bibmap 宏包和 bibmap 程序使用手册 [1]

胡振震[2]

2019-03-01

1 简介

1.1 bibmap 是什么

bibmap 是一个用于处理参考文献的 latex 宏包，包含一个 sty 文件，用于设置参考文献处理时的选项。一个 bibmap 程序，用于在后端处理参考文献数据。

bibmap 宏包加载了 natbib，chapterbib 等宏包，用于 latex 参考文献标注和文献表的生成，bibmap 宏包的工作原理有点类似 biblatex，但又是极度简化的，目的是直接利用现有的 latex 宏包 (比如 natbib，chapterbib)，避免像 biblatex 那样重写一整套的解决方案，整体来看 bibmap 宏包更像 gbt7714 宏包。

bibmap 后端程序类似 bibtex/biber 程序用于处理参考文献数据，其输出类似于 bibtex，输出的 bbl 文件包含 latex 直接能用的信息，而不像 biber 那样输出的是需要 biblatex 识别和处理的信息，但 bibmap 与 bibtex 的最大差别在于，bibmap 格式化文献表所用的样式文件是 python 数据和代码，因此是简单易懂的，目的是让用户可以更方便的设置参考文献格式，而不用去设计语法复杂的 bst 文件。

目前附带的 bibmap 程序是包括 python 源代码，可以直接运行 bibmap.PY 程序，在 windows 下可以利用打包成的 bibmap.exe 程序。

1.2 bibmap 的两大核心功能

1.2.1 参考文献生成

参考文献生成类似于传统基于 bibtex 的方法，基本用法一致。主要分如下几步：

1. 写 tex 文件源代码。

参考文献相关宏包加载仅使用 bibmap，无需再加载 natbib，也不能使用 biblatex(因为这是传统的方法，同时使用 biblatex 会产生冲突)。加载 bibmap 宏包时，可以为 bibmap 宏包设置一些选项 (见2.1节)。文档中正常使用 cite 等命令来引用文献，在需要生成文献表的地方加入命令`\bibliography{bibfile}`。

2. 编译文档源代码。

第一遍 xelatex 编译 tex 源代码比如：

```
xelatex test.tex
```

[1] Address: https://github.com/hushidong/biblatex-map

[2] Email: hzzmail@163.com

第二遍用 bibmap 程序运行，命令为:

```
python bibmap.py test
```

或

```
bibmap.exe test
```

第三和四遍使用 xelatex 编译 tex 源代码

```
xelatex test.tex
```

。

则能生成满足格式要求的参考文献表。

从实质上看，bibmap 宏包利用了 natbib 等宏包来形成合适的参考文献引用标注标签，利用 bibmap 程序生成格式化的 thebibliography 环境。可以说，bibmap 程序一定程度上是 bibtex 程序的替代，但它的参考文献格式设置要比 bst 文件简单得多，且由于使用现代 python 语言相比 bibtex 具有一些新的优势。而 bibmap 宏包则只是对 natbib 等宏包的集成和利用，通过一些方便的选项设置接口，简化一些格式选择。

1.2.2 bib 文件修改

bibmap 程序的另一大功能是对 bib 文件的数据进行修改。这可以与 tex 文档相关，也可以不相关。不相关时可以直接利用 bibmap 程序对指定 bib 文件做指定格式的数据处理，相关时则利用指定的格式对 bib 文件做类似 biblatex 和 biber 做的动态数据修改。

bibmap 程序对 bib 文件的数据修改，直接借鉴 biblatex 的设计，可以说是一套 python 的重新实现，使用逻辑也基本一致，唯一增加的功能是在域设置时可以设置 fieldfunction 选项，用于指定函数对域内容做处理。可以对 bib 文件的条目和域做非常细致的处理和修改。

2 参考文献生成的详细说明

参考文献生成的基本逻辑是这样的:

1) 首先，加载 bibmap 宏包的 tex 文件第一次编译后，bibmap 宏包将会在 aux 文件中写入一些设置信息，比如样式文件、bib 文件、文献引用等。

2) 接着，运行 bibmap 程序。

(a) 首先 bibmap 程序将读取 aux 文件，获取 bib 文件、样式文件、文献引用信息;

(b) 接着 bibmap 程序读取并解析 bib 文件，将有用信息放入 bibentries 字典中;

(c) 接着 bibmap 程序读取 bib 修改样式文件，根据样式文件中的选项对 bibentries 字典中各个条目做修改;

(d) 接着 bibmap 程序读取参考文献著录样式文件，根据设置的全局选项对引用

文献做排序处理，得到新的 bibentries 字典；

(e) 接着 bibmap 程序对排序后的 bibentries 字典中的各条参考文献 bibentry 进行格式化。格式化过程中根据选项信息，分条目类型处理。每个条目由各个输出项组成，每个输出项要处理相关的域格式，以及输出项前后的标点和字符串。整个条目的文本还需要处理替换字符串 (比如本地化字符串)、替换标点等，以形成最终输出的条目著录格式。所有的格式化后的条目组成一个字典便于输出；

(f) 最后 bibmap 程序首先输出新的 bib 文件，json 格式的文件，这是 bib 数据修改的结果。接着输出 bbl 文件，md 文件，html 文件，这是参考文献格式化后的结果。bbl 文件用于 latex 文档，md 用于文本复制，html 用于网页显示。

3) 接着再编译 tex 文件，这时 tex 编译器自动读取 bbl 文件用于生产参考文献表，此时的参考文献表的标签和正文引用的标签未完成。

4) 接着最后一遍编译 tex 文件，tex 编译器读取前一步编译写入到 aux 文件中的信息，来生成正文中文献引用的标注，以及文献表中的序号标签。

这四步编译逻辑与基于 bibtex 的方法是完全一致的，差别仅在于第二步的处理，利用 bibmap 程序代替了 bibtex 程序，以及内部的处理逻辑的差异。

2.1 bibmap 宏包及选项和命令

tex 文档使用 bibmap 宏包时,其加载方式与一般宏包一致,需要注意加载 bibmap 宏包后，无需再加载 natbib，也不能使用 biblatex。

例 1. bibmap 宏包加载

代码

```
1 \usepackage{bibmap}
```

bibmap 宏包可用的选项包括:

