THE SCOOP TEMPLATE ENGINE

ROLAND HERZOG

ABSTRACT. This document describes version 0.0.19 of the SCOOP TEMPLATE ENGINE.

CONTENTS

The scoop template engine is meant to facilitate the preparation of scientific articles written in LaTeX, abiding to the formatting standards of various scientific journals. It currently supports 106 different journals across several disciplines, listed in Section 11. Are you tired of looking up how a journal wants you to use its `\author`, `\address` and `\affil` commands? Would you rather prefer to format your manuscripts as easily as this?

```
ste prepare --template amspreprint
ste prepare --template sinum
ste prepare --template crelle
```

Then the Scoop Template Engine is for you.

Important Note

The Scoop Template Engine is intended to be used on Linux, macOS or Microsoft Windows operating systems. It is written in Python has been tested with Python 3.10 as well as TeX Live 2022. It is based on the assumption that you will be processing your `.tex` documents using `pdflatex`.

1. Installation

1.1. Installation Using `pip3`.

Launch a shell on your operating system and execute the following commands to install and initialize the Scoop Template Engine.

```
pip3 install scoop-template-engine
ste init
```

The initialization step will download and setup the journals' resources (such as `.cls`, `.sty` and `.bst` files) from the respective publisher's site.[1] It is safe to re-run `ste init` at any time.

1.2. Installation Using `pip3` into a Virtual Environment.

If you prefer to install the Scoop Template Engine into a virtual environment (say, called `ste`), you would first create this virtual environment in a convenient directory, and activate it:

```
python3 -m venv ste
source ste/bin/activate    # on Linux and macOS
ste\Scripts\activate.bat   # on Windows
```

[1]The reason why you have to download these resources separately, rather than obtain them bundled with the Scoop Template Engine, is that many publishers do not furnish a redistribution friendly license for their LaTeX resources.

Subsequently you install the Scoop Template Engine as above:

```
pip3 install scoop-template-engine
ste init
```

1.3. **Upgrading from an Earlier Version.**

To upgrade an existing installation using `pip3`, issue

```
pip3 install --upgrade scoop-template-engine
ste init
```

2. Quick Start

The primary purpose of the Scoop Template Engine is to allow the separation of layout (which depends on the journal and will be automatically generated) from content. To see the Scoop Template Engine in action, you can run

```
ste start
```

This will create the files for a sample document in the current folder. You can then prepare the document for three different journals as follows:

```
ste prepare --template amspreprint
ste prepare --template sinum
ste prepare --template crelle
```

Here `amspreprint` represents a preprint based on `amsart.cls`, `sinum` refers to the *SIAM Journal on Numerical Analysis*, and `crelle` stands for the *Journal für die reine und angewandte Mathematik*. These merely serve as examples from the selection of 106 journals currently supported.

The generated files `manuscript-amspreprint.tex`, `manuscript-sinum.tex` and `manuscript-crelle.tex` are now ready to be compiled using `pdflatex`, e. g., using your favorite LaTeX editing system.

How does this work? Figure 1 illustrates a typical workflow. The individual steps are as follows.[2]

(1) Create a data file stating the manuscript's title, the authors, their affiliations, the LaTeX preamble describing, e. g., the packages you would like to use, as well as the name(s) of the file(s) holding the actual content of your manuscript (`content.tex` in our example). The data file is in YAML syntax.

[2]All file names are merely examples and can be customized.

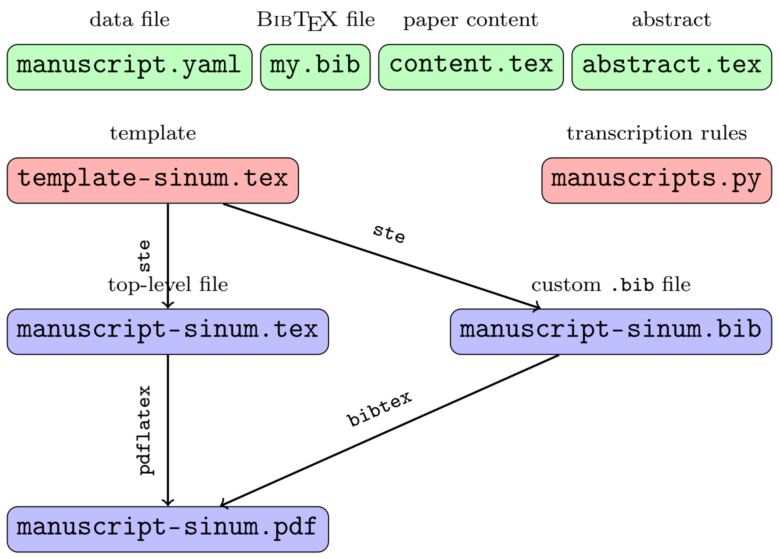

Figure 1. Illustration of a typical workflow to generate documents using the Scoop Template Engine. User defined input is shown in green. Files provided along with the Scoop Template Engine are shown in red. The generated output files are shown in blue. File names are examples only. `sinum` is the identifier for the *SIAM Journal on Numerical Analysis*.

Figure 2 shows a dummy data file named `manuscript.yaml`, matching the one created via the convenience command `ste start`. The full set of features of the data file will be explained in Section 4.

(2) Create the files holding the actual content of your manuscript. In our example, this is `content.tex`, which typically starts with something like `\section{Introduction}`. Create a separate file holding the abstract. In our example, this is `abstract.tex`. For your references, you may want to use a BibTeX/BibLaTeX file, such as `my.bib`. More on bibliography handling can be found in Section 5.

(3) Running, as in our example,

```
ste prepare --template sinum
```

will create the top-level file `manuscript-sinum.tex` (the style compliant with the *SIAM Journal on Numerical Analysis*, as an example). It will also create what we call a custom BibTeX file `manuscript-sinum.bib` from those references of `my.bib` which actually get cited inside the document. If necessary, entries in `my.bib` will be converted during the creation of `manuscript-sinum.bib` to ensure that the result will be understood by the `.bst` (BibTeX style) file used by the respective journal. Read more on this in Section 5.

(4) The top-level file `manuscript-sinum.tex`, its dependencies `content.tex` and `abstract.tex` and the accompanying custom BibTeX file `manuscript-sinum.bib` are now ready to be compiled. Use your favorite LaTeX editing system for this, or issue

```
pdflatex manuscript-sinum.tex
bibtex manuscript-sinum
pdflatex manuscript-sinum.tex
pdflatex manuscript-sinum.tex
```

or simply

```
latexmk -pdf manuscript-sinum.tex
```

on the command line, to obtain the final output `manuscript-sinum.pdf`.

(5) When writing your manuscript, you will frequently be making changes only to `content.tex` and then recompile the top-level file `manuscript-sinum.tex`. There is no need to re-run `ste prepare` over and over. You will only want to invoke `ste prepare` again when you make changes to the data file `manuscript.yaml` (e. g., change the title or add an author), when you add references to `my.bib`, or when you wish to format your manuscript for another outlet.

