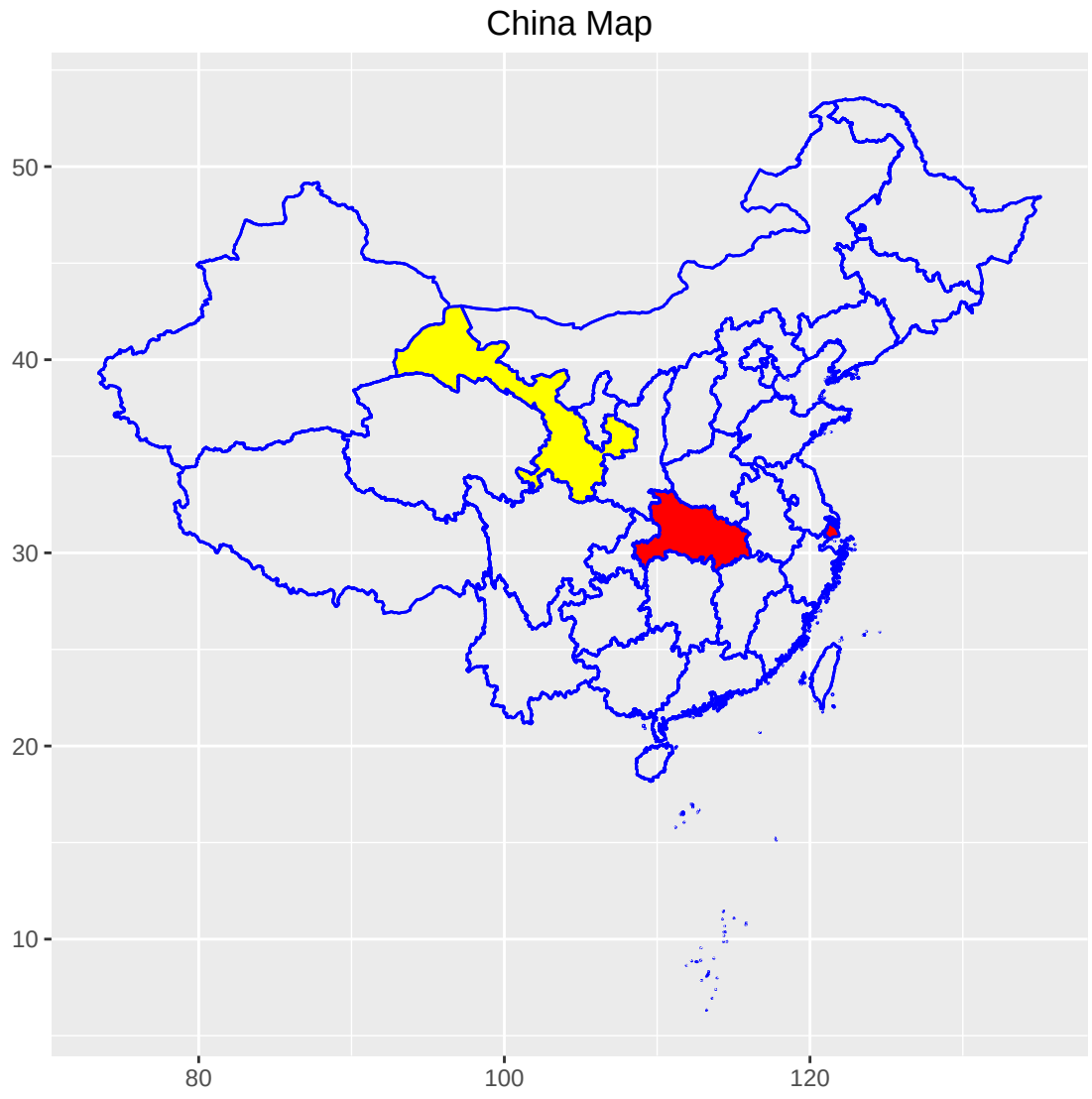
用knitr插入动态图片

在有图片生成的r块中，用fig.path指定图片的前缀。(注意在实际的页面中没有eval=FALSE参数)

```
```{.r fig.path='rfigures/p5304-',eval=FALSE}
library(rcnmap)
#分别使用红色红色黄色在中国地图上画出湖北，上海和甘肃省
cnmap(c("hubei","shanghai","gansu"),c("red","red","yellow"))
```
```

```
#> Regions defined for each Polygons
#> Regions defined for each Polygons
#> Regions defined for each Polygons
#> Regions defined for each Polygons
```

```
#> Regions defined for each Polygons
```



生成的图片在rfigures/p5304-unnamed-chunk-2-1.png（相对当前的工作目录）

然后的html中，图的链接为:

为了让这个图能够在最终的页面中显示出来，你需要把这个图拷贝到适当的地方。

对我而言，这个地址是: http://www.bagualu.net/wordpress/archives/rfigures/

为图片添加标题

在knitr的块属性中设置fig.cap, 如下所示（其中也添加了一个对齐参数，设置为居中对齐，后面可以看到这个参数的效果）

```
```{r fig.cap="我的中国地图",fig.align='center'}
library(rcnmap)
#分别使用红色红色黄色在中国地图上画出湖北，上海和甘肃省
cnmap(c("hubei","shanghai","gansu"),c("red","red","yellow"))
```
```

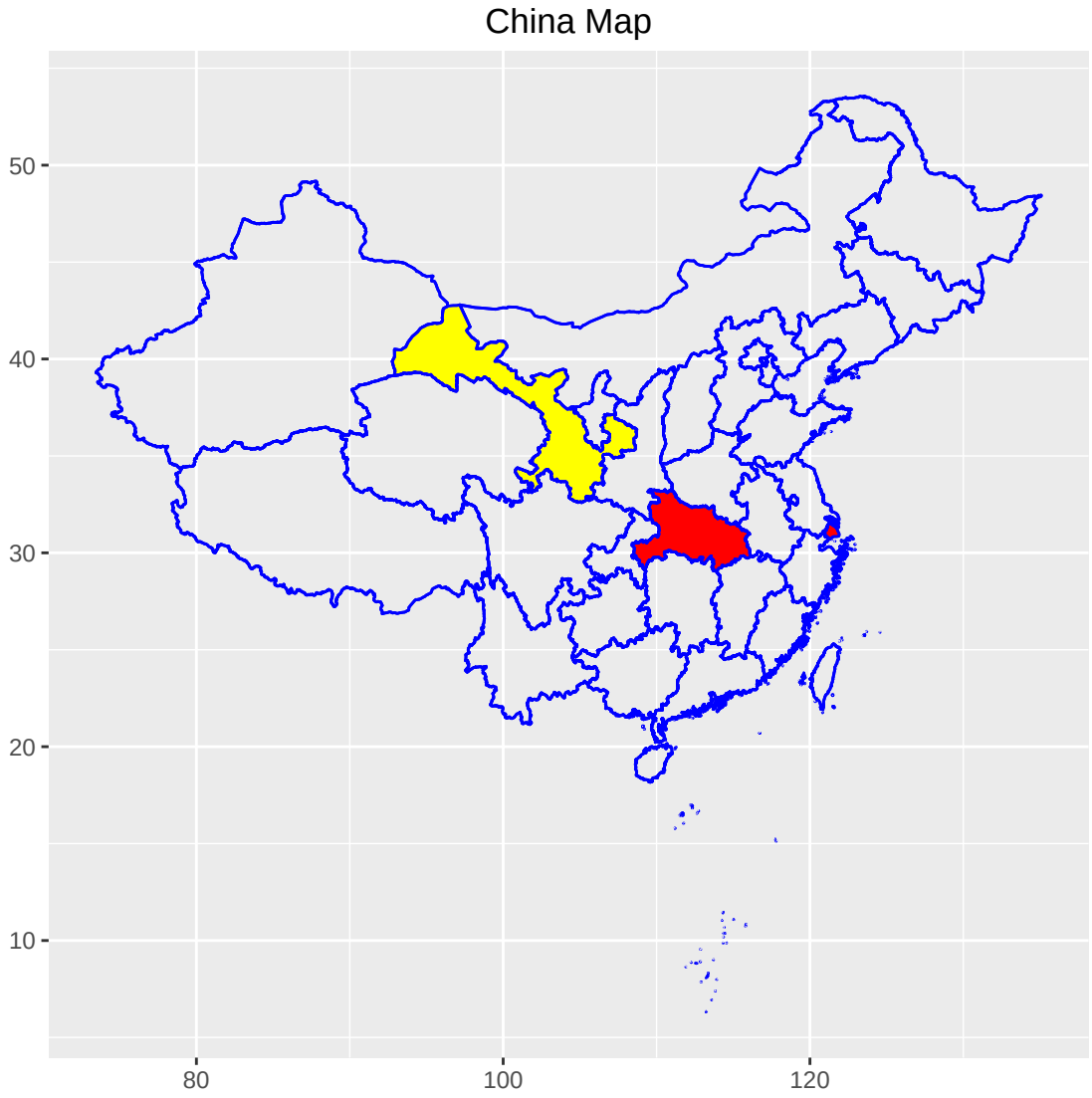


Figure 6.1: 我的中国地图

```
#> Regions defined for each Polygons
#> Regions defined for each Polygons
#> Regions defined for each Polygons
#> Regions defined for each Polygons
#> Regions defined for each Polygons
```

Chapter 7

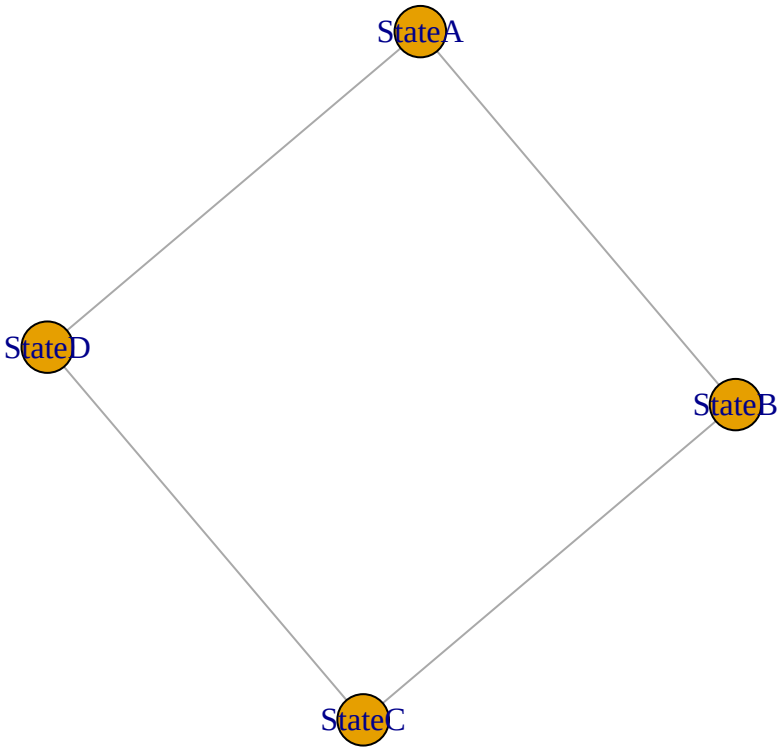
用R的igraph包创建和绘制流程图

这个包igraph主要是用来处理图。包括很复杂的图。这是一个很大的包，其手册长达400页。这里只是关心其中创建图和绘制图的部分。

创建图

先看一个简单的例子：利用graph_from_literal创建一个图，然后把它画出来：

```
library(igraph)
library(diagram)
g1=graph_from_literal(StateA---StateB---StateC---StateD,StateA---StateD)
plot(g1)
```