citestyle 用于指定文献引用的标注样式，主要包括 numeric，authoryear。所以的标注样式基于 natbib 宏包实现，设置 numeric 等价于为 natbib 设置 super,square,sort&compress 选项，设置 authoryear 等价于为 natbib 设置 authoryear 选项。

bibstyle 用于指定文献表的著录样式，主要包括 gb7714-2015,gb7714-2015ay，或者其它用户自定义的样式。设置 gb7714-2015 时调用宏包自带的 bibstylenumeric.py 样式文件生成文献表，设置 gb7714-2015ay 时调用宏包自带的 bibstyleauthoryear.py 样式文件生成文献表。当指定用户自定义的样式时则给出样式文件的文件名，比如 bibstyle=bibstyleexample，表示调用 bibstyleexample.py 样式文件来生成文献表。

mapstyle 用于指定参考文献数据修改样式，主要包括 default，或者其它用户自定义的样式。设置 default 时调用宏包自带的 bibmapdefault.py 对 bib 文件进行修改。当指定用户自定义的样式时则给出样式文件的文件名，比如 mapstyle=userdatamap，表示调用 userdatamap.py 样式文件来修改 bib 文件。

\citestyle `\citestyle{}`、`\setcitestyle{}`、`\bibpunct{}` 可以用来局部设置修改文献引用的标

注样式,比如在某一章中需要使用 numeric 样式,那么可以设置\citestyle{numeric}。这三个命令来源于 natbib。

\bmpbibstyle \bmpbibstyle可以用来局部设置文献的著录样式，比如在某一章中需要使用 authoryear 样式，那么可以设置\bmpbibstyle{bibstyleauthoryear.py}。或者其它用户自定义的样式，直接指定 py 文件即可。主要用于分章文献中。

\bmpmapstyle \bmpmapstyle可以用来局部设置 bib 文件修改的样式，比如在某一章中需要使用自定义的 userdefinedmap.py 样式, 那么可以设置\bmpbibstyle{userdefinedmap.py}。bibmap 程序处理时将局部调用 userdefinedmap.py 进行数据修改。主要用于分章文献中。

加载方式为:

例 2. bibmap 宏包加载

代码

```
\usepackage[citesytle=numeric,bibstyle=gb7714-2015]{bibmap}
```

2.2 参考文献生成示例

下面给出一个简单的示例:

例 3. 最小工作示例

代码

```
\documentclass{article}
    \usepackage{ctex}
    \usepackage{xcolor}
    \usepackage{toolbox}
    \usepackage{hyperref}
    \usepackage{lipsum}
    \usepackage[paperwidth=16cm,paperheight=10cm,top=10pt,bottom=10pt,left=1cm,right=1cm,
        showframe,showcrop]{geometry}
\usepackage[citestyle=numeric,bibstyle=gb7714-2015]{bibmap}
\usepackage{filecontents}
\begin{filecontents}{\jobname.bib}
@techreport{calkin,
  author = {Calkin, D and Ager, A and Thompson, M},
  title = {A Comparative Risk Assessment Framework for Wildland Fire
               Management: the 2010 Cohesive Strategy Science Report},
  number = {RMRS-GTR-262},
  year = {2011},
  pages = {8--9},
}

@incollection{buseck,
  author = {Buseck, P R and Nord, Jr, G L and Veblen, D R},
  title = {Subsolidus Phenomena in Pyroxenes},
  booktitle = {Pyroxense},
  address = {Washington, D.C.},
```

```
  publisher = {Mineralogical Society of America},
  year = {c1980},
  pages = {117--211},
}

@book{wfz,
  author = {王夫之},
  title = {宋论},
  edition = {刻本},
  address = {金陵},
  publisher = {湘乡曾国荃},
  year = {1865（清同治四年）},
}


@periodical{zgtsgxh,
  author = {中国图书馆学会},
  title = {图书馆学通讯},
  year = {1957/1990},
  number = {1-4},
  address = {北京},
  publisher = {北京图书馆},
}

@archive{bjsrmzfbgt,
  author = {北京市人民政府办公厅},
  title = {关于转发北京市企业投资项目核准暂行实施办法的通知:
                京政办发[2005]37号},
  year = {2005},
  date = {2005-07-12},
  urldate = {2011-07-12},
  url = {http://china.findlaw.cn/fagui/p_1/39934.html},
}
\end{filecontents}

   \begin{document}

   文献\cite{wfz,bjsrmzfbgt,zgtsgxh}\cite{calkin,buseck}

   \bibliography{\jobname}

   \end{document}
```

结果如图所示:

更多示例见 example 文件夹内:

egmwe.tex 展示了一个最小工作示例;

egcitations.tex 展示了标注的命令;

文献[1–3] [4,5]

文献[5]15 [4]25-30

参考文献

[1] 中国图书馆学会. 图书馆学通讯[J], 1957(1)-1990(4). 北京: 北京图书馆, 1957-1990.

[2] 王夫之. 宋论[M]. 刻本. 金陵: 湘乡曾国荃, 1865（清同治四年）.

[3] 北京市人民政府办公厅. 关于转发北京市企业投资项目核准暂行实施办法的通知: 京政办发 [2005]37 号[A/OL]. (2005-07-12)[2011-07-12]. http://china.findlaw.cn/fagui/p_1/39934.html.

[4] BUSECK P. R., NORD G. L., Jr, VEBLEN D. R. Subsolidus Phenomena in Pyroxenes[G]//Pyroxense. Washington, D.C.: Mineralogical Society of America, c1980: 117-211.