(6) Do not edit[3] any of the generated files `manuscript-sinum.tex` and `manuscript-sinum.bib`, since they will be overwritten next time you run `ste prepare`. Any changes necessary to `manuscript-sinum.tex` (e. g., adjustment of the manuscript's title) should be done by changing the data file `manuscript.yaml` and invoking `ste prepare`. Any changes necessary to `manuscript-sinum.bib` should likewise be done by changing `my.bib` and invoking `ste prepare`.

[3]To remind the user, all generated files will be write protected.

```yaml
# LaTeX related settings
latex:
  bibfiles: my.bib
  abstract: abstract.tex
  body: content.tex
  preamble: |-
    \usepackage{amsmath}
    \usepackage{cleveref}

# Information on your manuscript
manuscript:
  authors:
    - LMS
    - JIF
  title: Carrying the One Effectively

# Author data
authors:
  -
    familyname: Simpson
    givenname: Lisa M.
    institutions: [Harvard College, Springfield Heights]
    tag: LMS

  -
    familyname: Frink
    givenname: Jonathan I. Q.
    institutions: Springfield Heights
    tag: JIF

# Institution data
institutions:
  Harvard College: >-
    Harvard College,
    Cambridge,
    MA 02138,
    USA

  Springfield Heights: >-
    Springfield Heights Institute of Technology,
    Springfield,
    OR 97475,
    USA
```

FIGURE 2. A sample `manuscript.yaml` data file, similar to the one created by `ste start`.

3. Commands

The basic syntax of the Scoop Template Engine is

```
ste <command> [filter] [options]
```

More specifically, the following combinations exist.

command	meaning
`ste help`	show a help message and exit
`ste doc`	open this documentation file and exit
`ste version`	show the version information and exit
`ste list [filter]`	list available templates
`ste init [filter]`	download and initialize template resources
`ste start`	create files to start a new document
`ste prepare [options]`	prepare a document for LaTeX compilation

4. Data File

The data file, such as `manuscript.yaml` displayed in Figure 2, needs to be written in YAML[4] syntax. You will need to learn only the most basic YAML syntax features, and we introduce them here by example.[5]

`.yaml` data files for manuscripts in the Scoop Template Engine have five major keys: `control`, `latex`, `manuscript`, `authors` and `institutions`. Consequently, a `.yaml` file for the Scoop Template Engine has the following general layout:

```
control: ...
latex: ...
manuscript: ...
authors: ...
institutions: ...
```

Not all of these keys need to be present, and their order does not matter. For instance, in the `.yaml` file shown in Figure 2, the `control:` key is not present. In fact, a `.yaml` data file for the Scoop Template Engine may even be empty, although this does not constitute a very meaningful example.

We describe each of the five major keys in the following subsections.

[4]YAML Ain't Markup Language

[5]For a more thorough background, https://docs.ansible.com/ansible/latest/reference_appendices/YAMLSyntax.html and https://yaml.org/spec/ may be useful references.

4.1. **The `control`: part.** The `control`: part of a Scoop Template Engine `.yaml` file is used to control the behavior of `ste prepare`. It is organized as a dictionary. Each of the keys can be overriden by the corresponding setting on the command line.[6]

`.yaml` key	command line	meaning
`template`	`--template`	chooses the template
`outdir`	`--outdir`	sets the directory to write output to
`prefix`	`--prefix`	overrides the default output name
`nosuffix`	`--nosuffix`	suppresses the template suffix
`nocustombib`	`--nocustombib`	skips generation of custom BibTeX files
`nobib`	`--nobib`	prevents any `.bib` file based bibliography
`rulesfile`	`--rulesfile`	provides the substitution rules

The respective part of the `.yaml` data file thus looks like this:

```
control:
  template: ...
  outdir: ...
  prefix: ...
  nosuffix: ...
  nocustombib: ...
  nobib: ...
  rulesfile: ...
```

`template`: This selects the template to be used to generate the output. For instance, with

```
control:
  template: sinum
```

or equivalently,

```
ste prepare --template sinum
```

the template named `sinum` will be used.

`outdir`: This setting specifies the directory to which the generated files (such as `manuscript-sinum.tex` and `manuscript-sinum.bib`) are written. The default is to use the same directory from which you are calling `ste prepare` (the working directory). Using, for instance,

[6]The command line offers the additional `--datafile`, which selects the `.yaml` data file to be processed.

```
control:
  outdir: build
```

or equivalently,

```
ste prepare --outdir build
```

will write to the `build/` directory instead. This directory needs to exist and be writable to.

`prefix`: The default name of the output files (such as `manuscript-sinum.tex` and `manuscript-sinum.bib`) is composed of the name of the data file (`manuscript.yaml`) and the name of the template (`sinum`). Using the `prefix:` setting, you can override the first (`manuscript`) part of the output files' names. Using, for instance,

```
control:
  prefix: document
```

or equivalently,

```
ste prepare --prefix document
```

will use `document-sinum.tex` and `document-sinum.bib` as output file names, regardless of the name of the `.yaml` data file.

`nosuffix`: With this option, the template name will not become part of the output file names. Using, for instance, a data file `manuscript.yaml` and

```
control:
  nosuffix: True
```

or equivalently,

```
ste prepare --nosuffix
```

will use `manuscript.tex` and `manuscript.bib` as output file names.

`nocustombib`: Most journal templates use BibTeX to format article bibliographies. For those templates, `ste prepare` creates a custom BibTeX file (such as `manuscript-sinum.bib`). This custom BibTeX file will contain only the references which actually get cited in the document, and will make an effort to ensure it will be understood by the `.bst` (BibTeX style) file used by the respective journal. The sources of this single custom BibTeX file are the `.bib` files (such as `my.bib`) specified through the `bibfiles:` key; see `latex:` Section 4.2 below. Read more on bibliography handling in Section 5.

In any case, using

```
control:
  nocustombib: True
```

or equivalently,

```
ste prepare --nocustombib
```

will skip the generation of the custom BibTeX file, but if it already exists from a previous run, it will not be removed. Using the `nocustombib:` switch saves time during the execution of `ste prepare` and it can safely be used when a custom BibTeX file for the journal has been generated previously and the references cited have not changed since then.

`nobib:` In case you wish to use hand-crafted `\bibitem`s instead of a `.bib` file, use this switch in the `.yaml` data file

```
control:
  nobib: True
```

or equivalently,

```
ste prepare --nobib
```

It will disable the insertion of `\bibliography` (for BibTeX based templates) and `\printbibliography` (for BibLaTeX based templates) commands into the generated top-level file (such as `manuscriptsinum.tex`), which would cause bibliographies to be printed based on `.bib` files.

Using the `nobib:` switch implies the `nocustombib:` switch, and hence no custom BibTeX file will be generated since it would not be used anyway.