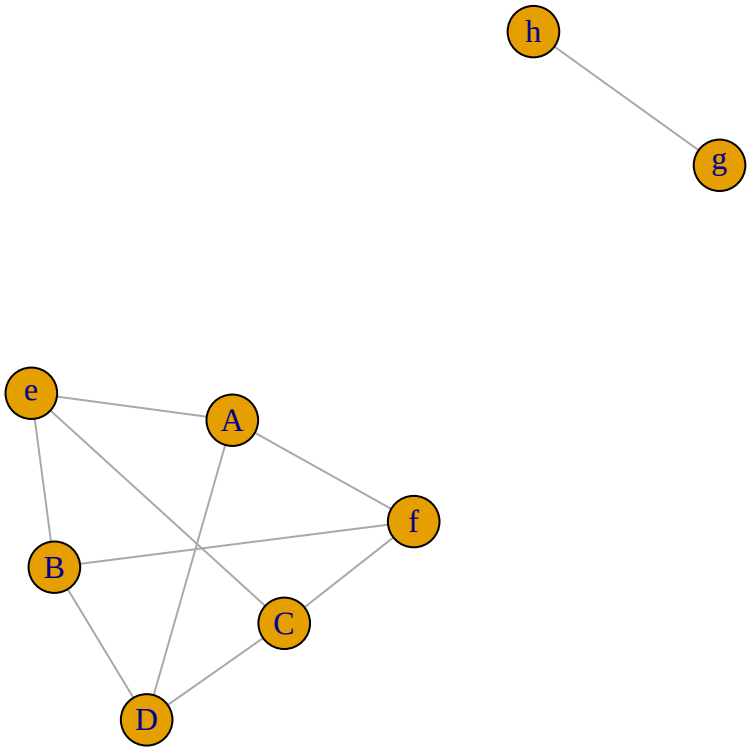


这里可以看到，由igraph创建并绘制图是很方便的。下面详细看看函数graph_from_literal的更多用法:

无向图

- 无向图之间的连接使用-，或者---,这个长度是任意的，如graph_from_literal(A-B) 和 graph_from_literal(A-----B) 等价 ， 多个顶点可以直接相连，如graph_from_literal(A---B----C---D)
- 图中的孤立点，可以逗号分开，作为独立的参数，如 graph_from_literal(A---B----C---D,e--f,g--h,i)
- 顶点集，使用:连接的顶点为一个顶点集，顶点集之间的点是全相连的。如

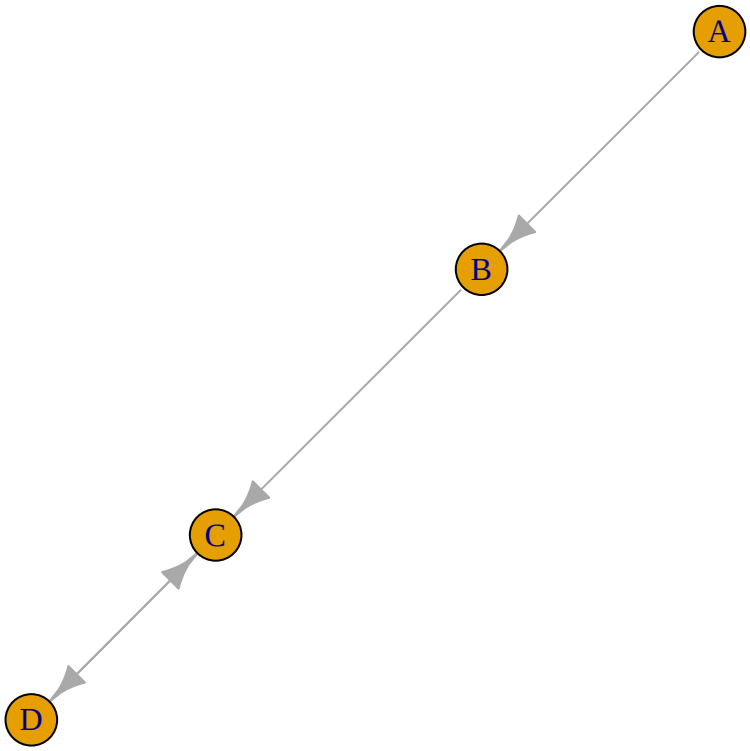
```
g2=graph_from_literal(A:B:C---D:e:f,g--h)
plot(g2)
```



有向图

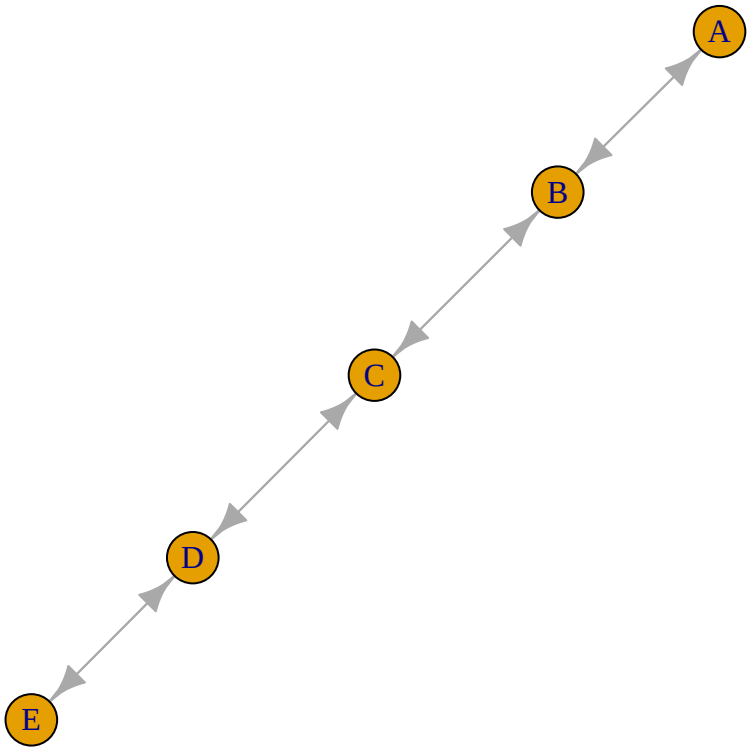
有向图使用+作为箭头，如

```
g3=graph_from_literal(A--+B--+C+-+D)
plot(g3)
```

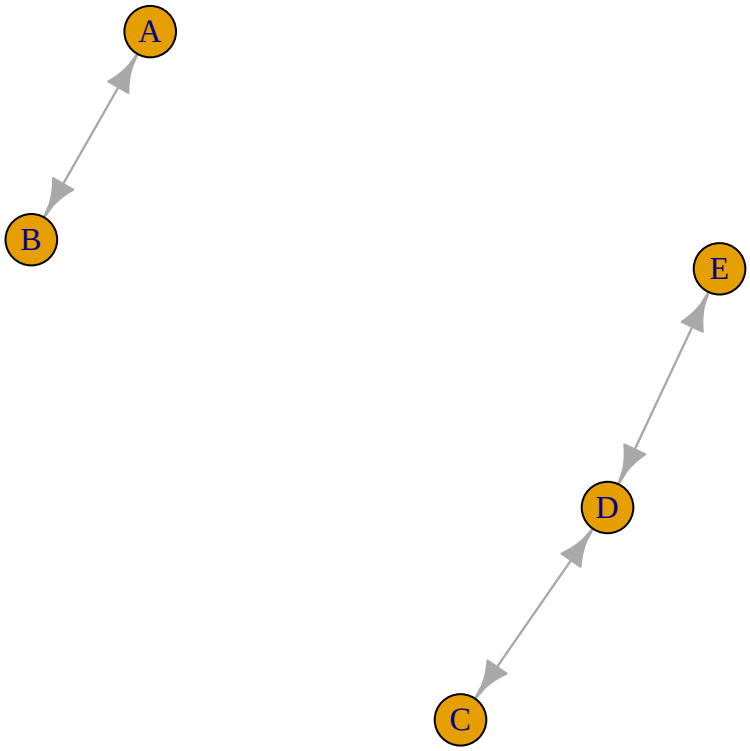


有向图中，如果顶点之间没有箭头，则表示这两个点不相连，双向的箭头可以使用一个或者两个+代替

```
g4=graph_from_literal( A +-+ B +---+ C ++ D + E)
plot(g4)
```




```
#在有向图中B C之间是不相连的
g5=graph_from_literal( A +-+ B --- C ++ D + E)
plot(g5)
```



图中的顶点，可以用引号来给出一个字符串的名字如

```
g5=graph_from_literal( “vertex A” +-+ “Vertex B” + “Vertex C” )
```

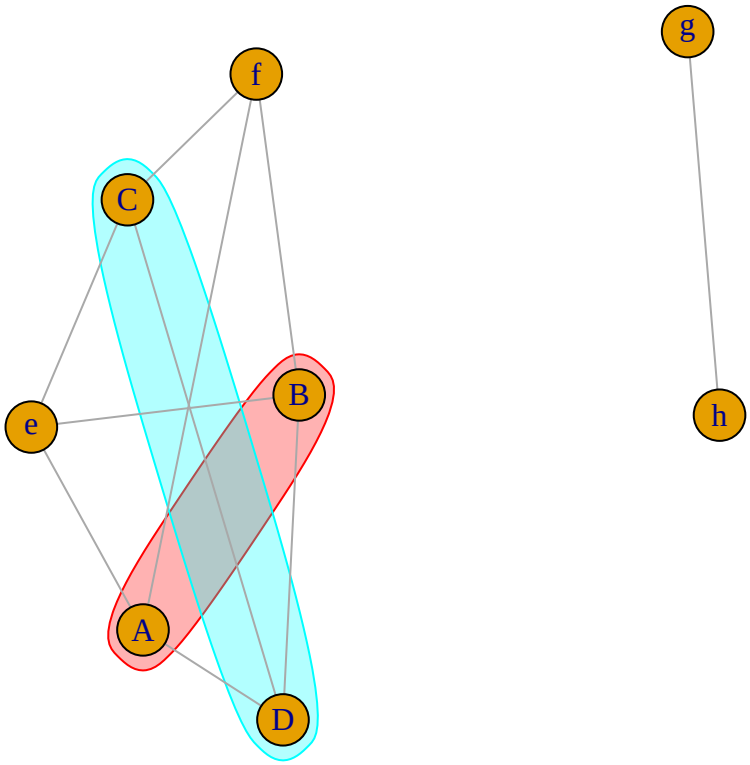
绘制图

前面使用了缺省的绘制函数，这里具体的看看这个绘制函数中都有一些什么参数:

```
## S3 method for class 'igraph'
plot(x, axes = FALSE, add = FALSE, xlim = c(-1, 1),
  ylim = c(-1, 1), mark.groups = list(), mark.shape = 1/2,
  mark.col = rainbow(length(mark.groups), alpha = 0.3),
  mark.border = rainbow(length(mark.groups), alpha = 1), mark.expand = 15,
  ...)
```