[5] CALKIN D, AGER A, THOMPSON M. A Comparative Risk Assessment Frame-

图 1. 最小工作示例结果

egmulticitesty.tex 展示了局部化的标注样式;

egchapterbib.tex 展示了分章文献的用法。

文献表条目的格式化操作，由指定的样式文件设置，设置的方法详见下一小节。

2.3 参考文献格式化设置说明

参考文献格式化工作主要由 bibmap 程序结合样式文件来实现。bibmap 程序主要实现的功能有:

1) 根据样式文件输出格式化后的 bbl 文件，便于 latex 文档直接使用。

2) 转存格式化的参考文献表文本为文本文件和网页文件，便于在其它文档中直接使用。

3) 除了 bibmap 程序内部处理逻辑外，样式文件可以全面控制参考文献的格式化。

4) bibmap 实现的内部处理: 根据选项进行排序，根据选项对姓名列表域、文本列表域、日期域、文本域、范围域格式化，根据选项对条目输出项进行组织和格式化。

有时间进一步介绍内部处理逻辑的实现。

2.3.1 样式文件的结构

样式文件主要由几个字典参数构成，分别是:

formatoptions 用于设置格式化的全局选项，所设选项必须是 bibmap 程序支持的选项，主要包括:

- ”style”: 写 bbl 信息的设置选项,authoryear,’numeric’

- ”nameformat”:’uppercase’,# 姓名处理选项:uppercase,lowercase,given-family,family-given,pinyin
- ”giveninits”:’space’,# 使用名的缩写，space 表示名见用空格分隔，dotspace 用点加空格，dot 用点，terse 无分隔，false 不使用缩写
- ”usesuffix”:True,# 使用后缀名
- ”maxbibnames”:3,#
- ”minbibnames”:3,#
- ”morenames”:True,#
- ”maxbibitems”:1,#
- ”minbibitems”:1,#
- ”moreitems”:False,#
- ”date”:’year’,#’ 日期处理选项’：year，iso，等
- ”urldate”:’iso’,#’ 日期处理选项’：year，iso，等
- ”origdate”:’year’,#’ 日期处理选项’：year，iso，等
- ”eventdate”:’year’,#’ 日期处理选项’：year，iso，等
- ’caseformat’:’none’,# 设计’none’,’sentencecase’,’titlecase’,’uppercase’,’lowercase’,’smallca
- ’numberformat’:’ordinal’,# 设计’ordinal’,’arabic’
- ”lanorder”:[’chinese’,’japanese’,’korean’,’english’,’french’,’russian’],# 文种排序，指定语言全面的顺序 [’chinese’,’japanese’,’korean’,’english’,’french’,’russian’],’none’
- ’sortascending’:True,# 排序使用升序还是降序，默认是升序，设置为 False 则为降序
- ”sorting”:[’author’,’year’,’title’],# 排序,或者指定一个域列表比如 [’key’,’author’,’year’,’tit ’none’

localstrings 用于设置本地化字符串，可以随意设置，只需符合规范。典型的设置方式为:

例 4. 简单的 localstrings 参数设置

代码

```
#本地化字符串
localstrings={
'andothers':{'english':'et al.','chinese':'等'},
'and':{'english':' and ','chinese':'和'},
'edition':{'english':' ed','chinese':'版'},#ed. 中的点不要，为方便标点处理
'in':{'english':'in: ','chinese':'见: '},
'nolocation':{'english':'[S.l.]','chinese':'[出版地不详]'},
'nopublisher':{'english':'[s.n.]','chinese':'[出版者不详]'},
'bytranslator':{'english':'trans by','chinese':'译'},
'volsn':{'english':'Vol.','chinese':'第'},
'volume':{'english':'','chinese':'卷'},
'numsn':{'english':'No.','chinese':'第'},
```

```
13    'number':{'english':'','chinese':'册'},
14    }
```

localpuncts 用于设置标点，可以随意设置，只需符合规范。典型的设置方式为:

例 5. 简单的 localpuncts 参数设置

代码

```
1   #标点
2   localpuncts={
3   'multinamedelim':', ',
4   'finalnamedelim':', ',
5   'andothorsdelim':', ',
6   'finalitemdelim':', ',
7   'multiitemdelim':', ',
8   'pagerangedelim':'-',
9   }
```

replacestrings 用于设置替换字符串，可以随意设置，只需符合规范。典型的设置方式为:

例 6. 简单的 replacestrings 参数设置

代码

```
1   #替换字符串
2   replacestrings={
3   '[出版地不详]: [出版者不详]':'[出版地不详 : 出版者不详]',
4   '[S.l.]: [s.n.]':'[S.l. : s.n.]',
5   '..':'.',
6   }
```

typestrings 用于设置类型和载体字符串，可以随意设置，只需符合规范。典型的设置方式为:

例 7. 简单的 typestrings 参数设置

代码

```
1   #类型和载体字符串
2   typestrings={
3   'book':'[M]',
4   'inbook':'[M]',
5   'standard':'[S]',
6   'periodical':'[J]',
7   'article':'[J]',
8   'newspaper':'[N]',
9   'patent':'[P]',
10  'online':'[EB]',
11  'www':'[EB]',
12  'electronic':'[EB]',
13  'proceedings':'[C]',
14  'inproceedings':'[C]',
15  'conference':'[C]',
16  'collection':'[G]',
17  'incollection':'[G]',
18  'thesis':'[D]',
19  'mastersthesis':'[D]',
20  'phdthesis':'[D]',
```

```
    'report':'[R]',
    'techreport':'[R]',
    'manual':'[A]',
    'archive':'[A]',
    'database':'[DB]',
    'dataset':'[DS]',
    'software':'[CP]',
    'map':'[CM]',
    'unpublished':'[Z]',
    'misc':'[Z]',
    }
```

datatypeinfo 用于将条目的域进行分类，6 个类型已经由 bibmap 程序设定，每一类域都有自身的特定处理逻辑。类型中包含的域可以由用户只有设定。典型的设置方式为:

例 8. 简单的 replacestrings 参数设置

代码

```
    #数据类型
    datatypeinfo={
    'namelist':['author','editor','translator','bookauthor'],
    'literallist':['location','address','publisher','institution','organization','school','
        language','keywords'],
    'literalfield':['title','journaltitle','journal','booktitle','subtitle','titleaddon','url
        ','doi','edition','version',
                'volume','number','endvolume','endnumber','type','note','labelnumber','
                    series'],
    'datefield':['date','enddate','year','endyear','urldate','origdate','eventdate','
        endurldate','endorigdate','endeventdate'],
    'rangefield':['pages'],
    'otherfield':['in','typeid','endpunct']#虚设的用于替换的域
    }
```

ibliographystyle 用于设定条目的著录格式，用户可以自由设定。一个条目类型作为一个字典项，项值内容则表示该条目的域的组织顺序及其内容格式，每个域的设置格式构成一个域格式字典，域格式字典中的项可以使用 bibmap 提供的项，以实现需要的格式，项值设置，可以使用前述的任何设置，包括全局选项同名的选项，用于局部化设置。