4.2. The `latex:` part.

The `latex:` part of a Scoop Template Engine `.yaml` file is used to specify LaTeX related settings. It is organized as a dictionary. The respective part of the `.yaml` data file looks like this:

```
latex:
  bibfiles: ...
  body: ...
  abstract: ...
  appendix: ...
  documentclassoptions: ...
  preamble: ...
  prepreamble: ...
  compatibility: ...
```

bibfiles: This key specifies the file name(s) of the bibliographic resources a manuscript is using. In case the respective template uses BibLaTeX, these files will be used directly via appropriate `\addbibresource` commands. In case the template uses uses BibTeX, the files specified through the `bibfiles:` key will serve as the basis to generate the custom BibTeX file.

The path of the `.bib` file(s) specified here is relative to the working directory. Use

```
latex:
  bibfiles: my.bib
```

or

```
latex:
  bibfiles:
  - my.bib
```

to specify a single bibliographic resource file and

```
latex:
  bibfiles:
    - my.bib
    - other.bib
```

for multiple files.

body: This key specifies the file name(s) of the `.tex` files holding the content of your manuscript. They will be `\input` into the generated top-level file (such as `manuscript-sinum.tex`) in the order of appearance. Use

```
latex:
  body: content.tex
```

or

```
latex:
  body:
    - content.tex
```

to specify a single content file and

```
latex:
  body:
    - content.tex
    - more-content.tex
```

for multiple files.

`abstract`: This key specifies the file name(s) of the `.tex` files holding the abstract of your manuscript. They will be `\input` into the generated top-level file (such as `manuscript-sinum.tex`) in the order of appearance. Use

```
latex:
  abstract: abstract.tex
```

or

```
latex:
  abstract:
    - abstract.tex
```

to specify a single abstract file and

```
latex:
  abstract:
    - abstract.tex
    - additional-abstract.tex
```

for multiple files.

`appendix`: This key specifies the file name(s) of the `.tex` files holding the appendix of your manuscript. They will be `\input` into the generated top-level file (such as `manuscript-sinum.tex`) in the order of appearance. Use

```
latex:
  appendix: appendix.tex
```

or

```
latex:
  appendix:
    - appendix.tex
```

to specify a single appendix file and

```
latex:
  appendix:
    - appendix.tex
    - additional-appendix.tex
```

for multiple files.

The convention that appendices are placed before the bibliography is applied throughout all templates, unless a journal explicitly suggests otherwise in their author guidelines.

`documentclassoptions`: Use this key to pass document class options to the document class used by the respective template. Be aware that not all document classes honor the same set of options. Use

```
latex:
  documentclassoptions: 12pt
```

to specify a single document class option and

```
latex:
  documentclassoptions:
    - 12pt
    - final
```

for multiple options.

`preamble`: Use this key to define a LaTeX preamble describing, e. g., the packages and custom commands you wish to use. For instance,

```
latex:
  preamble: |-
    \usepackage{amsmath}
    \usepackage{cleveref}
    \newcommand{\norm}[1]{\|#1\|}
```

will insert said commands into the preamble of the generated top-level file (such as `manuscript-sinum.tex`). The `preamble:` commands will be placed after the specification of the document class and also after the specification of packages required for the template (if any).

Note the `|-` specifier after the `preamble:` key, which is useful here. The `|` indicates YAML literal style, which preserves the content as is, including newlines. The (optional) `-` modifier removes any trailing newlines that may be following the `preamble:` block.

For preambles exceeding a few lines, it is recommended to place the respective commands into a custom file (such as `mypreamble.sty`) and reduce the content of the `preamble:` key in the `.yaml` file to

```
latex:
  preamble: |-
    \usepackage{mypreamble}
```

`prepreamble:` This key is similar to `preamble:` except that the respective content is inserted into the top-level file even before the specification of the document class. This is useful for instance in case a journal's `.cls` file loads a package and we would like to pass some options to it when it is loaded.

```
latex:
  prepreamble: |-
    \PassOptionsToPackage{lowtilde}{url}
```

`compatibility:` This key determines the degree of compatibility that Scoop Template Engine strives to achieve across all journal templates provided. The following compatibility levels are available:

setting	meaning
`none`	no effort is made to achieve any compatibility
`minimal`	ensures that a minimal set of commands is available
`packages`	ensures a certain set of packages can be used
`changes`	provides support for the `changes.sty` package
`theorems`	provides a set of theorem-like environments
`testing`	adjusts settings to facilitate template testing

If the compatibility level is not set, the default value `minimal` is used. To enable, for instance, the compatibility level `packages`, use

```
latex:
  compatibility: packages
```

or

```
latex:
  compatibility:
    - packages
```

in the `.yaml` data file. To enable both the compatibility levels `packages` and `theorems`, use

```
latex:
  compatibility:
    - packages
    - theorems
```

in the `.yaml` data file. Read more on this in Section 6.

4.3. **The `manuscript:` part.** The `manuscript:` part of a Scoop Template Engine `.yaml` file is used to specify essential information about your manuscript. It is organized as a dictionary. The respective part of the `.yaml` data file looks like this:

```
manuscript:
  authors: ...
  corresponding: ...
  title: ...
  subtitle: ...
  shorttitle: ...
  keywords: ...
  msc: ...
  date: ...
  funding: ...
  dedication: ...
```

`authors:` This key is used to specify the authors of a manuscript and their order. Authors are identified through their tag here, while their names, affiliations etc. are defined through the `authors:` key; see Section 4.4 for details. Use

```
manuscript:
  authors: LMS
```

or

```
manuscript:
  authors:
    - LMS
```

to specify a single author and

```
manuscript:
  authors:
    - LMS
    - JIF
```

for multiple authors.

`corresponding`: This key is used to specify which of the `authors`: are the corresponding authors. Use

```
manuscript:
  corresponding: LMS
```

or

```
manuscript:
  corresponding:
    - LMS
```

to specify a single corresponding author and

```
manuscript:
  corresponding:
    - LMS
    - JIF
```

for multiple corresponding authors.

`title`: Use this key to specify a manuscript's title.

```
manuscript:
  title: Carrying the One Effectively
```

Notice that in case the title contains a colon you will need to quote it to conform to YAML's syntax:

```
manuscript:
  title: "Carrying the One: Theory and Practice"
```

`subtitle`: Use this key to specify a manuscript's subtitle.

```
manuscript:
  subtitle: Theory and Practice
```

shorttitle: Use this key to specify a manuscript's short title. This is used as the running title by many templates.

```
manuscript:
  shorttitle: Carrying the One
```

If unspecified or empty, `shorttitle:` will be set identical to the content of the `title:` key.

keywords: This key is used to specify a manuscript's keywords. Use

```
manuscript:
  keywords: linear systems
```

or

```
manuscript:
  keywords:
    - linear systems
```

to specify a single keyword and

```
manuscript:
  keywords:
    - linear systems
    - iterative solver
```

for multiple keywords.

msc: This key is used to specify Mathematics Subject Classification (MSC) codes. Use

```
manuscript:
  msc: 65F10 # iterative methods for linear systems
```

or

```
manuscript:
  msc:
    - 65F10 # iterative methods for linear systems
```

to specify a single MSC code and

```
manuscript:
  msc:
    - 65F10 # iterative methods for linear systems
    - 65F08 # preconditioners for iterative methods
```

for multiple MSC codes. The optional comments merely serve as a reminder of the respective category.