其中mark.groups是一个列表，用来给顶点分组，每组会被高亮显示，而后面的参数都是关于这个分组的。如下所示

```
g2=graph_from_literal(A:B:C---D:e:f,g--h)
plot(g2,mark.groups=list(1:2,3:4))
```



从这个函数看，看起来不能指定每个顶点的形状和颜色。但实际上后面的...参数提供了指定这些参数的可能性。这里是所有...中可以使用的参数。

要指定形状，使用vertex.shape=c([shape]) ， 支持的形状有“circle”， “square”， “csquare”， “rectangle”，“crectangle”， “vrectangle”， “pie” (see vertex.shape.pie)， ‘sphere’ ，and “none” 。

要指定顶点的标签，使用label参数，缺省的情况下使用顶点的ID(也就是前面定义图时指定的名字)。

指定顶点标签的字体，使用label.family，label.font

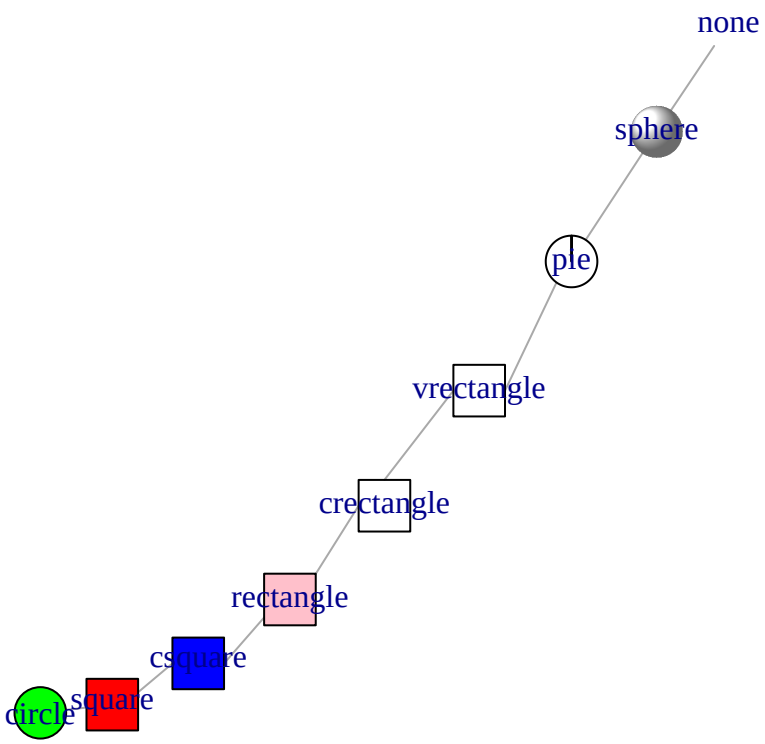
指定顶点标签的颜色，使用label.color

指定顶点的填充的颜色，使用vertex.color

下面我们来画一个有些属性变化的流程图

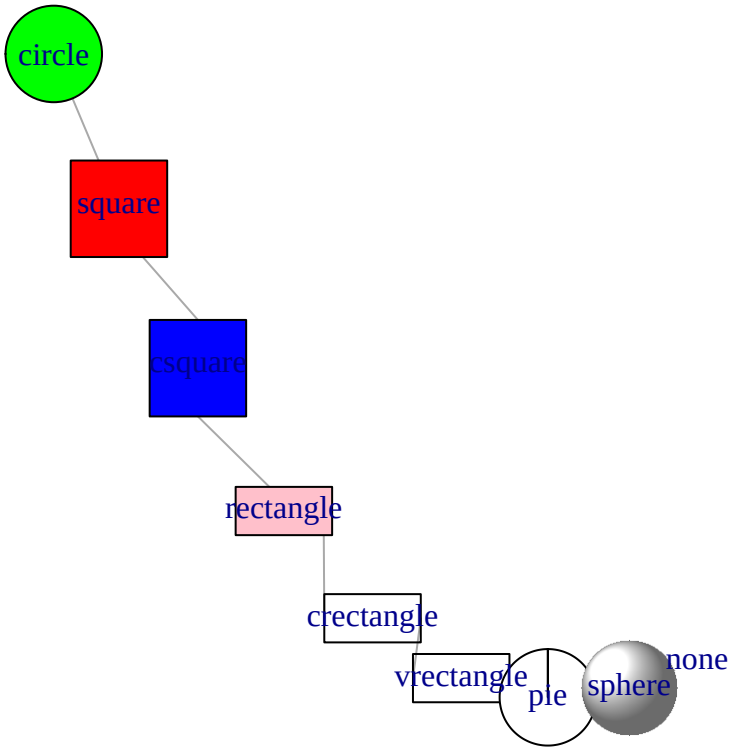
```
g1=graph_from_literal(A--B--C--D--e--f-g-h-i)
shape=c("circle","square","csquare","rectangle","crectangle","vrectangle","pie","sphere","none")
color=c("green","red","blue","pink")
label=c("circle","square","csquare","rectangle","crectangle","vrectangle","pie","sphere","none")

plot(g1,vertex.shape=shape,vertex.color=color,vertex.label=label)
```



这里看到有些文字已经超出了图形的限制，可以使用vertex.size来指定顶点的大小，其缺省值为15，我们把它放大一倍到30,看看效果:

```
plot(g1,vertex.shape=shape,vertex.color=color,vertex.label=label,vertex.size=30)
```



可以看到某些图形发生了重叠，这里的绘制方法应该需要一些改进。

更多创建图的方法

graph_from_edgelist

这个函数可以从一个边矩阵来创建一个图，其函数原型为

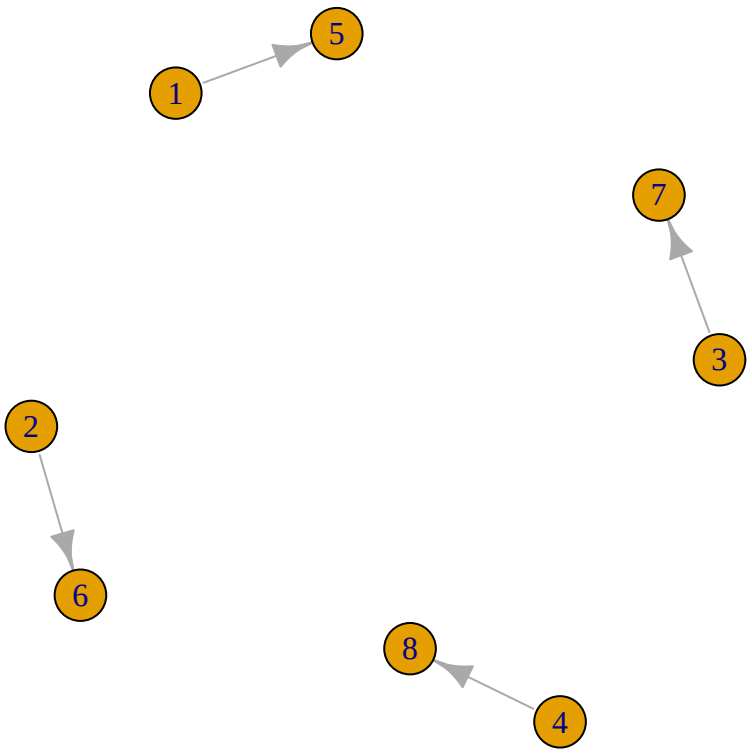
```
graph_from_edgelist(el, directed = TRUE)
```

其中

- el ： 两列的矩阵，数字的或者是字符串
- directed: 是否创建有向图

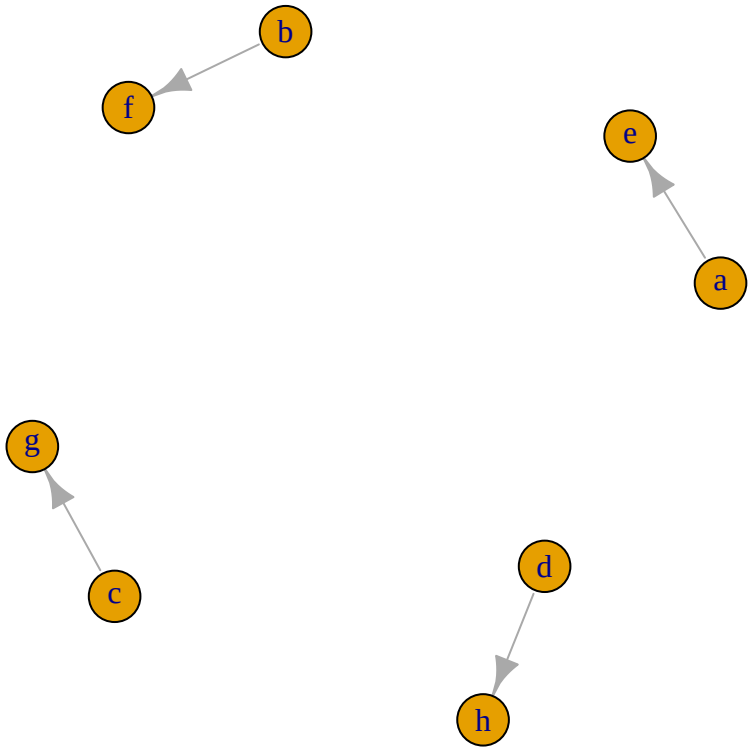
看下面这个实例:

```
mm=matrix(1:8,ncol=2)
g=graph_from_edgelist(mm)
plot(g)
```



或者，其中的矩阵也可以是字符形式的，如下所示

```
mm=matrix(letters[1:8],ncol=2)
g=graph_from_edgelist(mm)
plot(g)
```



graph_from_adj_list

这个函数从图的邻接点来建立图，该函数的原型如下：

```
graph_from_adj_list(adjlist, mode = c("out", "in", "all", "total"),
  duplicate = TRUE)
```