- ”fieldsource”, 用于设置当前输出项的域名
- ’options’, 用于设置当前输出项的格式选项
- ’prepunct’, 用于设置当前输出项前的标点
- ’prepunctifnolastfield’, 用于设置当前输出项前的标点，仅在前一项不存在时输出
- ”prestring”, 用于设置当前输出项前的字符串
- ’prestringifnumber’, 用于设置当前输出项前的字符串，仅当当前项为数字时输出

- 'replstring', 用于设置替换当前输出项前的字符串
- "posstring", 用于设置当前输出项后的字符串
- 'posstringifnumber', 用于设置当前输出项后的字符串，仅当当前项为数字时输出
- "omitifnofield", 用于忽略当前项，当该选项指定的域都不存在时，忽略当前项
- "omitiffield", 用于忽略当前项，当该选项指定的域只要一个存在，则忽略当前项
- 'pospunct', 用于设置当前输出项后的标点

2.3.2 样式文件的设置原理

样式设置的核心在 bibliographystyle 参数，它设置了条目的域的顺序、域本身的格式、域前后的标点和字符串。所以如 localstrings 等其它参数都是为 bibliographystyle 参数服务的，也包括全局的选项。全局选项与 localstrings 的差别在于，它还在 bibliographystyle 没有设置局部选项的情况下使用。选项的优先级是 bib 文件中条目本身给出的选项优先于 bibliographystyle 设置中给出的选项，进一步优先于全局的选项。总的来说，格式设置选项分三级，第一级是 bib 文件中条目给出的选项，第二级是条目类型格式设置时给出的选项，第三级才是全局选项。

一个简单 bibliographystyle 参数设置如例9所示，其中设置了条目类型"misc" 的著录格式，"book" 等其他所有类型复用 misc 类型的格式，复用的方式很简单，就是把 misc 作为其它类型的键值，比如:`"book":"misc"`。

misc 类型的格式中，第一项，输出的备选域包括作者，编者，译者，方式为: `"fieldsource":['author','editor','translator']`, 因为这是姓名列表类型的域，给它指定了条目类型一级的选项: `'options':{'nameformat':'uppercase'}`，意为姓名采用全大写字母的格式。

第二项，输出的备选域为标题，方式为: `"fieldsource":['title']`; 因为这是文本类型的域，给它指定了给它指定了条目类型一级的选项:

`'options':{'caseformat':'sentencecase'}`, 意为标题采用句子模式的大小写格式; 另外设置了前置标点，前一项不存在时输出的标点，后置的字符串，方式为:

`'prepunct':". ",'prepunctifnolastfield':'','posstring':r"\allowbreak\typestri`

其中默认前置标点为一个点加空格，当前一项不存在时则标点变为空，后置一个字符串中加入了 `\allowbreak`命令，以及一个`\typestring`，`\typestring`最后会根据 typestrings 参数的设置替换为对应的字符串，若 typestrings 设置如例7所示，那么`\typestring`会替换为 [z]。

第三项,输出的备选域为'howpublished',方式为: `"fieldsource":['howpublished']`; 仅设置了一个前置标点为点加空格: `'prepunct':". "`。

第四项，输出的备选域为地址，location 和 address 都表示地址，有时会混用，所

以两者都给出，方式为: `"fieldsource":['location','address']`; 仅设置了一个前置标点为点加空格: `'prepunct':". "`。

第五项,输出的备选域为出版者,institution 和 publisher 都可以表示出版者,有时会混用，所以两者都给出，方式为: `"fieldsource":['institution','publisher']`; 设置了一个前置标点为冒号加一个空格，当前一项即地址项不存在时则输出标点为点加空格: `'prepunct':": ",'prepunctifnolastfield':'. '`。

第六项，输出的备选域为日期，'date' 和'year' 都可以表示日期，有时会混用，所以两者都给出，方式为: `"fieldsource":['date','year']`; 设置了一个前置标点为逗号加一个空格: `'prepunct':", "`。

第七项，输出的备选域为页码，方式为: `"fieldsource":['pages']`; 设置了一个前置标点为冒号加一个空格: `'prepunct':": "`。

第八项，输出的备选域为网址访问日期，方式为: `"fieldsource":['urldate']`; 设置了一个前、后字符串为方括号，目的是把访问日期用方括号包围起来:

`'prestring':"[","posstring":"]"`。

第九项，输出的备选域为网址，方式为: `"fieldsource":['url']`; 设置了一个前置标点为点加空格，前置字符串为一个 newblock 命令和 url 命令和花括号，后置字符串为花括号。目的是把网址包围在 url 命令内:

`'prepunct':r". ",'prestring':r'\newblock\url{','posstring':'}'`。

第十项，输出的备选域为 doi，方式为: `"fieldsource":['doi']`; 设置了一个前置标点为点，前置字符串为一个 newblock 命令加 DOI: 和 url 命令和花括号，后置字符串为花括号。目的是把网址包围在 url 命令内:

`'prepunct':".",'prestring':r'\newblock DOI:\doi{','posstring':'}'`。

最后一项是，虚设的域 endpunct，目的就是用来做替换。这里将其替换为一个点。方式为: `{"fieldsource":['endpunct'],'replstring':"."}`。

例 9. 简单的 bibliographystyle 参数设置

代码

```
#条目的著录格式
bibliographystyle={
"misc":[
{"fieldsource":['author','editor','translator'],'options':{'nameformat':'uppercase'}},
{"fieldsource":['title'],'options':{'caseformat':'sentencecase'},'prepunct':". ",'
    prepunctifnolastfield':'','posstring':r"\allowbreak\typestring"},
{"fieldsource":['howpublished'],'prepunct':". "},
{"fieldsource":['location','address'],'prepunct':". "},
{"fieldsource":['institution','publisher'],'prepunct':": ",'prepunctifnolastfield':'. '},
{"fieldsource":['date','year'],'prepunct':", "},
{"fieldsource":['pages'],'prepunct':": "},
{"fieldsource":['urldate'],'prestring':"[","posstring":"]"},
{"fieldsource":['url'],'prepunct':r". ",'prestring':r'\newblock\url{','posstring':'}'},
{"fieldsource":['doi'],'prepunct':".",'prestring':r'\newblock DOI:\doi{','posstring':'}'},
```

```
{"fieldsource":['endpunct'],'replstring':"."}
],
"book":"misc",
"article":"misc",
"newspaper":"misc",
"inbook":"misc",
"inproceedings":"misc",
"incollection":"misc",
"proceedings":"misc",
"collection":"misc",
"standard":"misc",
"patent":"misc",
"online":"misc",
"www":"misc",
"electronic":"misc",
"report":"misc",
"techreport":"misc",
"periodical":"misc",
"thesis":"misc",
"manual":"misc",
"unpublished":"misc",
"database":"misc",
"dataset":"misc",
"software":"misc",
"map":"misc",
"archive":"misc",
"phdthesis":"misc",
"mastersthesis":"misc",
}
```