`date:` Use this key to specify the date of the manuscript.

```
manuscript:
  date: \today
```

`funding:` This key is used to provide information about funding related to a manuscript.

```
manuscript:
  funding: >-
    This work was kindly supported through free coffee
    by the Springfield Heights Coffee Shop.
```

Note the `>-` specifier after the `funding:` key, used here only for formatting purposes. The `>` indicates YAML folding style, which preserves the content as is but converts newlines to spaces. The (optional) `-` modifier removes any trailing newlines that may be following the `funding:` block.

`dedication:` This key is used to specify a manuscript's dedication.

```
manuscript:
  dedication: >-
    This work is dedicated to someone special
    on the occasion of their 60th birthday.
```

4.4. **The `authors:` part.** The `authors:` part of a Scoop Template Engine `.yaml` file is used to specify information about the authors. It is organized as a list of dictionaries. The respective part of the `.yaml` data file looks like this:

```
authors:
  - # some author's data
    givenname: ...
    familyname: ...
    institutions: ...
    emails: ...
    urls: ...
    tag: ...
    color: ...

  - # another author's data
    givenname: ...
    familyname: ...
    institutions: ...
    emails: ...
    urls: ...
    tag: ...
    color: ...
```

The order in which the authors appear in this section of the `.yaml` data file is irrelevant. The order in which authors appear in the manuscript is determined through the order in which author tags appear in the `manuscript:` Section 4.3 above.

`givenname:` This key is used to specify an author's given name(s).

```
authors:
  -
    givenname: Lisa M.
```

`familyname:` This key is used to specify an author's family name(s).

```
authors:
  -
    familyname: Simpson
```

`institutions:` This key is used to specify the institutions an author is affiliated with. To identify an institution, you are free to choose an arbitrary identifier/tag. The institutions themselves are described through the `institutions:` key; see Section 4.5 below. Use

```
authors:
  -
    institutions: Springfield Heights
```

or

```
authors:
  -
```

```
    institutions:
      - Springfield Heights
```

to specify an author affiliated with a single institution and

```
authors:
  -
    institutions:
      - Harvard College
      - Springfield Heights
```

for an author affiliated with multiple institutions.

`emails`: This key is used to specify an author's email address(es). Use

```
authors:
  -
    emails: lms@dayrep.com
```

or

```
authors:
  -
    emails:
      - lms@dayrep.com
```

for an author who wants a single email address listed and

```
authors:
  -
    emails:
      - lms@dayrep.com
      - lisamsimpson@gustr.com
```

for an author who wants multiple email addresses listed.

`urls`: This key is used to specify the URL(s) of an author's home page(s). Use

```
authors:
  -
    urls: https://example.com/lms.html
```

or

```
authors:
  -
    urls:
      - https://example.com/lms.html
```

for an author who wants a single home page listed and

```
authors:
  -
    urls:
      - https://example.com/lms.html
      - https://gitlab.com/scoopgroup-public
```

for an author who wants multiple home pages listed.

`tag:` This key is used to specify an author's tag. This tag is used to identify an author as an author or corresponding author of a manuscript as described in the `authors:` Section 4.4 above.

```
authors:
  -
    tag: LMS
```

`color:` This key is used to specify an author's color. It is used exclusively in combination with compatibility level `changes`. In this setting, the `color:` specification defines an author's color markup in the commands from the `changes.sty` package. The compatibility level `changes` is meant to facilitate the use of the `changes.sty` package, which can be used to highlight changes in a manuscript, see Section 6.4.

```
authors:
  -
    color: blue!80!black
```

Colors can be specified as in `xcolor.sty`'s `\colorlet` command; see Section 6.4 for details.

4.5. **The `institutions:` part.** The `institutions:` part of a Scoop Template Engine `.yaml` data file is used to specify information about the authors' institutions. It is organized as a dictionary whose keys comprise the institution identifiers used in the `authors:` Section 4.4 above.

The respective part of the `.yaml` data file looks like this:

```
institutions:
  ...
  ...
```

To specify an institution's address, use

```
institutions:
  Springfield Heights: >-
    Springfield Heights Institute of Technology,
    Springfield, OR 97475,
    USA
```

The `>` indicates YAML folding style, which preserves the content as is but converts newlines to spaces. The (optional) `-` modifier removes any trailing newlines that may be following the respective block.

5. Bibliography

The Scoop Template Engine supports all three possibilities to handle bibliographies for manuscripts.

5.1. BibLaTeX.

Some templates templates process their bibliography using BibLaTeX, notably `amspreprint`; see Section 11. For these templates, the `.bib` file(s) you specify in the `.yaml` data file using, e. g.,

```
latex:
  bibfiles: my.bib
```

will be added directly as bibliographic resources to the generated top-level file (such as `manuscript-preprint.tex`) using `\addbibresource`. Moreover, a `\printbibliography` command will be inserted at the end of the generated top-level file (such as `manuscript-sinum.tex`).

Compile the generated top-level file using, e. g., `pdflatex` and `biber`, or more conveniently, using `latexmk -pdf`. In case you are using a LaTeX editor, notice that you may need to manually switch the processing pipeline from `bibtex` to `biber`.

For templates using BibLaTeX, the `nocustombib:` setting has no effect since no custom bibilography is generated anyway.

5.2. BibTeX.

All other journal templates use BibTeX to process bibliographies. For those templates, the `.bib` file(s) you specify in the `.yaml` data file (such as `my.bib`) will be processed behind the scenes by the Scoop Template Engine (actually, by its companian `spbf`) to produce a custom BibTeX file (such as `manuscript-sinum.bib`) every time you run `ste prepare`.

This custom BibTeX file is then used in the following way. A command similar to the following will be inserted into the generated top-level file (such as `manuscript-sinum.tex`):

```
\IfFileExists{./manuscript-sinum.bib}{%
\bibliography{./manuscript-sinum.bib}
}{%
\bibliography{my.bib}
}
```

This causes the custom BibTeX file to be used to produce the bibliography. However, if, for any reason, the custom BibTeX file does not exist, the unprocessed `.bib` file(s) specified in the `latex`: Section 4.2 will be used as a fallback.

The reasoning behind using custom BibTeX files is two-fold. First, the custom `.bib` file will contain only those references which actually get cited in the document. Therefore, you may use, for instance, one central `.bib` file for several of your publications and yet when you submit your manuscript to a journal, include just the generated BibTeX file (such as `manuscript-sinum.bib`) which contains only the required entries from your central `.bib` file. Second, the `.bib` file(s) you specify in the `.yaml` data file (such as `my.bib`) may contain entry types and fields which are not supported by the `.bst` (BibTeX style) file used by the respective journal, resulting in missing information or even errors when it gets processed directly. Moreover, at the very least, entries in the `.bib` file in the UTF8 character set need to be recoded into LaTeX accented characters, e. g., ä becomes `\"{a}`.

All of this is carried out through Scoop Prepare BibTeX File (`spbf`), which is bundled with the Scoop Template Engine but may also be used independently. We therefore describe it in a separate document (try `spbf doc`).