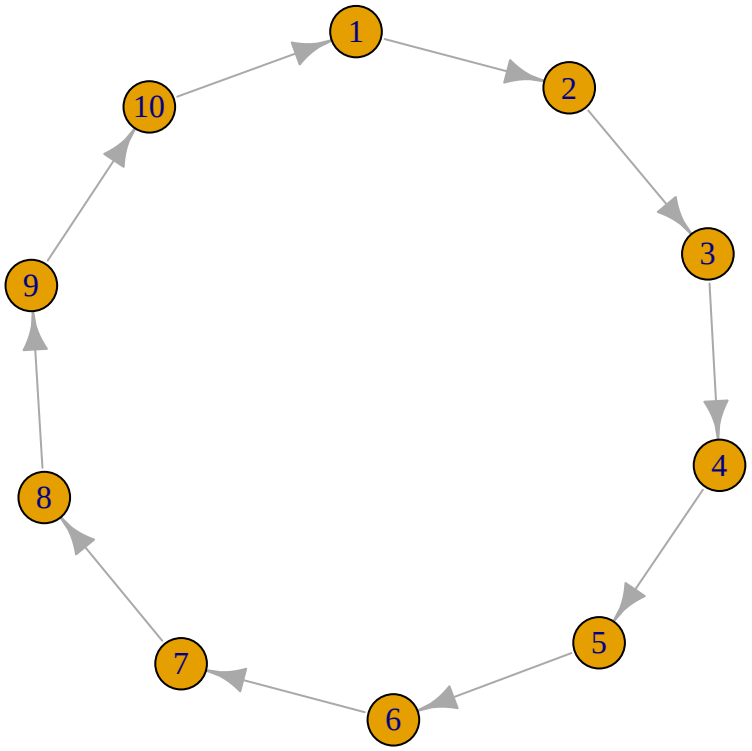
其中

- adjlist: 邻接列表，给出与每个顶点相邻的点
- mode: 说明邻接点的性质，无向图还是有向图，对有向图而言，是入点还是出点
- duplicate: 在out和in模式下无效

看一个实例：

```
g = make_ring(10,dir=T)
a = as_adj_list(g,mode="out")
#看看a是什么样子
a
#> [[1]]
#> + 1/10 vertex:
```

```
#> [1] 2
#>
#> [[2]]
#> + 1/10 vertex:
#> [1] 3
#>
#> [[3]]
#> + 1/10 vertex:
#> [1] 4
#>
#> [[4]]
#> + 1/10 vertex:
#> [1] 5
#>
#> [[5]]
#> + 1/10 vertex:
#> [1] 6
#>
#> [[6]]
#> + 1/10 vertex:
#> [1] 7
#>
#> [[7]]
#> + 1/10 vertex:
#> [1] 8
#>
#> [[8]]
#> + 1/10 vertex:
#> [1] 9
#>
#> [[9]]
#> + 1/10 vertex:
#> [1] 10
#>
#> [[10]]
#> + 1/10 vertex:
#> [1] 1
g2=graph_from_adj_list(a)
plot(g2)
```



好吧，基本上，igraph提供了一些基本的功能，具体的diagram可以把图形画得比较漂亮。

因此，使用igraph来生成矩阵，利用diagram来画图应该是个不错的选择。

Chapter 8

R 中用diagram包绘制流程图

此处关注的是R中绘制流程图的方法，而不是统计图的绘制。

关于流程图的绘制中，我找到了几个相关的包:

shape

其中最基础的包应该是shape，由包名可猜到这个包应该提供的是一些绘制基本图形的工具。可以利用help(package="shape")看看这个包提供了哪些函数，其中包含了箭头，圆，圆柱，矩形，多边形的绘制函数以及两个调色版函数。一般情况下，我们不会直接调用这里的函数。

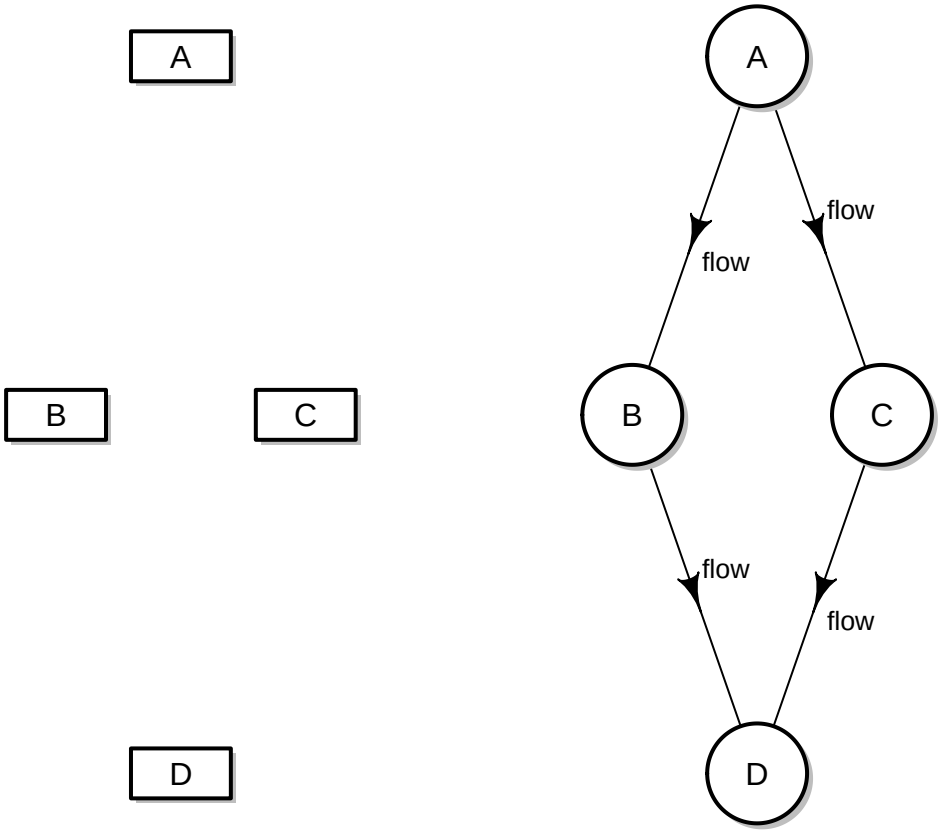
diagram

diagram包是建立在shape之上的一个包。这是一个绘制流程图，电路图，以及进行图可视化（基于转移矩阵）的工具。我们最关心的是其中的核心函数plotmat

它绘制一个方阵，代表的是一个图的链接关系（列->行）。使用help(plotmat,package='diagram')可以看到该函数的详细说明。下面是几个实例

```
library(diagram)
par(mar = c(1, 1, 1, 1), mfrow = c(1, 2))
names <- c("A", "B", "C", "D")
M <- matrix(nrow = 4, ncol = 4, byrow = TRUE, data = 0)
plotmat(M, pos = c(1, 2, 1), name = names, lwd = 1,
         box.lwd = 2, cex.txt = 0.8, box.size = 0.1,
         box.type = "square", box.prop = 0.5)


M[2, 1] <- M[3, 1] <- M[4, 2] <- M[4, 3] <- "flow"
plotmat(M, pos = c(1, 2, 1), curve = 0, name = names, lwd = 1,
         box.lwd = 2, cex.txt = 0.8, box.type = "circle", box.prop = 1.0)
```



简单的说明下，

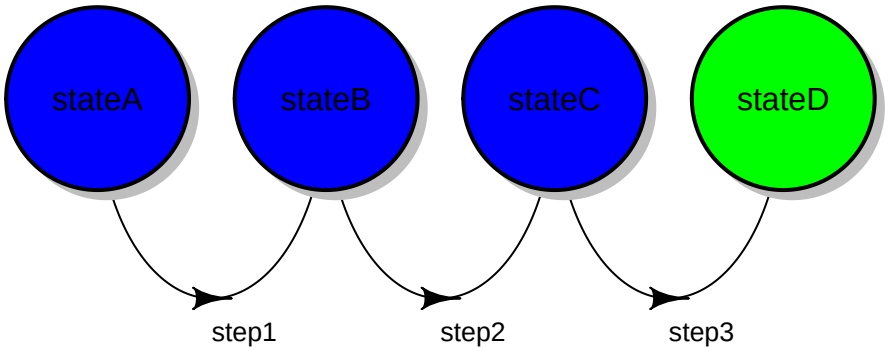
- 第一个参数为一个方阵，表明图的连接关系，矩阵的行数表示图的顶点的个数，
- 第二个参数pos指定了图的形状，上面的c(1,2,1)表示第一行1个点，第二行2个点，第三行1个点。如果要把所有的点画在一行，使用一个数字pos=c(4)就可以了。
- name参数指定每个顶点的名字。
- 边界的名字为矩阵元素的值。
- box.type为顶点的形状（支持rect, ellipse, diamond, round, hexa,multi），
- 还有box.col表示每个顶点的填充颜色。
- curve参数指定了连线的形状，缺省为直线0，1为弧线

再如下例:

```
M=matrix(rep(0,16),nrow=4)
M[2,1] = "step1"
M[3,2] = "step2"
M[4,3] = "step3"

names=c("stateA","stateB","stateC","stateD")
cols=c("blue","blue","blue","green")
plotmat(M, pos = c(4), curve = 1, name = names, lwd = 1,
```

```
        box.lwd = 2, cex.txt = 0.8, box.type = "circle", box.prop = 1.0,box.col=cols)
```

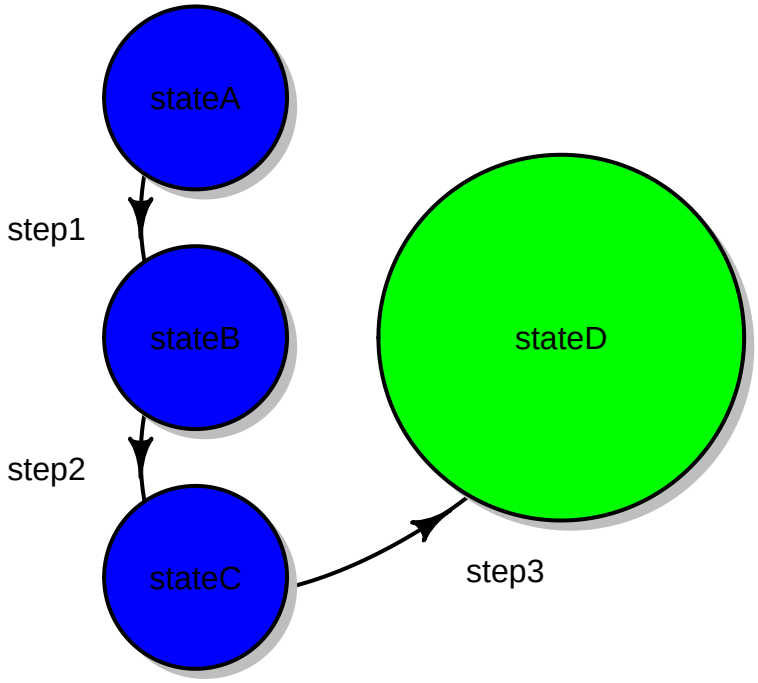


另外，如果要精细的控制每个图元的位置，可以使用矩阵形式的pos参数，此时这个矩阵是nx2,n为顶点数，每行两个数字，分别指定x和y的值。这里x和y的值在0~1之间。也就是说整张图的大小是1x1的。

每个图元的大小可以使用box.size来控制，对于上面的这个图，我们减小圆的大小,同时让三个圆在左边，一个圆在右边，并且右边的圆比左边的大，可以这样

```
pos=cbind(c(0.3,0.3,0.3,0.7),
      c(0.8,0.5,0.2,0.5))
boxsize=c(0.1,0.1,0.1,0.2)

plotmat(M, pos = pos, name = names,
         box.type = "circle", box.prop = 1.0,box.col=cols,
         box.size=boxsize)
```