例9给出的参考文献著录格式，仅对 misc 类型做了设置，其它类型全部复用该类型的格式。如果要对其它类型做更专门的设置，可以采用类似 misc 的方式进行设置。宏包提供的 bibstylenumeric.py 样式文件就是精确实现国标要求的设置。

需要注意的是，bibliographystyle 参数设置格式，主要设置各输出项所用的域，以及标签，前后添加的字符串等。而域内容本身的格式，则由 bib 文件中条目本身提供的选项、各输出项设置时给出的条目类型一级的选项，以及 formatoptions 参数给出的全局选项控制。格式在 bibmap 程序中实现，不同类型的域实现逻辑不同。主要分为：'namelist'(姓名列表域，用于处理姓名列表，姓名成分解析和姓名格式化)、'literallist'(文本列表域，用于处理文本列表，解析和格式化)、'literalfield'(文本列表域，包含字符串、数字域等，有数字序号、字母大小写等处理)、'datefield'(日期域，用于处理日期和日期范围，日期成分解析和日期格式化)、'rangefield'(范围域，主要是页码域，处理页码间隔符)、'otherfield'(虚设域用于替换)。各个域的处理逻辑在 bibmap 程序中实现，用户可以控制部分，都可以通过三个层级的选项来进行控制。

3 bib 文件修改的详细说明

3.1 bib 文件修改示例

bib 文件修改功能主要有如下几个:

1) bib 文件的读取和解析

2) bib 文件的转存，包括从大的 bib 文件抽取引用的文献保存为一个小的 bib 文件，将 bib 文件的内容存储为 json 格式的问题。

3) 参考文献条目的修改，包括条目类型的修改，条目内部的域的修改等，包括删除、变化、转换等等。

修改的逻辑是 bibmap 程序读取数据修改样式文件内的 sourcemaps 参数，这个 sourcemaps 参数与 biblatex 中 sourcemap 非常像，基本的逻辑也是一样的，只是把 tex 的写法转换成了 python 方便表示的写法，就是把 biblatex 用 tex 命令表示的`\maps`、`\map`和`\step`转换为用 json 格式表示。sourcemaps 参数的设置方法见下一节。

3.2 bib 文件修改设置说明

3.2.1 数据修改样式文件的结构

数据修改样式文件是一个 py 文件，内部但要保存一个或多个 sourcemaps 参数，但要注意虽然可以存在多个 sourcemaps，但后面的 sourcemaps 会覆盖前面的 sourcemaps，也就是只有最后一个 sourcemaps 会起作用。

典型设置如例11所示，其中 sourcemaps 是一个列表参数，包含了所有 map 步骤，每个 map 也是一个列表，一个 map 就是对所有条目进行一次遍历修改。一个 map 列表中包含有任意数量的 step(步骤)，一个 step 是由一个 key-val 参数构成字典数据结构。数据的修改是由 step 步的 key-val 参数字典决定的。

简单来说，数据修改逻辑是这样的，bibmap 程序读取 sourcemaps 参数后，开始遍历所有的 map，这一个 map 处理需要遍历所有的 bib 条目，对于每个条目，遍历当前 map 内的所有 step 步，step 步的执行规则由 step 步内的参数以及处理结果确定。

例 10. bib 数据修改的处理逻辑

代码

```
for map in maps #遍历maps中的所有map
      for entry in entries #对所有的条目均执行该map
            for step in map #遍历map中的所有step
                  code for the step to modify the bib entry
```

例11给出了 10 次 map 的数据修改。

第一次修改, 内部只有一个 step 步, 目的是将 ELECTRONIC 类型转换为 online

类型:

```
{"typesource":"ELECTRONIC","typetarget":"online"}。
```

其中使用了键 typesource 和 typetarget，意为将类型为 typesource 值的条目，转换为 typetarget 值的类型。比如这里将 ELECTRONIC 类型条目转换为 online 条目。

第二次修改，内部只有一个 step 步，目的是将 source 域转换为 url 域:

```
{"fieldsource":"source","fieldtarget":"url"}
```

其中使用了键 fieldsource 和 fieldtarget，意为将条目中域名为 fieldsource 值的域，转换为名为 fieldtarget 值的域。比如这里将 source 域转换为 url 域。

第三次修改，内部只有一个 step 步，目的是将 urldate 域的信息“yyyy-m-d”转换为“yyyy-mm-dd”便于日期能够正常解析。

```
{"fieldsource":"urldate","match":r'(\d\d\d\d)\-(\d)\-(\d)',"replace":r'\1-0\2
```

其中使用了键 fieldsource、match、replace，意为对条目中域名为 fieldsource 值的域做匹配，当匹配到 match 键的值时，将匹配内容体会为 replace 键的值。这里的匹配和替换可以使用普通的字符串，也可以使用正则表达式，这里就是使用的正则表达式。但是要注意一旦涉及到已经存在的域内容的修改，都需要选项 overwirte 的支持，也就是要加上 overwrite 键。就是说，这次修改实际不会执行，而下一次的修改，由于 overwrite 存在才会执行。

第四次修改，基本同第三次修改，由于 overwrite 键设置为 true，则修改会实际执行:

```
{"fieldsource":"date","match":r'(\d\d\d\d)\-(\d)\-(\d)',"replace":r'\1-0\2-0\
```

第五次修改，类似于第二次修改，将 refdate 域转换为 urldate 域:

```
{"fieldsource":"refdate","fieldtarget":"urldate"}
```