5.3. **bibitems.** In case you decide to not use BibLaTeX nor BibTeX (depending on the template) but go with hand-crafted `\bibitem`s instead, you can do that as well. In this case, add your

```
\begin{thebibliography}{99}
\bibitem{...}
\bibitem{...}
\end{thebibliography}
```

environment to the appropriate place in the file holding the content of your manuscript (such as `content.tex`). You will then need to specify

```
control:
  nobib: True
```

in your `.yaml` data file, or equivalently, use

```
ste prepare --nobib
```

to avoid the insertion of any `\bibliography` or `\printbibliography` commands into the top-level file.

6. Compatibility

Being able to format a manuscript for a specific journal unfortunately does not mean that the LaTeX code in your content file (such as `content.tex`) immediately compiles without errors. For instance, the `.cls` files of some journal might load a specific package which may turn out to be incompatible with other packages you wish to use. This issue has nothing to do with the Scoop Template Engine but rather the design of some journals' `.cls` files.

The Scoop Template Engine makes an effort to support the user in writing their document in an outlet independent way. To this end, it offers various levels of compatibility support, which can be set using

```
latex:
  compatibility: ...
```

in the `.yaml` data file. These levels are `none`, `minimal`, `packages`, `changes` and `theorems`. They are explained below. The default compatibility level is `minimal`. Compatibility levels can be combined by specifying them as a list, using, e. g.,

```
latex:
  compatibility:
    - packages
    - changes
```

The implementation of compatibility levels is done as follows. For templates requiring an effort to ensure compatibility, the Scoop Template Engine provides a preamble file which takes the desired compatibility level as a package option. Code loading this preamble is automatically inserted when `ste prepare` generates the top-level file (such as `manuscript-sinum.tex`). In fact, since compatibility code may need to be inserted before the document class is loaded, after the document class is loaded but before the user-defined `preamble:` is executed, or after the user-defined `preamble:`, the

Scoop Template Engine provides up to three separate such preamble files for every template. They are named according to the pattern `prepreamble-sinum.sty`, `preamble-sinum.sty` and `postpreamble-sinum.sty`. There is no need for the user to change those files or their content.

A few journal templates are distributed by the publisher with obvious issues, for instance, incorrect file names, errors and typos in their `.bst` (BibTeX style) file, missing `.bst` files, etc. We take the liberty to fix those issues directly during `ste init`, by having the respective `init.py` script modify the offending files, regardless of the `compatibility:` level.

6.1. Compatibility level `none`.

In this compability level, no effort is made to achieve any degree of compatibility of the journal's template with the user's LaTeX code. This is not recommended and it may very well cause the generated top-level file (such as `manuscript-sinum.tex`) fail to compile. For instance, some journals' `.bst` files define a `\doi` command which does not handle DOIs with underscores. Moreover, the custom BibTeX file may contain `\href` or `\url` commands and thus rely on these commands to be available, which is not the case for some journal templates unless, e. g., the user explicitly loads the `hyperref.sty` package (through `preamble:`).

6.2. Compatibility level `minimal`.

This compatibility level ensures a minimal degree of compatibility between the journal's template and the user's LaTeX code. For illustration purposes, here are some issues and fixes which get applied for some templates in compatibility level `minimal`:

- Rudimentary `\href` and `\url` commands are provided at the beginning of the document. In case those commands already exist they will not be overwritten.

- In case the journal's `.bst` file causes the definition of a `\doi` command to be written into the `.bbl` file which does not handle underscores, that `\doi` command is replaced by an improved version.

- Settings related to `latin1` encodings are overridden by the `utf8` encoding.

6.3. Compatibility level `packages`.

Among the hundreds of available LaTeX packages, some will likely cause issues with some of the journal templates. It is not our intention to track let alone fix all possible incompatibilites that might occur. Instead, we make it our goal to ensure compatibility with of the following, frequently used packages:

- `inputenc.sty`, specifically `\usepackage[utf8]{inputenc}`
- `fontenc.sty`, specifically `\usepackage[T1]{fontenc}`
- `changes.sty`
- `mathtools.sty` and thus `amsmath.sty`

- `tikz.sty`
- `hyperref.sty`
- `cleveref.sty`

Clearly, this selection of these packages is subjective and expresses the author's own preferences.

In case your favorite package is not listed above, this does not necessarily mean that tweaking is required to make it function with the templates you intend to use. On the other hand, the packages listed above are being systematically tested for compatibility with all templates provided by the Scoop Template Engine when the compatibility setting `packages` is used.

The compatibility level `packages` includes all settings made in compatibility level `minimal`.

It is important to mention that none of the above packages are loaded automatically by any generated top-level file (such as `manuscript-sinum.tex`). In case you would like to use a package, you will have to specify the respective command such as `\usepackage[utf8]{inputenc}` through the `preamble:` key of the `.yaml` data file.

For illustration purposes, here are some issues and corresponding fixes which get applied for some templates in compatibility level `packages`:

- The `\todo` and `\comment` commands or environments are reset to undefined since the `changes.sty` package defines its own commands with the same name.

- The `program.sty` package loaded by some of the `.cls` files makes `|` an active character, which causes `tikz.sty` to fail loading and consequently all packages relying on `tikz.sty`, including `changes.sty`.[7] Therefore, the role of `|` needs to be reset outside of `program` environments.

- A `part` counter needs to be defined for compatibility with `cleveref.sty`.[8]

- The `\fpage` (reference to first page) needs to be redefined when `hyperref.sty` is used.[9]

6.4. **Compatibility level `changes`.** This compatibility level is meant to facilitate the use of the `changes.sty` package, which can be used to highlight changes in a manuscript. When this compatibility level is enabled, LaTeX code is automatically inserted into the generated top-level file (such as `manuscript-sinum.tex`) which configures every author of the manuscript

[7] https://tex.stackexchange.com/questions/619280/
[8] https://tex.stackexchange.com/questions/482167/
[9] https://tex.stackexchange.com/questions/371955/

via an appropriate \definechangesauthor commands, provided that the changes.sty actually has been loaded by the user.

For instance,

```
authors:
  -
    givenname: Lisa M.
    familyname: Simpson
    tag: LMS
    color: blue
```

will result in the following command to be inserted into the top-level file's preamble:

```
\definechangesauthor[name={Lisa M. Simpson}, color={blue}]{LMS}
```

Consequently, commands such as

```
\added[id=LMS]{added text}
\deleted[id=LMS]{deleted text}
\replaced[id=LMS]{new text}{old text}
```

will then be available. Please refer to the documentation of the changes.sty package for more information on how to use these commands.

Notice that selecting the compatibility level changes does not actively load the changes.sty package. You will still need to request it, e. g., via

```
latex:
  preamble: |-
    \usepackage{changes}
```

in your .yaml data file. Also, you can set the color for anonymous markups in this way:

```
latex:
  preamble: |-
    \usepackage[defaultcolor = orange!80!black]{changes}
```

Notice also that the changes.sty package honors the final document class option, so when you specify

```
latex:
  documentclassoptions:
    - final
```

all markup from the `changes.sty` package will be suppressed.