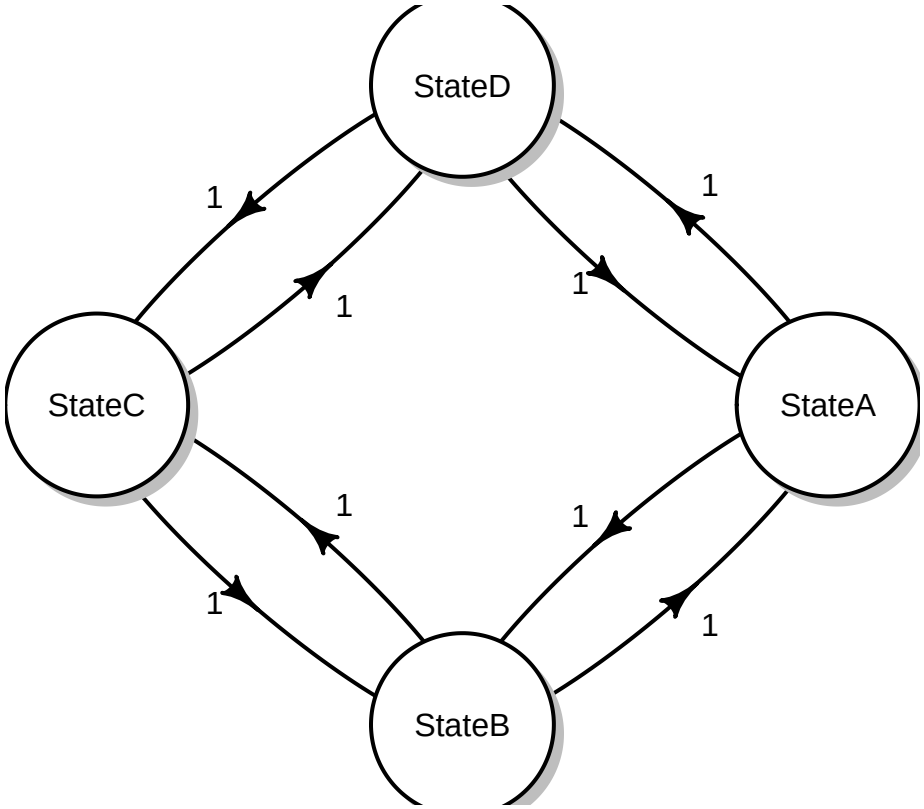
利用igraph包来创建连接矩阵

前面看出，利用diagram包来画图是很方便的，但是要手工收入连接矩阵是一件很麻烦的事情，如果如变得复杂，这件事情用手来做也变得不可行了。前面看到igraph可以生成连接矩阵，我们可利用igraph生成的连接矩阵，然后利用diagram来绘制。如下所示:

```
library(igraph)
g1=graph_from_literal(StateA---StateB---StateC---StateD,StateA---StateD)
#m1是一个s4类
m1=as_adj(g1)
m1
#> 4 x 4 sparse Matrix of class "dgCMatrix"
#>        StateA StateB StateC StateD
#> StateA      .      1      .      1
#> StateB      1      .      1      .
#> StateC      .      1      .      1
#> StateD      1      .      1      .
#转化为矩阵
```

```
m11=as.matrix(m1)

#plotmat is in diagram package
plotmat(m11)
```



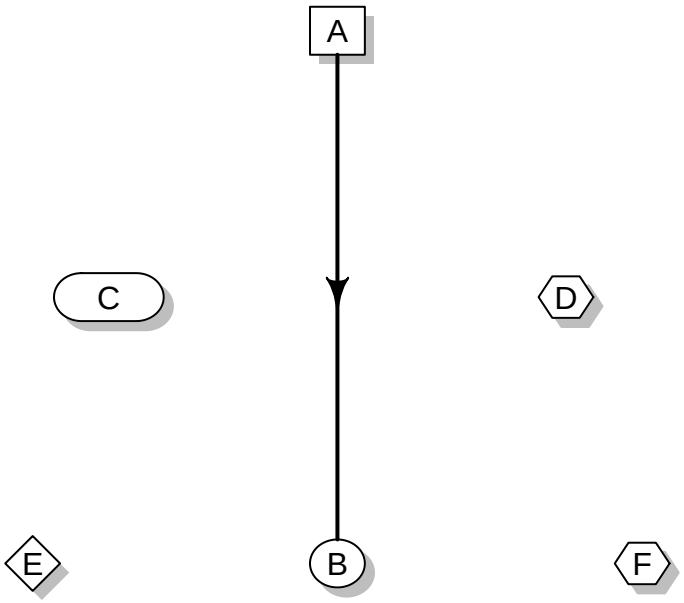
这里可以看到，由igraph创建的矩阵，可以由diagram绘制出来。所以由igraph包来创建图是很方便的。这是一个推荐的绘图方式。

直接绘图

前面讨论的是利用连接矩阵绘图，除了这个方法以外，还可以使用函数来绘制图形。基本过程是

```
openplotmat()  #创建一个空白的图
#假设你要绘制6个点，设定6个点的位置
pos <- coordinates(c(1,2,3))
#在各个顶点上绘制图形
textrect(pos[1,],radx=0.03,lab="A")
textellipse(pos[5,],radx=0.03,lab="B")
```

```
textround(pos[2,],radx=0.03,lab="C")
texthexa(pos[3,],radx=0.03,lab="D")
textdiamond(pos[4,],radx=0.03,lab="E")
textmulti(pos[6,],radx=0.03,lab="F")
#连接各顶点
straightarrow(from=pos[1,]-c(0,0.03),to=pos[5,]+c(0,0.03))
```



pathdiagram

另外还有一个包叫pathdiagram，这个包的作者说他使用diagram时，有些需求无法满足，因而开发了pathdiagram包，这个包应该比较简单。由于我暂时还用不到它，所以只是列在这里，以后需要的时候再来看。

gRbase

前面的diagram绘制流程图表现不错，但是直接用手写连接矩阵是意见痛苦的事情，不过包gRbase可以简化这件事情。如下所示，可见这样简单的建立一个连接矩阵：（此包依赖于两个不是CRAN上的包RBGL和graph，因此安装失败，以下代码不能

运行)

```
library(gRbase)
M=ug(~a:b + b:c:d + d:e + a:e, result="matrix")
plotmat(M)
```

gRbase还可以用来创建有向图，并可以直接绘图。

```
M=dag(~a:b + b:c:d + d:e + a:e)
plot(M)
```

igraph

igraph包是一个专门用来处理图的包。其中有很多用来创建和处理图的包。这个包可以很方便的创建上面的图以及连接矩阵。详细可参见这篇文章

Chapter 9

通过Rmarkdown包调用knitr和pandoc

通过rmarkdown来调用pandoc

rmarkdown是一个R包，其目的是为了在R中方便的生成各种文档。这个包除了处理了一些R语言的代码外，后台调用了pandoc来实现各种页面的生成。rmarkdown缺省的会为你指定一些参数，模板等。但是这些参数不一定适合你的需要，所以需要了解如何在rmakrdown中修改这些参数。在R中调用rmarkdown的方法是rmarkdown::render("input.rmd")，缺省会生成html。

rmarkdown中指定html的生成参数

在rmarkdown中做这件事情的方法是，修改_output.yaml，添加md_extensions。　　这里_output.yaml是md目录下的一个文件，用来为rmarkdown指定生成html的格式，它对当前目录下的所有md文件有效。它实际上是指定了rmarkdown包中函数html_document的参数。而这些参数将会传递到对pandoc的调用中去。如下是一个例子:

```
html_document:
        self_contained: false
        highlight: pygments
        toc: true
        template: ./template/frag.html
        md_extensions: -ascii_identifiers
```

关于这个html_document更多的参数，可以参考它源码rmarkdown on git 中的html_document.R文件。下面是html_document函数的参数列表，每个参数都可以在上面的yaml文件中修改。

```
html_document <- function(toc = FALSE,
                          toc_depth = 3,
                          toc_float = FALSE,
                          number_sections = FALSE,
                          fig_width = 7,
                          fig_height = 5,
                          fig_retina = if (!fig_caption) 2,
                          fig_caption = FALSE,
                          dev = 'png',
                          smart = TRUE,
                          self_contained = TRUE,
                          theme = "default",
                          highlight = "default",
                          mathjax = "default",
                          template = "default",
                          extra_dependencies = NULL,
```

```
                        css = NULL,
                        includes = NULL,
                        keep_md = FALSE,
                        lib_dir = NULL,
                        md_extensions = NULL,
                        pandoc_args = NULL,
                        ...) {
```

前面看这个包rmarkdown的内容时，感觉东西很多，后来看到了它的手册。觉得学习它的手册是比较好的方法。从最上层函数开始一层层的往下看，基本上这个包的结构就清楚了。

这个包将R Markdown文件转换成各种格式，包括HTML，Word，PDF以及PPT。其最上层的调用函数为render。

最高级别的函数render

以下是一些调用示例，这里的render是该包的最高级的函数调用。

```
render("input.Rmd", html_document())
render("input.Rmd", pdf_document())

render("input.Rmd", html_document(toc = TRUE))
render("input.Rmd", pdf_document(latex.engine = "lualatex"))
render("input.Rmd", beamer_presentation(incremental = TRUE))

render("input.Rmd", pdf_document(toc = TRUE, "--listings"))

#也可以处理md文件，这样就跳过了knitr
render("input.md", pdf_document(toc = TRUE, "--listings"))
```

由此可见，可以把输入变成html，pdf，beamer格式，其中每种格式也可以设置额外的参数。这里需要关注的是如何设置各种参数。以及如何设置knitr和pandoc的参数。

第二个级别的函数 [html|pdf|*]_document

这个级别的函数会被render调用，为各种类型的文档定义缺省的render参数。

具体的，上面的html_document本身也是一个函数，该函数的定义如下:

```
> html_document
function (toc = FALSE, toc_depth = 3, toc_float = FALSE, number_sections = FALSE,
    fig_width = 7, fig_height = 5, fig_retina = if (!fig_caption) 2,
    fig_caption = FALSE, dev = "png", code_folding = c("none",
        "show", "hide"), smart = TRUE, self_contained = TRUE,
    theme = "default", highlight = "default", mathjax = "default",
    template = "default", extra_dependencies = NULL, css = NULL,
    includes = NULL, keep_md = FALSE, lib_dir = NULL, md_extensions = NULL,
    pandoc_args = NULL, ...)

#其返回值部分为:
    output_format(knitr = knitr_options_html(fig_width, fig_height,
        fig_retina, keep_md, dev), pandoc = pandoc_options(to = "html",
        from = from_rmarkdown(fig_caption, md_extensions), args = args),
        keep_md = keep_md, clean_supporting = self_contained,
        pre_processor = pre_processor, base_format = html_document_base(smart = smart,
            theme = theme, self_contained = self_contained, lib_dir = lib_dir,
```

```
            mathjax = mathjax, template = template, pandoc_args = pandoc_args,
            extra_dependencies = extra_dependencies, ...))
```

也即，该函数返回一个output_format函数的返回值。其中调用了knitr_options_* 和pandoc_options(...)