第六次修改，内部有 2 个 step 步，目的是为 newspaper 类型的条目，设置 note 域为 news。

第一步为: `{"pertype":"newspaper"}`，是做了条目类型约束，pertype 键表示只有其键值给出的条目类型才做修改。

第二步为:

```
{"fieldset":"note","fieldvalue":"news","overwrite":True}
```

使用了键包括 fieldset、fieldvalue、overwrite。fieldset 表示对其值表示的域做修改，fieldvalue 就是修改后的值，overwrite 表示域有原内容的情况下做覆盖修改。若不该处 overwrite，那么当 note 域原来就存在情况下，当前步不实际执行。

第七次修改，内部有 2 个 step 步，目的是设置 edition 域等于 version。

第一步为: `{"fieldsource":"version","final":True}`

是做域查找，当条目中 version 存在时，并记录 version 的域值。final 键的目的是，当条目中不存在 version 域时，本次 map 直接终止，即后面的 step 步不再执行。

第二步为:

`{"fieldset":"edition","origfieldval":True}`

使用了键包括 fieldset、origfieldval。fieldset 去设置 edition 域，域的值为 origfieldval 表示的内容，即前面一步 fieldsource 查找并记录下的 version 域的值。origfieldval 表示前面的最近一步 fieldsource 记录下的域值信息。

第八次修改，与第七次类似，目的是将 entrykey 域的内容设置给 keywords 域。

第九次修改，与第七次类型，目的是对于存在 note 域的情况，将其值添加到 keywords。需要注意的是第二步中，使用了 append 键，目的是修改过程中，不是覆盖修改，而是做添加修改。overwrite 键也需要给出，否则无法对已经存在的域做修改。

第二步设置为: `{"fieldset":"keywords","origfieldval":True,"overwrite":True,"app`

第十次修改，内部由两个 step 步，目的是根据标题的字符编码范围确定标题的语言类型。

第一步: `{"fieldsource":"title","match":r'[\u2FF0-\u9FA5]',"final":True}`

目的是搜索 fieldsource 指出的 title 域，是否匹配 match 指出的正则表达式，当不能匹配时，根据 final 键，直接终止当前 map，即第二步 step 不再处理。

第二步: `{"fieldset":"userd","fieldvalue":"chinese"}`

目的是，当前一步匹配成功，那么将当前条目的 userd 域设置为 fieldvalue 指出的值即 chinese。

例 11. 典型的 bib 数据修改样式参数设置

代码

```
#
#数据修改的设置
#
#
# 为数据map设置参数
# 选项的逻辑与biblatex基本一致，
# 差别包括:overwrite选项可以放到step步中表示
#[]=maps
#[[],[]]=map in maps
#[[{optionkey:optionval},{optionkey:optionval}],[]]=step in map in mpas
#注意python正则表达方式与perl的略有不同，比如unicode表示
#python用\xHH,\uHHHH,\UHHHHHHHH表示，而perl直接用\x{HHHH}表示。
sourcemaps=[#maps
       [#map1:将ELECTRONIC类型转换为online类型
              {"typesource":"ELECTRONIC","typetarget":"online"}#step1
       ],
       [#map2:将source域转换为url域
              {"fieldsource":"source","fieldtarget":"url"}#step1
       ],
       [#map3:将urldate域的信息“yyyy-m-d”转换为“yyyy-mm-dd”,注意正则表达式直接写不用在外面
           套""
              {"fieldsource":"urldate","match":r'(\d\d\d\d)\-(\d)\-(\d)','replace":r'
```

```
                \1-0\2-0\3'}#step1
    ],
    [#map4:将urldate域的信息“yyyy-m-d”转换为“yyyy-mm-dd”,注意正则表达式直接写不用在外面套""
        {"fieldsource":"date","match":r'(\d\d\d\d)\-(\d)\-(\d)',"replace":r'\1-0\2-0\3',"overwrite":True}#step1
    ],
    [#map5:将refdate域转换为urldate域
        {"fieldsource":"refdate","fieldtarget":"urldate"}#step1
    ],
    [#map6:对于newspaper类型，设置note为news
        {"pertype":"newspaper"},#step1
        {"fieldset":"note","fieldvalue":"news","overwrite":True}#step2
    ],
    [#map7:设置edition域等于version
        {"fieldsource":"version","final":True},#step1
        {"fieldset":"edition","origfieldval":True}#step2
    ],
    [#map8:设置entrykey域设置给keywords
        {"fieldsource":"entrykey"},#step1
        {"fieldset":"keywords","origfieldval":True}#step2
    ],
    [#map9:对于存在note域的情况，将其值添加到keywords
        {"fieldsource":"note","final":True},#step1
{"fieldset":"keywords","origfieldval":True,"overwrite":True,"append":True}#step2
    ],
     [#map10:根据标题的字符编码范围确定标题的语言类型
        {"fieldsource":"title","match":r'[\u2FF0-\u9FA5]',"final":True},#step1
        {"fieldset":"userd","fieldvalue":"chinese"}#step2
    ],
]
```

例11大体上给出了数据修改参数设置的方法，下一节详细介绍一下 bibmap 支持的键。

3.2.2 数据修改支持的选项

bibmap 数据修改所用的选项，即各 step 步中的 key 值，完全借鉴 biblatex 中的设计，其意义也是一致的，根据常用程度做了取舍，并增加了部分选项如 fieldfunction，实现的选项包括：

- typesource 对应的值为 entrytype 名
- typetarget 对应的值为 entrytype 名
- fieldsource 对应的值为 entryfield 名
- fieldtarget 对应的值为 entryfield 名
- match 对应的值为 regexp(正则表达式)

- notmatch 对应的值为 regexp(正则表达式)
- replace 对应的值为 regexp(正则表达式)
- notfield 对应的值为 entryfield 名
- final 对应的值为 true 和 false(bool 值)
- origfieldval 对应的值为 true, false(bool 值)
- append 对应的值为 true, false(bool 值)
- pertype 对应的值为 entrytype 名即条目类型
- pernottype 对应的值为 entrytype 名即条目类型
- fieldset 对应的值为 entryfield 名
- fieldvalue 对应的值为 string(字符串)
- null 对应的值为 true, false(bool 值)
- origfield 对应的值为 true, false(bool 值)
- origentrytype 对应的值为 true, false(bool 值)
- origfieldval 对应的值为 true, false(bool 值)
- fieldfunction 对应的值为用户指定的函数名,目前提供的函数包括: 'sethzpinyin','sethzstro 'setsmallcaps','setalltitlecase'。在域内容处理时,当给出'fieldfunction':'sethzpinyin' 选项时，程序会调用 sethzpinyin 函数以域内容为参数，输出其对应的拼音。 'sethzstroke' 设置用于排序的笔画顺序字符串。其它的函数用于设置字符串字母的大小写格式。