When no `color:` is specified for an author, the next color out of a palette of fallback colors will be used. Color specification is possible within the limits of the `\definechangesauthor` command, which in turn uses `xcolor.sty`'s command `\colorlet`. Consequently, color specifications as in

```
authors:
  -
    color: blue!80!black
```

are valid. Also, you can define custom colors in the preamble, e. g.,

```
latex:
  preamble: |-
    \definecolor{myorange}{HTML}{EE7733}
```

and use them as an author's markup color:

```
authors:
  -
    color: myorange
```

Finally, in order to access, for instance, `xcolor.sty`'s `CornflowerBlue` color, you could add

```
latex:
  prepreamble: |-
    \PassOptionsToPackage{svgnames}{xcolor}
```

to your `.yaml` data file and use

```
authors:
  -
    color: CornflowerBlue
```

6.5. **Compatibility level `theorems`.** The compatibility level `theorems` is used to provide a set of theorem-like environments for each template. Moreover, we make sure that these environments are supported by the `cleveref.sty` package (although this package is not loaded automatically and should be requested through the `preamble:` setting in your `.yaml` data file if desired). The theorem-like environments provided are

- `assumption`
- `corollary`
- `definition`

- `example`
- `lemma`
- `proposition`
- `remark`
- `theorem`

as well as their starred variants `assumption*`, `corollary*`, `definition*`, `example*`, `lemma*`, `proposition*`, `remark*`, `theorem*`. The policy of implementation is to try and depart as little as possible from the way the `.cls` file of the template provided by the respective publisher handles theorem-like environments. This implies that for some templates, all theorem-like environments share a common counter (e. g., Definition 1 may be followed by Theorem 2), while for others, they are numbered independently (e. g., Definition 1 is followed by Theorem 1). Moreover, for some templates, theorem-like environments are numbered per section (e. g., Theorem 3.1) while for others they are not. In some cases, unnumbered theorem-like environments are not available and then the starred environments are mapped to their respective numbered counterparts.

6.6. **Compatibility level `testing`.** This compatibility level is meant to be used to facilitate the testing of templates. It is of no particular use during production. For instance, it turns off the two-column layout in some templates, which otherwise may impede the verification of the list of references.

7. Best Practices

(1) Put your preamble in a separate file, e. g., `mypreamble.sty`, and use

```
latex:
  preamble: |-
    \usepackage{mypreamble}
```

This avoids the need to re-run `ste prepare` whenever you need to make changes, e. g., add a package.

(2) When writing the first draft of a manuscript, you may prefer to use a template which uses BibLaTeX rather than BibTeX, such as the `amspreprint` template. This way, references you add to the `.bib` file (such as `my.bib`, as specified through the `bibfiles:` setting) will immediately be available and do not require a pass of `ste prepare`.

By contrast, when you are writing using a BibTeX based template, you will either want to use `ste prepare` with the `--nocustombib` switch (which, however, may cause issues in case your `.bib` entries contain features not supported by the journal's `.bst` file), or else you will have to re-run `ste prepare` every time you `\cite` a new reference in order to re-create the BibTeX file customized to the journal's `.bst` file.

(3) The available text width can vary widely between journals. When formatting a manuscript for a specific journal, it may therefore be necessary to fine-tune breaks in formulas etc. If you wish to format your manuscript to look decent with several templates at once (such as the `amspreprint` template and a journal template), conditional formatting sometimes comes in handy which takes decisions based on the `.cls` file loaded. Here is an example of such a construction for an equation which fits on a single line in the `cocv` template (which uses the document class `cocv.cls`) but requires a line break in templates based on the `svjour3.cls` document class.

```
\makeatletter
\ltx@ifclassloaded{cocv}{%
\begin{equation}
...
\end{equation}
}{}
\ltx@ifclassloaded{svjour3}{%
\begin{multline}
... \\
...
\end{multline}
}{}
\makeatother
```

You will need to add `\usepackage{ltxcmds}` to your `preamble:` in order for the command `\ltx@ifclassloaded` to be available.

(4) When you are writing a manuscript together with co-authors who are not using the Scoop Template Engine themselves, it is up to you to generate the top-level `.tex` and `.bib` files and share them. When you are using a version control system, put the generated top-level files under version control too. This makes it easy to spot if a co-author accidentally edits those generated files manually, instead of updating the `.yaml` data file and re-running `ste prepare`.

(5) When you find yourself needing to pass lots of arguments to `ste`, such as

```
ste prepare --template amspreprint --outdir build \
  --prefix paper --nosuffix
```

then you may prefer to configure these settings in the `.yaml` data file instead, e. g.,

```
control:
  template: preprint
  outdir: build
  prefix: paper
  nosuffix: True
```

so that you can invoke the script simply as `ste prepare` and still override these settings from the command line if needed.

8. Contributing

If you are using the Scoop Template Engine, I would like to hear from you at roland.herzog@iwr.uni-heidelberg.de. If you would like to help make it better, there are various ways in which you can contribute. Your help is greatly appreciated!

(1) You may file bug reports and feature requests via `https://gitlab.com/scoopgroup-public/scoop-template-engine/-/issues`.

(2) You may fork the project from `https://gitlab.com/scoopgroup-public/scoop-template-engine`, make improvements yourself and send a merge request.

9. Extensions

9.1. New Journal Templates.

In case you would like to develop your own templates, for instance a preprint style different from the one provided or a template for a new journal, it's best to start from one of the existing templates and modify it to your needs. The main challenge in developing a new template is usually to get the formatting of authors and their affiliations in the way required by the journal right.

To see which templates are already available, you may want to run the integration test locally on your machine, which compiles a test manuscript for all 106 out of the 106 templates which are actually different (i. e., templates of file type; see Section 11). To this end, go to the `manuscripts/IntegrationTest/` directory and run `./prepare-all.sh` to generate all `.tex` and `.bib` files, then `./compile-all.sh` to produce the associated `.pdf` files.

If you conclude that you need to implement a new variant of author and institution formatting, find a template which is close to what you need and take a look at the implementation of the respective formatting function in `manuscripts/manusripts.py` as a starting point.

9.2. **New Template Classes.** All templates currently shipping with the Scoop Template Engine are meant for publications in scientific journals. They all reside in the `manuscripts/` directory, along with the corresponding `rulesfile`: `manuscripts/manuscripts.py`. However, it is perfectly possible to use the Scoop Template Engine for other LaTeX document types, such as presentations, theses, books, etc.

Other document types will require other data structures. For instance, for a presentation you might want to include logos of the authors' institutions but you do not need an abstract. If you decide to write a template for a document type different from a journal manuscript, it is recommended to create a new directory (such as `presentations/`, on the same level as `manuscripts/`) and create a new rules file (such as `presentations/presentations.py`) there.