另外，输出为pdf时，pdf_document()函数的定义是这样的

```
> pdf_document
function (toc = FALSE, toc_depth = 2, number_sections = FALSE,
    fig_width = 6.5, fig_height = 4.5, fig_crop = TRUE, fig_caption = FALSE,
    dev = "pdf", highlight = "default", template = "default",
    keep_tex = FALSE, latex_engine = "pdflatex", citation_package = c("none",
        "natbib", "biblatex"), includes = NULL, md_extensions = NULL,
    pandoc_args = NULL)

#其返回值部分为
    output_format(knitr = knitr_options_pdf(fig_width, fig_height,
        fig_crop, dev), pandoc = pandoc_options(to = "latex",
        from = from_rmarkdown(fig_caption, md_extensions), args = args,
        latex_engine = latex_engine, keep_tex = keep_tex), clean_supporting = !keep_tex,
        pre_processor = pre_processor, intermediates_generator = intermediates_generator)
```

第三个级别的函数 knitr_options_*和pandoc_options

这个级别的函数被第二级函数调用，如上面的返回值部分，用来指定knitr和pandoc的参数。看看knitr_options_*的实现

```
> knitr_options_html
function (fig_width, fig_height, fig_retina, keep_md, dev = "png")
{
    opts_chunk <- list(dev = dev, dpi = 96, fig.width = fig_width,
        fig.height = fig_height, fig.retina = fig_retina)
    if (keep_md)
        opts_chunk$fig.retina <- NULL
    knitr_options(opts_chunk = opts_chunk)
}

> knitr_options_pdf
function (fig_width, fig_height, fig_crop, dev = "pdf")
{
    opts_knit <- NULL
    opts_chunk <- list(dev = dev, fig.width = fig_width, fig.height = fig_height)
    if (dev == "pdf") {
        if (utils::packageVersion("knitr") >= "1.5.31") {
            opts_chunk$dev.args <- list(pdf = list(useDingbats = FALSE))
        }
        else grDevices::pdf.options(useDingbats = FALSE)
    }
    knit_hooks <- NULL
    crop <- fig_crop && !is_windows() && nzchar(find_program("pdfcrop"))
    if (crop) {
        knit_hooks = list(crop = knitr::hook_pdfcrop)
        opts_chunk$crop = TRUE
    }
    knitr_options(opts_knit = opts_knit, opts_chunk = opts_chunk,
        knit_hooks = knit_hooks)
}
```

```
#以上两个函数会调用knitr_options函数，如下所示

> knitr_options
function (opts_knit = NULL, opts_chunk = NULL, knit_hooks = NULL,
    opts_template = NULL)
{
    list(opts_knit = opts_knit, opts_chunk = opts_chunk, knit_hooks = knit_hooks,
        opts_template = opts_template)
}
```

knitr_options_*设置了相关的knitr参数以后，最后调用knitr_options，而这个函数分别指定了knitr的各级别参数，第一个是包级别的参数，第二个是块级别的参数，第三个是指定hook，最后是模板。

该函数返回一个list，不会对最终的文档生成产生影响，应该放在output_format的参数中被调用，否则这些设置不会生效。

pandoc_options函数的定义比较简单，如下:

```
> pandoc_options
function (to, from = rmarkdown_format(), args = NULL, keep_tex = FALSE,
    latex_engine = c("pdflatex", "lualatex", "xelatex"), ext = NULL)
{
    list(to = to, from = from, args = args, keep_tex = keep_tex,
        latex_engine = match.arg(latex_engine), ext = ext)
}
```

关于output_format函数

这是最终返回给render的值。指定了最终的参数。只有对这个output_format函数产生了影响的函数才会对最终的生成结果产生影响。

看看这个函数:

```
> output_format
function (knitr, pandoc, keep_md = FALSE, clean_supporting = TRUE,
    pre_processor = NULL, intermediates_generator = NULL, post_processor = NULL,
    base_format = NULL)
{
    format <- structure(list(knitr = knitr, pandoc = pandoc,
        keep_md = keep_md, clean_supporting = clean_supporting &&
            !keep_md, pre_processor = pre_processor, intermediates_generator = intermediates_generator,
        post_processor = post_processor), class = "rmarkdown_output_format")
    if (!is.null(base_format))
        merge_output_formats(base_format, format)
    else format
}
```

由此可见，传入render函数的是一个“rmarkdown_output_format”类。因此可以使用自己的函数来替换这个html_document()。

如何修改knitr和pandoc的参数

有了上面的知识，如果你需要修改knitr或者pandoc的参数，可以直接调用html_document()或者pdf_document()之后，修改这个函数的返回值。然后再将这个修改过的值传给render函数。因此靠谱的做法是先把这个返回值的缺省值打印出来看看

```
str(rmarkdown::html_document())
#> List of 7
#>  $ knitr                 :List of 4
#>   ..$ opts_knit    : NULL
#>   ..$ opts_chunk   :List of 5
#>   .. ..$ dev       : chr "png"
#>   .. ..$ dpi       : num 96
#>   .. ..$ fig.width : num 7
#>   .. ..$ fig.height: num 5
#>   .. ..$ fig.retina: num 2
#>   ..$ knit_hooks   : NULL
#>   ..$ opts_template: NULL
#>  $ pandoc                :List of 6
#>   ..$ to          : chr "html"
#>   ..$ from        : chr "markdown+autolink_bare_uris+ascii_identifiers+tex_math_single_backslash-implicit_figures"
#>   ..$ args        : chr [1:8] "--smart" "--email-obfuscation" "none" "--self-contained" ...
#>   ..$ keep_tex    : logi FALSE
#>   ..$ latex_engine: chr "pdflatex"
#>   ..$ ext         : NULL
#>  $ keep_md               : logi FALSE
#>  $ clean_supporting      : logi TRUE
#>  $ pre_processor         :function (...)
#>  $ intermediates_generator:function (original_input, encoding, intermediates_dir)
#>  $ post_processor        :function (metadata, input_file, output_file, clean, verbose)
#>  - attr(*, "class")= chr "rmarkdown_output_format"
str(rmarkdown::pdf_document())
#> List of 7
#>  $ knitr                 :List of 4
#>   ..$ opts_knit    : NULL
#>   ..$ opts_chunk   :List of 5
#>   .. ..$ dev       : chr "pdf"
#>   .. ..$ fig.width : num 6.5
#>   .. ..$ fig.height: num 4.5
#>   .. ..$ dev.args  :List of 1
#>   .. .. ..$ pdf:List of 1
#>   .. .. .. ..$ useDingbats: logi FALSE
#>   .. ..$ crop      : logi TRUE
#>   ..$ knit_hooks   :List of 1
#>   .. ..$ crop:function (before, options, envir)
#>   ..$ opts_template: NULL
#>  $ pandoc                :List of 6
#>   ..$ to          : chr "latex"
#>   ..$ from        : chr "markdown+autolink_bare_uris+ascii_identifiers+tex_math_single_backslash-implicit_figures"
#>   ..$ args        : chr [1:8] "--template" "/home/xuyang/R/x86_64-pc-linux-gnu-library/3.2/rmarkdown/rmd/latex/default
#>   ..$ keep_tex    : logi FALSE
#>   ..$ latex_engine: chr "pdflatex"
#>   ..$ ext         : NULL
#>  $ keep_md               : logi FALSE
#>  $ clean_supporting      : logi TRUE
#>  $ pre_processor         :function (metadata, input_file, runtime, knit_meta, files_dir,
#>     output_dir)
#>  $ intermediates_generator:function (original_input, encoding, intermediates_dir)
#>  $ post_processor        : NULL
#>  - attr(*, "class")= chr "rmarkdown_output_format"
```

修改的实例

Hadley 的bookdown中提供了一个修改定制rmarkdown输出格式的实例,贴在这里仅供参考, 源代码链接

```
#' @export
html_chapter <- function(raw = FALSE, toc = NULL, code_width = 80) {
  base <- rmarkdown::html_document(
    self_contained = FALSE,
    lib_dir = "www",
    template = if (raw) system.file("raw-html.html", package = "bookdown") else system.file("chapter-html.html", package
    mathjax = if (raw) NULL else "default"
  )
  # Remove --section-divs option
  base$pandoc$args <- setdiff(base$pandoc$args, "--section-divs")
  base$pandoc$from <- markdown_style



  if (!is.null(toc)) {
    old_p <- base$pre_processor
    base$pre_processor <- function(yaml_front_matter, utf8_input, runtime,
                                   knit_meta, files_dir, output_dir) {
      update_links(utf8_input, toc)
      old_p(yaml_front_matter, utf8_input, runtime,
        knit_meta, files_dir, output_dir)
    }
  }

  base
}

#' @export
tex_chapter <- function(chapter = NULL,
                        latex_engine = c("xelatex", "pdflatex", "lualatex"),
                        code_width = 65) {
  options(digits = 3)
  set.seed(1014)

  latex_engine <- match.arg(latex_engine)
  rmarkdown::output_format(
    knitr_opts("latex", chapter),
    rmarkdown::pandoc_options(
      to = "latex",
      from = markdown_style,
      ext = ".tex",
      #args = c("--chapters", pandoc_latex_engine_args(latex_engine))
      args = c("--chapters")
    ),
    clean_supporting = FALSE
  )
}

markdown_style <- paste0(
  "markdown",
  "+autolink_bare_uris",
  "-auto_identifiers",
```

```
  "+tex_math_single_backslash",
  "-implicit_figures"
)

knitr_opts <- function(type = c("html", "latex"), chapter, code_width = 65) {
  type <- match.arg(type)

  pkg <- list(
    width = code_width
  )

  chunk <- list(
    comment = "#>",
    collapse = TRUE,
    cache.path = paste0("_cache/", chapter, "/"),
    cache = TRUE,
    fig.path = paste0("_figures/", chapter, "/"),
    fig.width = 4,
    fig.height = 4,
    fig.retina = NULL,
    dev = if (type == "html") "png" else "pdf",
    dpi = if (type == "html") 96 else 300
  )

  hooks <- list(
    #plot = if (type == "latex") html_plot(),
    small_mar = function(before, options, envir) {
      if (before)
        par(mar = c(4.1, 4.1, 0.5, 0.5))
    }
  )

  rmarkdown::knitr_options(pkg, chunk, hooks)
}
```

根据以上这些修改，现在假定我们要修改fig.path， 这个参数指定了R生成图片的位置和名字。即，如果你指定fig.path= ‘./rfigures/p123-’ ，那么生成的图片位置就是./rfigures，而图片的名字为p123-chunk-\d-\d.png 其中的\d是当前代码块的编号以及块中图片的编号。