其中绝大部分在前一节以及讲过，未介绍的包括 notmatch 与 match 类型，只是搜索的是不匹配的情况。notfield 是做约束，当 notfield 指出的域不存在时，才执行操作，通常和 final 联用。pernottype 类似于 pertype，也是做约束，当不是 pernottype 指定的条目类型时，才执行操作。null 表示域值不存在，即用于删除当前域。origfield 用于标签前面 step 步中最后一次 fieldsource 搜索并记录下的域的域名，而 origfieldval 则是域值。origentrytype 则是前面最近一次 typesource 搜索成功情况下记录下来的条目类型。

未实现的选项包括:

- entryclone=?clonekey?
- entrynew=?entrynewkey?
- entrynewtype=?string?
- entrytarget=?string?
- entrynocite=true, false default: false

- entrynull=true, false default: false
- matchi=?regexp?
- notmatchi=?regexp?

未实现的主要是条目整体复制和处理相关的内容，以及区分字符大小写的正则匹配。

4 bibmap 程序命令行用法

4.1 bibmap 程序输入参数

bibmap.py 或 bibmap.exe

filename 单个输入文件的文件名，可带后缀名如 bib 或 aux，无后缀名时默认为辅助文件.aux

[-h] 输出帮助

[-a AUXFILE] 辅助文件的文件名, 可带后缀名.aux, 如果 filename 已经设置 aux 文件则无效

[-b BIBFILE] 文献数据库文件名，可带后缀名.bib，如果 filename 已经设置 bib 文件则无效

[-s STYFILE] 设置文献样式文件的文件名，可带后缀名.py，不给出则使用默认样式文件

[-m MAPFILE] 数据库修改设置文件文件名，可带后缀名.py，不给出则使用默认设置文件

`[--addpinyin]` 给出该选项则将为每个文献条目增加带有拼音的 key 域。

`[--nofmt]` 给出该选项则不做格式化输出

`[--nobdm]` 给出该选项则不做 bib 数据修改

其中涉及到三种文件:

一是 aux 文件，如果是要得到格式化的文献表，那么这是最重要的文件，由 tex 编译生成，当使用 bibmap 宏包时，可以通过宏包选项设置样式文件，而 bib 文件通过 bibliography 命令也会在该文件中指出。

二是 bib 文件，这是参考文献数据源文件，可以由通过 bibliography 命令在 aux 文件内给出，也可以直接利用选项给出。

三是 py 文件，这是用于设置数据修改和文献格式化的文件，是 python 代码。宏包自带的样式，通常 bibmap*.py 是用于 bib 文件数据修改的，而 bibstyle*.py 是用于格式化文献表的。

4.2 bib 文件数据修改

直接在命令行输入脚本及其参数：

例 12. bib 文件数据修改命令-默认情况

代码

```
python bibmap.py biblatex-map-test.bib
或
bibmap.exe biblatex-map-test.bib
```

此时，bibmap 读取 biblatex-map-test.bib 文件，并根据默认的数据修改设置 bibmapdefault.py 做修改，此时还会自动的做格式化后的文献表输出。

例 13. bib 文件数据修改命令-不输出格式化文献表

代码

```
python bibmap.py biblatex-map-test.bib --nofmt
或
bibmap.exe biblatex-map-test.bib --nofmt
```

此时不再输出格式化后的文献表。

例 14. bib 文件数据修改命令-指定数据修改设置

代码

```
python bibmap.py biblatex-map-test.bib --nofmt -m bibmapaddkw.py
或
bibmap.exe biblatex-map-test.bib --nofmt -m bibmapaddkw.py
```

此时使用指定的数据修改设置 bibmapaddkw.py 代替默认的 bibmapdefault.py 对数据库 bib 文件做修改。

又比如给中文的参考文献条目增加带有拼音的 key 域，可以采用如下方式：

例 15. bib 文件数据修改命令-增加拼音信息的 key 域

代码

```
bibmap.exe biblatex-map-test.bib --nofmt --addpinyin
或
bibmap.exe biblatex-map-test.bib --nofmt -m bibmapaddpinyinkey.py
或
python bibmap.py biblatex-map-test.bib --nofmt --addpinyin
或
python bibmap.py biblatex-map-test.bib --nofmt -m bibmapaddpinyinkey.py
```

4.3 参考文献格式化输出

直接在命令行输入脚本及其参数：

例 16. 参考文献格式化命令-默认情况

代码

```
python bibmap.py egtest
或
bibmap.exe egtest
```

此时输入一个辅助文件 egtest.aux，其它所有的参数根据对 egtest.aux 的解析来获取，如果没有解析到，若存在默认的设置，则使用默认的设置文件。若没有默认设

置，则可以通过可选参数来指定：

例 17. 参考文献格式化命令-指定 bib 文件

代码

```
python bibmap.py egtest -b biblatex-map-test.bib
或
bibmap.exe egtest -b biblatex-map-test.bib
```

当 aux 文件未给出格式化设置文件时，也可以用-s 选项给出，格式化设置文件(即文献样式文件)，比如

例 18. 参考文献格式化命令-指定样式文件

代码

```
python bibmap.py egtest -s bibstyleauthoryear.py
或
bibmap.exe egtest -s bibstyleauthoryear.py
```

5 结论

本宏包最初的工作，只是想设计一个 python 脚本，方便对 bib 文件进行修改，而不去使用 bib 管理软件或者使用 biber 程序 (biber 做 bib 修改需要 tex 文件支持，因而为修改 bib 必须写一个 tex 文件)。做到一定程度后，本想为 biblatex-check 脚本做个 pr 了事，但没想到又接着写下去了，慢慢变成了两个主要功能，一是参考文献 bib 文件数据修改，二是参考文献格式化，既便于 bib 文件的修改、转换和保存，也便于格式化后直接给 latex 应用或者文本复制和网页显示。第二个功能整体上已经类似于 bibtex，所以也就继续完善下来，最终形成了目前的 bibmap 宏包。基于传统的参考文献生成方法，利用 natbib 等现成宏包，替代 bibtex 程序，形成一个便于参考文献生成的程序和宏包。

目前功能已经大体实现，欢迎大家使用，并提出各方面的意见建议!