You will probably be able to reuse some code from `manuscripts.py` for your project. While you are writing the code doing the formatting and designing your new templates, you will immediately be able to use them. There is no need nor mechanism to register them with the Scoop Template Engine. Try, for instance,

```
scoop-template-engine.py --listtemplates presentations
```

to see only templates inside the `presentations/` directory.

I would like to hear about your template projects at roland.herzog@iwr.uni-heidelberg.de. Also, feel free to fork the Scoop Template Engine from `https://gitlab.com/scoopgroup-public/scoop-template-engine` and send a merge request.

10. Known Issues, Limitations

(1) The `date:` is currently honored only by few templates.

(2) There is no support to provide and typeset an ORCID yet.

(3) There is no support yet to provide a unified choice of packages for the typesetting of algorithms, such as `algpseudocode` and `algorithm2e`.

(4) Templates are currently tested with `pdflatex` on the TeX Live 2022 distribution only but should work also on TeX Live 2019 and later. There is currently no support for `xelatex` or other TeX engines. Some templates do compile with `xelatex` but not all of them.

(5) Funding and dedication information is currently in the scope of the entire manuscript, not on a per-author basis.

(6) There is currently no support for journals operating a double blind peer review process.

(7) There is currently no support to typeset research highlights which some journals require authors to explicitly list.

(8) Author biographies and pictures which some journals offer to include are not yet supported.

In case you encounter further problems, please create an issue at `https://gitlab.com/scoopgroup-public/scoop-template-engine/-/issues`.

11. List of Templates

The following table lists all templates currently supported.

- The `Template` column contains the name of the respective template as it can be specified `--template` or using the

  ```
  control:
    template: ...
  ```

  setting in the data file.

- The `BibLaTeX` column indicates whether the respective template processes `.bib` files using BibLaTeX or BibTeX.
  - `*` means BibLaTeX is used; see Section 5.1 for details.
  - `-` means that BibTeX is used. This implies that the `ste` will create a custom BibTeX file (unless `--nocustombib` is being used); see Section 5.2 for details.

- The `Journal Name` column contains the full name of the journal the respective template is meant for.

- The `Template Location` column shows where the respective template resides.

- The `Type` column shows the type of template.
  - `F` means that the template is a regular file.
  - `L` means that the template is a symbolic link (i. e., identical to another file-type template).

Template	BibLaTeX	Journal Name	Template Location	Type
acdm	-	Advances in Continuous and Discrete Models	manuscripts/Springer/bmcart.cls/template-acdm.tex	F
acom	-	Advances in Computational Mathematics	manuscripts/Springer/sn-jnl.cls/template-acom.tex	F
ag	*	Algebraic Geometry	manuscripts/EMS/ag/template-ag.tex	F
amo	-	Applied Mathematics & Optimization	manuscripts/Springer/sn-jnl.cls/template-amo.tex	F
amspreprint	*	amsart preprint	manuscripts/scoop/template-amspreprint.tex	F
bbiici	*	Bioinspiration & Biomimetics	manuscripts/IOP/template-bbiici.tex	F
bioffn	*	Biofabrication	manuscripts/IOP/template-bioffn.tex	F
bit	-	BIT Numerical Mathematics	manuscripts/Springer/bit/template-bit.tex	F
bmbucs	*	Biomedical Materials	manuscripts/IOP/template-bmbucs.tex	F
bpeeae	*	Biomedical Physics & Engineering Express	manuscripts/IOP/template-bpeeae.tex	F
bvp	-	Boundary Value Problems	manuscripts/Springer/bmcart.cls/template-bvp.tex	F
calcolo	-	Calcolo	manuscripts/Springer/sn-jnl.cls/template-calcolo.tex	F
camcos	-	Communications on Applied Mathematics and Computation	manuscripts/Springer/sn-jnl.cls/template-camcos.tex	F
cm	-	Computational Mechanics	manuscripts/Springer/sn-jnl.cls/template-cm.tex	F
coam	-	Computational and Applied Mathematics	manuscripts/Springer/sn-jnl.cls/template-coam.tex	F
coap	-	Computational Optimization and Applications	manuscripts/Springer/sn-jnl.cls/template-coap.tex	F
cqgrdg	*	Classical and Quantum Gravity	manuscripts/IOP/template-cqgrdg.tex	F
cvpde	-	Calculus of Variations and Partial Differential Equations	manuscripts/Springer/sn-jnl.cls/template-cvpde.tex	F
dcg	-	Discrete & Computational Geometry	manuscripts/Springer/sn-jnl.cls/template-dcg.tex	F
dmatb7	*	2D Materials	manuscripts/IOP/template-dmatb7.tex	F
ejphd4	*	European Journal of Physics	manuscripts/IOP/template-ejphd4.tex	F
ejssbg	*	ECS Journal of Solid State Science and Technology	manuscripts/IOP/template-ejssbg.tex	F
em	*	Elemente der Mathematik	manuscripts/EMS/em/template-em.tex	F
erc	*	Environmental Research Communications	manuscripts/IOP/template-erc.tex	F
ercl	*	Environmental Research: Climate	manuscripts/IOP/template-ercl.tex	F
ere	*	Environmental Research: Ecology	manuscripts/IOP/template-ere.tex	F
erenbl	*	Engineering Research Express	manuscripts/IOP/template-erenbl.tex	F
erh	*	Environmental Research: Health	manuscripts/IOP/template-erh.tex	F
erisal	*	Environmental Research: Infrastructure and Sustainability	manuscripts/IOP/template-erisal.tex	F
erlnal	*	Environmental Research Letters	manuscripts/IOP/template-erlnal.tex	F
esltac	*	Electronic Structure	manuscripts/IOP/template-esltac.tex	F
fcsuah	*	Functional Composites and Structures	manuscripts/IOP/template-fcsuah.tex	F
focm	-	Foundations of Computational Mathematics	manuscripts/Springer/sn-jnl.cls/template-focm.tex	F
fpelab	*	Flexible and Printed Electronics	manuscripts/IOP/template-fpelab.tex	F
ijemkf	*	International Journal of Extreme Manufacturing	manuscripts/IOP/template-ijemkf.tex	F
ijot	-	International Journal of Thermophysics	manuscripts/Springer/sn-jnl.cls/template-ijot.tex	F
inge	-	Information Geometry	manuscripts/Springer/sn-jnl.cls/template-inge.tex	F
ip	*	Inverse Problems	manuscripts/IOP/template-ip.tex	F
isoccm	*	IOP SciNotes	manuscripts/IOP/template-isoccm.tex	F
jbrobw	*	Journal of Breath Research	manuscripts/IOP/template-jbrobw.tex	F
jcapbp	*	Journal of Cosmology and Astroparticle Physics	manuscripts/IOP/template-jcapbp.tex	F
jcomel	*	Journal of Physics: Condensed Matter	manuscripts/IOP/template-jcomel.tex	F
jia	-	Journal of Inequalities and Applications	manuscripts/Springer/bmcart.cls/template-jia.tex	F