现在我们为了图片的名字不重复，需要根据输入文件的名字来命名输出图片的名字，即输入文件名为p\d+.rmd ，其对应的图片前缀为rfigures/[文件名]-,这个设置应该在调用rmarkdown的时候设置好。即

```
rmarkdown::render('p123.rmd',rwp::html_doc('p123.rmd'))
```

其中的html_doc()中，需要根据输入文件名来设置fig.path的路径。实现的方法借用了上面的代码，具体的为:

```
html_doc<-function(inputfile)
{
  out=rmarkdown::html_document(
        self_contained = FALSE,
        highlight= 'kate',
        toc=TRUE,
        template= './template/frag.html',
        md_extensions='-ascii_identifiers',
        lib_dir='./lib'
        )
  out$knitr=knitr_opts("html", FALSE,inputfile=inputfile)
  out
```

```
}

knitr_opts <- function(type = c("html", "latex"), chapter, inputfile, code_width = 65) {
  type <- match.arg(type)

  pkg <- list(
    width = code_width
  )

  #p655.rmd ->p655
  ff = stringr::str_extract(inputfile,"p\\d+")

  chunk <- list(
    comment = "#>",
    collapse = TRUE,
    fig.path = paste0("rfigures/", ff, "-"),
    fig.width = 6,
    fig.height = 6,
    fig.retina = NULL,
    dev = if (type == "html") "png" else "pdf",
    dpi = if (type == "html") 96 else 300
  )

  hooks <- list(
    small_mar = function(before, options, envir) {
      if (before)
        par(mar = c(4.1, 4.1, 0.5, 0.5))
    }
  )

  rmarkdown::knitr_options(pkg, chunk, hooks)
}
```

用一个R包来封装对rmarkdown的格式设置

对于上面实现的这些函数，可以通过一个R包来进行封装，这样我们就可以用一句话很方便的调用这些函数，如下所示:

```
rmarkdown::render('input.rmd',rwp::html_doc('input.rmd'))
```

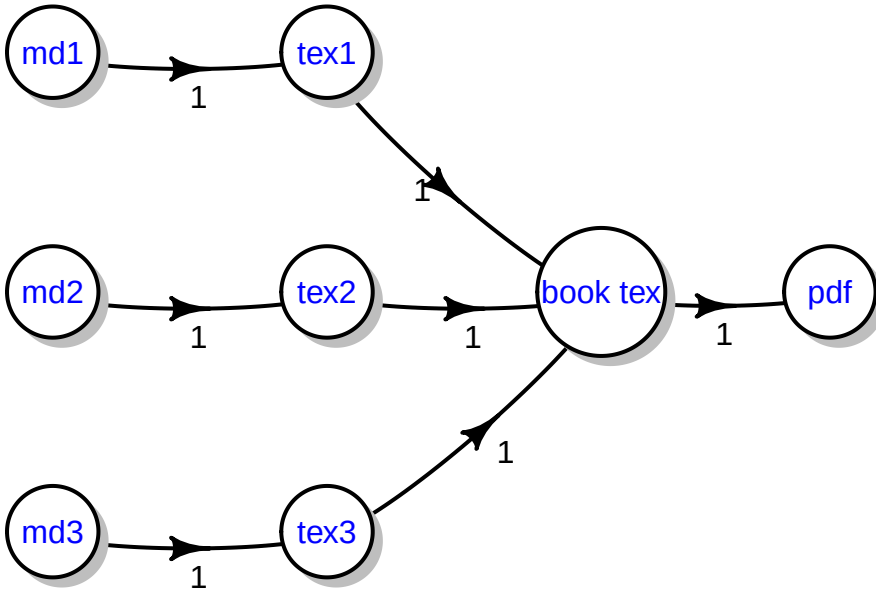
这样，很简单的就能实现这些格式控制。这是一个推荐的实现方式。而且用devtools来创建一个R包相当容易。devtools是Hadley开发的一个用于进行R包开发的工具，以下简单的说明下用devtools开发一个R包的流程:

1. devtools::create(“path/to/newpackage”)
2. 编码，添加roxygen注释（使用vim插件）
3. 修改DESCRIPTION，添加Imports域声明要导入的包
4. devtools::document() 生成注释和NAMESPACE。（在该包所在的目录运行这个）
5. devtools::build() 生成该包的压缩文件（即可安装的压缩包）
6. devtools::install() 安装刚刚生成的包
7. 测试，使用<包名>::func() 对新的包进行测试

Chapter 10

利用多篇markdown文件生成pdf书籍

这里想法是：一本书由多个章节组成，每章为一个md/rmd文件。现在要把这些md文件生成一本pdf书籍。要生成最后的pdf，就需要这个书的一个完整latex文件。做法是，先将每个md文件变成tex，然后使用include在最后的Latex文件中包含各章的tex文件,如下图所示:



因此首先要将每个md文件变成tex

利用pandoc为每个md生成tex文件

这个使用pandoc是比较容易做到的，只是需要注意这里的pandoc调用时必须含有--chapters参数，这样生成的文档以\chapter{}开始。下面的命令行可供参考:

`pandoc -f markdown -t latex --latex-engine=xelatex --chapters -o output.tex input.md`

该命令行生成的部分tex文件为:

\chapter{包的元数据}\label{ux5305ux7684ux5143ux6570ux636e}

\texttt{DESCRIPTION}(描述文件)的作用是存储包中重要的元数据。
当你第一次开发包时，会使用这个文件记录你的包运行时所需要的包。
然而，随着时间的流逝，当你开始与他人分享包时

...

通过一个书籍模板来生成pdf

在将md文件变成tex以后，最后需要将它们合成一个大的tex，用以生成最后的书籍。这里就需要有一个书籍的模板，用来将各章的内容包含进去，同时做一些全局的设置。最后生成书的目录等。

本文最后是一个生成书籍时使用的模板，实际使用中，只需要把xxtitle换成你的标题，xxchapter换成\include{chap1}就可以了。（注意不要带.tex扩展名，并且这些tex文件和这个模板文件在同一个目录下）

这个模板中，含有对中文的支持。使用到的字体是KaiTi_GB2312，FangSong_GB2312，SimHei。关于字体的安装可以参考这里： http://www.bagualu.net/wordpress/archives/5396

假设该模板文件为mybook.tex，那么生成pdf文件的命令行为

```
xelatex -interaction=batchmode mybook
```

如果没有错误，一个pdf书籍就生成了，但这时候的书籍是没有目录的。

生成目录

如果要生成目录，需要在书籍的模板中使用\tableofcontents，而且上面这个命令要运行两遍。也就是这样

```
xelatex -interaction=batchmode mybook
xelatex -interaction=batchmode mybook
```

\setcounter{tocdepth}{3}可用来指定目录的深度，在这个深度一下的标题将不会列入到目录中。

生成图表目录

如果要生成图表目录，需要对每个图表进行编号和命名。

- 对表命名的方式是在knitr::kable函数中使用caption参数。
- 对图命名的方式是在knitr的代码块中指定fig.cap参数

指定了这两个参数后，生成的图和表会被自动编号(latex中)。

然后在最后的书籍模板中使用\listoftables和\listoffigures就可以了，这两句话放置的地方是\tableofcontents之后。

书籍模板

下面是生成pdf书籍使用的模板:

```
\documentclass[]{book}
\usepackage{lmodern}
\usepackage{amssymb,amsmath}
\usepackage{ifxetex,ifluatex}
\usepackage{fixltx2e} % provides \textsubscript
\ifnum 0\ifxetex 1\fi\ifluatex 1\fi=0 % if pdftex
  \usepackage[T1]{fontenc}
  \usepackage[utf8]{inputenc}
\else % if luatex or xelatex
  \ifxetex
    \usepackage{mathspec}
    \usepackage{xltxtra,xunicode}
  \else
    \usepackage{fontspec}
  \fi
  \defaultfontfeatures{Mapping=tex-text,Scale=MatchLowercase}
  \newcommand{\euro}{  }
    \setmainfont{KaiTi_GB2312}
    \setsansfont{FangSong_GB2312}
    \setmonofont[Mapping=tex-ansi]{SimHei}
    \XeTeXlinebreaklocale "zh"
\renewcommand\labelitemi{\ensuremath{\bullet}}
\fi
% use upquote if available, for straight quotes in verbatim environments
\IfFileExists{upquote.sty}{\usepackage{upquote}}{}
% use microtype if available
\IfFileExists{microtype.sty}{%
\usepackage{microtype}
\UseMicrotypeSet[protrusion]{basicmath} % disable protrusion for tt fonts
}{}
\usepackage[a4paper, centering, scale=0.8]{geometry}
\usepackage{color}
\usepackage{fancyvrb}
\newcommand{\VerbBar}{|}
\newcommand{\VERB}{\Verb[commandchars=\\\{\}]}
\DefineVerbatimEnvironment{Highlighting}{Verbatim}{commandchars=\\\{\}}
% Add ',fontsize=\small' for more characters per line
\newenvironment{Shaded}{}{}
\newcommand{\KeywordTok}[1]{\textcolor[rgb]{0.00,0.44,0.13}{\textbf{{#1}}}}
\newcommand{\DataTypeTok}[1]{\textcolor[rgb]{0.56,0.13,0.00}{{#1}}}
\newcommand{\DecValTok}[1]{\textcolor[rgb]{0.25,0.63,0.44}{{#1}}}
\newcommand{\BaseNTok}[1]{\textcolor[rgb]{0.25,0.63,0.44}{{#1}}}
\newcommand{\FloatTok}[1]{\textcolor[rgb]{0.25,0.63,0.44}{{#1}}}
\newcommand{\CharTok}[1]{\textcolor[rgb]{0.25,0.44,0.63}{{#1}}}
\newcommand{\StringTok}[1]{\textcolor[rgb]{0.25,0.44,0.63}{{#1}}}
\newcommand{\CommentTok}[1]{\textcolor[rgb]{0.38,0.63,0.69}{\textit{{#1}}}}
\newcommand{\OtherTok}[1]{\textcolor[rgb]{0.00,0.44,0.13}{{#1}}}
\newcommand{\AlertTok}[1]{\textcolor[rgb]{1.00,0.00,0.00}{\textbf{{#1}}}}
\newcommand{\FunctionTok}[1]{\textcolor[rgb]{0.02,0.16,0.49}{{#1}}}
\newcommand{\RegionMarkerTok}[1]{{#1}}
\newcommand{\ErrorTok}[1]{\textcolor[rgb]{1.00,0.00,0.00}{\textbf{{#1}}}}
\newcommand{\NormalTok}[1]{{#1}}
```

```
\ifxetex
  \usepackage[setpagesize=false, % page size defined by xetex
              unicode=false, % unicode breaks when used with xetex
              xetex]{hyperref}
\else
  \usepackage[unicode=true]{hyperref}
\fi
\hypersetup{breaklinks=true,
            bookmarks=true,
            pdfauthor={},
            pdftitle={},
            colorlinks=true,
            citecolor=blue,
            urlcolor=blue,
            linkcolor=magenta,
            pdfborder={0 0 0}}
\urlstyle{same}  % don't use monospace font for urls
\setlength{\parindent}{0pt}
\setlength{\parskip}{6pt plus 2pt minus 1pt}
\setlength{\emergencystretch}{3em}  % prevent overfull lines
\setcounter{secnumdepth}{0}

\date{\today}

\title{xxtitle}
\author{江航}

\begin{document}

\frontmatter
\maketitle

{
\hypersetup{linkcolor=black}
\setcounter{tocdepth}{3}
\tableofcontents
}

\include{copyright}
\include{preface}

\mainmatter


xxchapter

\end{document}
```