值得说明的是，sty 文件中标注命令相关内容，详细参考了 gbt7714 宏包和 ucas-thesis 模板，部分代码直接借用，对其作者 Lee Zeping 和 Mo Huangrui 表示由衷的感谢!

因为 bibmap 宏包的标注样式基于 natbib 宏包实现，而要使用分章的参考文献，那么需要使用 chapterbib 宏包，因此对这两个宏包做一些介绍方便使用。

A natbib 宏包用法简介

A.1 基本概念

natbib 宏包主要用来生成标注标签，在不同的章节可以使用不同的标注样式。

A.2 用法和注意事项

1. 当与 chapterbib 联用时，sectionbib 选项需要在 natbib 宏包加载时给出。给 chapterbib 设置无效。

2. 文献表的样式仍然设置用`\bibliographystyle{}`命令。

3. natbib 提供了常用的 citet 和 citep 命令，通常 citet 会提供作者信息，citep 会加上包围的符号。当然在 natbib 加载不同的选项时，会有不同的表现。

4. natbib 提供了三个 plainnat.bst abbrvnat.bst unsrtnat.bst 样式文件。

5. thebibliography 环境的语法:

```
\bibitem[Jones et al.(1990)]{jon90}...
```

或

```
\bibitem[Jones et al.(1990)Jones, Baker,and Williams]{jon90}...
```

其中 (年份) 前的是缩略的姓名列表。后面的则是详细的列表。该列表在首次引用需要详细姓名的样式中会用到。

6. 提供了大量的 cite 类命令，可以输出作者，年份，数字等信息。可以大小写变化。这些按需使用，最常用的还是前面提过的 citet 和 citep。

7. 还可以利用`\defcitealias`命令定义别名标签, 某些情况下要生成特定的标签会非常有用。

8. 可以利用`\setcitestyle`命令来局部设置标注的样式,该命令的参数是 key=val 或 key 结构。也可以利用`\bibpunct`命令来设置标注的样式，主要是标点。也可以利用`\bibstyle@xxx`命令来预设置一种标注样式，然后用`\citestyle`命令来使用。

比如: `\newcommand{\bibstyle@agu}{\bibpunct{[}{]}{;}{a}{,}{,~}}`

然后使用: `\citestyle{agu}`

natbib 已经预定义的样式包括: plain、plainnat、agu、egu、agms、cospar、nature。

setcitestyle, bibpunct, citestyle 的设置的样式会覆盖宏包加载时的样式。

9. 可以使用如下命令进一步格式化:

`\bibsection` `\bibpreamble` `\bibfont` `\citenumfont` `\bibnumfmt` `\bibhang` `\bibsep`

10. 标注的排序和压缩，使用 sort&compress 选项。

11. 顺序编码样式中的，分组引用，可以使用 merge 选项，然后使用 * 来构成分组。比如: `\citep{feynmann,*salam,*epr}`会形成一个数字。

12. 作者年制中的长标注，比如第一次引用时，使用长标注。可以使用`\citet*`等命令。

13. 常用选项包括:

round (default) for round parentheses;

square for square brackets;

curly for curly braces;

angle for angle brackets;

semicolon (default) to separate multiple citations with semi-colons;

colon the same as semicolon, an earlier mistake in terminology;

comma to use commas as separators;

authoryear (default) for authoryear citations;

numbers for numerical citations;

super for superscripted numerical citations, as in Nature;

sort orders multiple citations into the sequence in which they appear in the list of references;

sort&compress as sort but in addition multiple numerical citations are compressed if possible (as 36, 15);

compress to compress without sorting, so compression only occurs when the given citations would produce an ascending sequence of numbers;

longnamesfirst makes the first citation of any reference the equivalent of the starred variant (full author list) and subsequent citations normal (abbreviated list);

sectionbib redefines thebibliography to issue section* instead of chapter*; valid only for classes with a chapter command; to be used with the chapterbib package;

nonamebreak keeps all the authors' names in a citation on one line; causes overfull hboxes but helps with some hyperref problems;

merge to allow the * prefix to the citation key, and to merge such a citation's reference with that of the previous citation;

elide to elide common elements of merged references, like the authors or year;
mcite to recognize (and ignore) the merging syntax

14. 注意: 当出现 unskip 命令不能出现在竖直环境中的错误时，说明引用命令没有进入水平模式，只要在引用命令前加上文本即可。

B chapterbib 宏包用法简介

B.1 基本概念

分章文献的基本原理是利用 include，来对不同的章生成不同的 aux 文件，然后对各个 aux 文件信息仅需解析，然后处理 bib 文件，进而生成多个 bbl 文件即多个 thebibliography 环境，进而可以生成多个文献表。

因此对于利用 bibtex 处理 bib 文件的方法,每个文件中必须要`\bibliographystyle`和 `\bibliography`命令指出相应的 bst 样式和 bib 文件。才能让 bibtex 读取每个 aux 文件时获取足够的信息。

chapterbib 也提供了 cbunit 和`\cbinput`命令，可以不基于 include 方法来生成多个文献表。但是这种方法需要比较麻烦的编译过程，所以不推荐使用。

B.2 用法和注意事项

1. 在 book 和 report 类中, 参考文献的标题会被当做是一个 chapter* 的标题, 但若要使得参考文献的标题降一个层级，比如降到 section*。那么可以使用 sectionbib 选项，但与 natbib 连用时，该选项会失效，替代的方法是给 natbib 设置这个选项。

2. 要在主文档中使用一个完全不同的参考文献,那么不能再使用`\bibliography`命令，因为主文档 bbl 会包含值文档的。所以只能使用手写的 thebibliography 环境。

3. 要在主文档生成一个包含所有子文档文献的文献表,那么可以使用`\bibliography`命令。而要避免报错，`\bibliographystyle` 必须要放在所有的 include 之前。或者也可以在不同的编译步使用或不使用 [rootbib] 选项来实现。

4. 要把所有各章的文献表收集起来放到主文档中，那么可以使用 [gather] 选项，然后在主文档中使用 `bibliography`命令。也可以通过定义`\StartFinalBibs`来重定义文献表的标题。

5. 如果文献表不仅要收集起来，也需要在各章显示，那么可以使用 [duplicate] 选项。