Template	BibLaTeX	Journal Name	Template Location	Type
jionas	*	Journal of Instrumentation	manuscripts/IOP/template-jionas.tex	F
jmi	-	Journal of Mathematics in Industry	manuscripts/Springer/bmcart.cls/template-jmi.tex	F
jmiv	-	Journal of Mathematical Imaging and Vision	manuscripts/Springer/sn-jnl.cls/template-jmiv.tex	F
jmmiez	*	Journal of Micromechanics and Microengineering	manuscripts/IOP/template-jmmiez.tex	F
jneiez	*	Journal of Neural Engineering	manuscripts/IOP/template-jneiez.tex	F
jogo	-	Journal of Global Optimization	manuscripts/Springer/sn-jnl.cls/template-jogo.tex	F
joopca	*	Journal of Optics	manuscripts/IOP/template-joopca.tex	F
jota	-	Journal of Optimization Theory and Applications	manuscripts/Springer/jota/template-jota.tex	F
jpamb5	*	Journal of Physics A: Mathematical and Theoretical	manuscripts/IOP/template-jpamb5.tex	F
jpapbe	*	Journal of Physics D: Applied Physics	manuscripts/IOP/template-jpapbe.tex	F
jpapeh	*	Journal of Physics B: Atomic, Molecular and Optical Physics	manuscripts/IOP/template-jpapeh.tex	F
jpcofp	*	Journal of Physics Communications	manuscripts/IOP/template-jpcofp.tex	F
jpcogq	*	Journal of Physics: Complexity	manuscripts/IOP/template-jpcogq.tex	F
jpeoey	*	Journal of Physics: Energy	manuscripts/IOP/template-jpeoey.tex	F
jpgped	*	Journal of Physics G: Nuclear and Particle Physics	manuscripts/IOP/template-jpgped.tex	F
jpmoc4	*	Journal of Physics: Materials	manuscripts/IOP/template-jpmoc4.tex	F
jppokr	*	Journal of Physics: Photonics	manuscripts/IOP/template-jppokr.tex	F
jrprea	*	Journal of Radiological Protection	manuscripts/IOP/template-jrprea.tex	F
jsc	-	Journal of Scientific Computing	manuscripts/Springer/svjour3.cls/template-jsc.tex	F
jsmtc6	*	Journal of Statistical Mechanics: Theory and Experiment	manuscripts/IOP/template-jsmtc6.tex	F
maana	-	Mathematische Annalen	manuscripts/Springer/sn-jnl.cls/template-maana.tex	F
mafeb2	*	Methods and Applications in Fluorescence	manuscripts/IOP/template-mafeb2.tex	F
mathbio	-	Journal of Mathematical Biology	manuscripts/Springer/sn-jnl.cls/template-mathbio.tex	F
mathprog	-	Mathematical Programming	manuscripts/Springer/sn-jnl.cls/template-mathprog.tex	F
mlstck	*	Machine Learning: Science and Technology	manuscripts/IOP/template-mlstck.tex	F
mmor	-	Mathematical Methods of Operations Research	manuscripts/Springer/sn-jnl.cls/template-mmor.tex	F
mmuabd	*	Multifunctional Materials	manuscripts/IOP/template-mmuabd.tex	F
mpc	-	Mathematical Programming Computation	manuscripts/Springer/svjour3.cls/template-mpc.tex	F
mqtaaz	*	Materials for Quantum Technology	manuscripts/IOP/template-mqtaaz.tex	F
mreac3	*	Materials Research Express	manuscripts/IOP/template-mreac3.tex	F
ms	-	Mathematical Sciences	manuscripts/Springer/sn-jnl.cls/template-ms.tex	F
msmeeu	*	Modelling and Simulation in Materials Science and Engineering	manuscripts/IOP/template-msmeeu.tex	F
mstcep	*	Measurement Science and Technology	manuscripts/IOP/template-mstcep.tex	F
mtrgau	*	Metrologia	manuscripts/IOP/template-mtrgau.tex	F
nceecn	*	Neuromorphic Computing and Engineering	manuscripts/IOP/template-nceecn.tex	F
neaxa4	*	Nano Express	manuscripts/IOP/template-neaxa4.tex	F
nfaub3	*	Nano Futures	manuscripts/IOP/template-nfaub3.tex	F
njopfm	*	New Journal of Physics	manuscripts/IOP/template-njopfm.tex	F
nnoter	*	Nanotechnology	manuscripts/IOP/template-nnoter.tex	F
nonle5	*	Nonlinearity	manuscripts/IOP/template-nonle5.tex	F
nufuau	*	Nuclear Fusion	manuscripts/IOP/template-nufuau.tex	F
numa	-	Numerische Mathematik	manuscripts/Springer/sn-jnl.cls/template-numa.tex	F
numalg	-	Numerical Algorithms	manuscripts/Springer/sn-jnl.cls/template-numalg.tex	F

Template	BibLaTeX	Journal Name	Template Location	Type
opte	-	Optimization and Engineering	manuscripts/Springer/svjour3.cls/template-opte.tex	F
pberb8	*	Progress in Biomedical Engineering	manuscripts/IOP/template-pberb8.tex	F
pbhiat	*	Physical Biology	manuscripts/IOP/template-pbhiat.tex	F
pdea	-	Partial Differential Equations and Applications	manuscripts/Springer/sn-jnl.cls/template-pdea.tex	F
perndg	*	Progress in Energy	manuscripts/IOP/template-perndg.tex	F
pheda7	*	Physics Education	manuscripts/IOP/template-pheda7.tex	F
phmba7	*	Physics in Medicine & Biology	manuscripts/IOP/template-phmba7.tex	F
phstbo	*	Physica Scripta	manuscripts/IOP/template-phstbo.tex	F
pmeae3	*	Physiological Measurement	manuscripts/IOP/template-pmeae3.tex	F
ppcfet	*	Plasma Physics and Controlled Fusion	manuscripts/IOP/template-ppcfet.tex	F
prelcz	*	Plasma Research Express	manuscripts/IOP/template-prelcz.tex	F
psteeu	*	Plasma Sources Science and Technology	manuscripts/IOP/template-psteeu.tex	F
qstuah	*	Quantum Science and Technology	manuscripts/IOP/template-qstuah.tex	F
rlm	*	Rendiconti Lincei - Matematica e Applicazioni	manuscripts/EMS/rlm/template-rlm.tex	F
rpphag	*	Reports on Progress in Physics	manuscripts/IOP/template-rpphag.tex	F
smster	*	Smart Materials and Structures	manuscripts/IOP/template-smster.tex	F
ssteet	*	Semiconductor Science and Technology	manuscripts/IOP/template-ssteet.tex	F
stmpcw	*	Surface Topography: Metrology and Properties	manuscripts/IOP/template-stmpcw.tex	F
sustef	*	Superconductor Science and Technology	manuscripts/IOP/template-sustef.tex	F
svva	-	Set-Valued and Variational Analysis	manuscripts/Springer/sn-jnl.cls/template-svva.tex	F

(R. Herzog) Interdisciplinary Center for Scientific Computing, Heidelberg University, 69120 Heidelberg, Germany

Email address: `roland.herzog@iwr.uni-heidelberg.de`