书籍模板的生成方法

这个书籍模板不是一成不便的，当pandoc升级以后，模板就需要做相应的修改。下面是一个pandoc升级以后，修改书籍模板的方法:

1. 利用pandoc -D latex导出该版本的模板
2. 修改模板的中文字体

3. 利用新的模板生成一个tex文件。在生成的tex文件中，就已经没有各种模板变量了。以此为基础来构建我们的书籍模板
4. 对这个新生成的tex文件进行修改，修改的方法是
    1. 将第一行的\documentclass[]{article}修改为\documentclass[]{book}
    2. 添加页面设置\usepackage[a4paper, centering, scale=0.8]{geometry}
    3. 找到其中的\begin{document},然后利用我们书籍模板中的内容替换到文件中此行及以后的所有内容
    4. 在\begin{document}这行前添加date,title,author三个声明，如上面的模板所示。
    5. 将这个模板命名为template.tex，放到rwp包的inst目录下，替换原来的版本即可

Chapter 11

生成在线书籍

生成图书的网站

Chapter 12

pandoc filter

pandoc filter是什么 ？

pandoc fitler是一个可执行程序， 它从stdin读入，输出到stdout。filter的输入是pandoc内部语法树(AST)的json表示。filter在对这个语法树进行处理以后，把修改过的语法树输出到stdout，pandoc可以进一步的处理。简单的理解，pandoc的filter相当于是pandoc的一个插件。

filter可以通过参数来指定： 这个命令pandoc --filter ./caps.py -t latex指定了一个filter ./caps.py， 这个命令等价于pandoc -t json | ./caps.py | pandoc -f json -t latex

- 参考文档
  - 官方的说明文档 (有点慢)。
  - python 版的pandocfilters包 ，在github上，速度还可以

filter的输入

filter的输入，也就是pandoc的内部语法树，先看个简单的例子，有个大概的了解:

```
## my header

this is line *one*
some `codesample` here and a list below:

* item1

* item2
```

假设上面的文件名为aa.md，命令pandoc -t native aa.md的输出如下:

```
[Header 2 ("my-header",[],[]) [Str "my",Space,Str "header"]
,Para [Str "this",Space,Str "is",Space,Str "line",Space,Emph [Str "one"],SoftBreak,Str "some",Space,Code ("",[],[]) "code
,BulletList
 [[Para [Str "item1"]]
 ,[Para [Str "item2"]]]]
```

如果转成json格式是这样: (其中./tt的内容由pandoc aa.md -t json > tt 生成，下面用R处理一下是为了代码缩进，看起来更方便)

```
> library(rjson)
> jj=fromJSON(file="./tt")
> str(jj)
List of 2
```

```
$ :List of 1
 ..$ unMeta: list()
$ :List of 3
 ..$ :List of 2
 .. ..$ t: chr "Header"
 .. ..$ c:List of 3
 .. .. ..$ : num 2
 .. .. ..$ :List of 3
 .. .. .. ..$ : chr "my-header"
 .. .. .. ..$ : list()
 .. .. .. ..$ : list()
 .. .. ..$ :List of 3
 .. .. .. ..$ :List of 2
 .. .. .. .. ..$ t: chr "Str"
 .. .. .. .. ..$ c: chr "my"
 .. .. .. ..$ :List of 2
 .. .. .. .. ..$ t: chr "Space"
 .. .. .. .. ..$ c: list()
 .. .. .. ..$ :List of 2
 .. .. .. .. ..$ t: chr "Str"
 .. .. .. .. ..$ c: chr "header"
 ..$ :List of 2
 .. ..$ t: chr "Para"
 .. ..$ c:List of 21
 .. .. ..$ :List of 2
 .. .. .. ..$ t: chr "Str"
 .. .. .. ..$ c: chr "this"
 .. .. ..$ :List of 2
 .. .. .. ..$ t: chr "Space"
 .. .. .. ..$ c: list()
 .. .. ..$ :List of 2
 .. .. .. ..$ t: chr "Str"
 .. .. .. ..$ c: chr "is"
 .. .. ..$ :List of 2
 .. .. .. ..$ t: chr "Space"
 .. .. .. ..$ c: list()
 .. .. ..$ :List of 2
 .. .. .. ..$ t: chr "Str"
 .. .. .. ..$ c: chr "line"
 .. .. ..$ :List of 2
 .. .. .. ..$ t: chr "Space"
 .. .. .. ..$ c: list()
 .. .. ..$ :List of 2
 .. .. .. ..$ t: chr "Emph"
 .. .. .. ..$ c:List of 1
 .. .. .. .. ..$ :List of 2
 .. .. .. .. .. ..$ t: chr "Str"
 .. .. .. .. .. ..$ c: chr "one"
 .. .. ..$ :List of 2
 .. .. .. ..$ t: chr "SoftBreak"
 .. .. .. ..$ c: list()
 .. .. ..$ :List of 2
 .. .. .. ..$ t: chr "Str"
 .. .. .. ..$ c: chr "some"
 .. .. ..$ :List of 2
```

```
.. .. .. ..$ t: chr "Space"
.. .. .. ..$ c: list()
.. .. ..$ :List of 2
.. .. .. ..$ t: chr "Code"
.. .. .. ..$ c:List of 2
.. .. .. .. ..$ :List of 3
.. .. .. .. .. ..$ : chr ""
.. .. .. .. .. ..$ : list()
.. .. .. .. .. ..$ : list()
.. .. .. .. ..$ : chr "codesample"
.. .. ..$ :List of 2
.. .. .. ..$ t: chr "Space"
.. .. .. ..$ c: list()
.. .. ..$ :List of 2
.. .. .. ..$ t: chr "Str"
.. .. .. ..$ c: chr "here"
.. .. ..$ :List of 2
.. .. .. ..$ t: chr "Space"
.. .. .. ..$ c: list()
.. .. ..$ :List of 2
.. .. .. ..$ t: chr "Str"
.. .. .. ..$ c: chr "and"
.. .. ..$ :List of 2
.. .. .. ..$ t: chr "Space"
.. .. .. ..$ c: list()
.. .. ..$ :List of 2
.. .. .. ..$ t: chr "Str"
.. .. .. ..$ c: chr "a"
.. .. ..$ :List of 2
.. .. .. ..$ t: chr "Space"
.. .. .. ..$ c: list()
.. .. ..$ :List of 2
.. .. .. ..$ t: chr "Str"
.. .. .. ..$ c: chr "list"
.. .. ..$ :List of 2
.. .. .. ..$ t: chr "Space"
.. .. .. ..$ c: list()
.. .. ..$ :List of 2
.. .. .. ..$ t: chr "Str"
.. .. .. ..$ c: chr "below:"
..$ :List of 2
.. ..$ t: chr "BulletList"
.. ..$ c:List of 2
.. .. ..$ :List of 1
.. .. .. ..$ :List of 2
.. .. .. .. ..$ t: chr "Para"
.. .. .. .. ..$ c:List of 1
.. .. .. .. .. ..$ :List of 2
.. .. .. .. .. .. ..$ t: chr "Str"
.. .. .. .. .. .. ..$ c: chr "item1"
.. .. ..$ :List of 1
.. .. .. ..$ :List of 2
.. .. .. .. ..$ t: chr "Para"
.. .. .. .. ..$ c:List of 1
.. .. .. .. .. ..$ :List of 2
```

```
.. .. .. .. .. .. ..$ t: chr "Str"
.. .. .. .. .. .. ..$ c: chr "item2"
```

python 版的filter

python有一个包叫pandocfilters, 可以通过pip install pandocfilters来安装。这个包提供了4个函数，通常你只要写一个处理函数就可以了。

第一个filter，小写转大写

通过一个具体的实例来看，比如下面的这个filter的功能是把md文件中所有的字符串转为大写,代码链接等不会被转。

```
#!/usr/bin/env python

from pandocfilters import toJSONFilter, Str

def caps(key, value, format, meta):
  if key == 'Str':
    return Str(value.upper())

if __name__ == "__main__":
  toJSONFilter(caps)
```

可以看到，只有Str类型的数据被处理。对照filter输入部分看，就可以大概知道处理的结果。

第二个filter，英文翻译

另外如果要做复杂的filter，可以先看看pandocfilters包的源代码，它的源代码不多。但是不那么容易看明白。它的源码在这里 ， 主要函数是walk,它会被文档中的每一个节点调用。(实际上是只被调用一次，然后递归到树中的每个节点，如果你在上层节点调用的时候改掉这棵树，那么接下来的递归调用就使用的是新生成的树，而不是老的树了，下面对Para和Plain的处理就是这样，它在Para级别的树中修改了整个Para节点。)

现在想作一个filter，实现的功能是，将md中的英文翻译成中文，不影响md的格式，也不翻译代码、链接部分。代码的主要事情是把Para和Plain中的文本部分给组合起来，并且保留段落中的其他格式。

代码如下:

```
#!/usr/bin/env python
#coding=utf8

def transPara(key, value, format, meta):
      global paraString
      if key == 'Para'  or key == 'Plain':
          out= []
          curstr=""
          for sv in value:
              #print "processing...", sv
              if sv['t'] == "Str":
                  curstr = curstr + sv['c']
              elif sv['t'] == "Space" or sv['t'] == "SoftBreak" or sv['t'] == "LineBreak":
                  curstr = curstr + ' '
              else: #othe items , just keep them
                  if len(curstr) > 0 :
                      out.append(Str(transWrapper(curstr)))
                      curstr=""
```

```
                    out.append(sv)
                #print "out put:", out
            if len(curstr) > 0:
                out.append(Str(transWrapper(curstr)))
            #print 'final out for the para:' , out
            if key == 'Para' :
                return Para(out)
            elif key == 'Plain':
                return Plain(out)
        #assume simple strong ,Emph and header, no more sub structure in the strong/Emph/Header
        elif key == 'Strong':
            hstr = stringify(value)
            trStr=transWrapper(hstr)
            return Strong([Str(trStr)])
        elif key == 'Emph':
            hstr = stringify(value)
            trStr=transWrapper(hstr)
            return Emph([Str(trStr)])
        elif key == 'Header':
            hstr = stringify(value)
            trStr=transWrapper(hstr)
            return Header(value[0],value[1],[Str(trStr)])
        else:
            return None

if __name__ == "__main__":
  toJSONFilter(transPara)
```

其中transWrapper函数会实现英文到中文的翻译。要完全理解上面的代码需要明白pandocfilters.py中的walk函数。

我把这个代码放在github上去了，链接在这里.